

ESTTA Tracking number: **ESTTA416233**

Filing date: **06/24/2011**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE  
BEFORE THE TRADEMARK TRIAL AND APPEAL BOARD

Proceeding	91193335
Party	Defendant RStudio, Inc.
Correspondence Address	CHARLES E. WEINSTEIN FOLEY HOAG LLP 155 SEAPORT BLVD, STE 1600 BOSTON, MA 02210-2600 UNITED STATES ARufo@foleyhoag.com, JHuston@foleyhoag.com, USTRademark@foleyhoag.com, cweinstein@foleyhoag.com, jjarvis@foleyhoag.com
Submission	Testimony For Defendant
Filer's Name	Anthony E. Rufo
Filer's e-mail	arufo@foleyhoag.com
Signature	/Anthony E. Rufo/
Date	06/24/2011
Attachments	003 Exhibits 8 part 2.pdf ( 172 pages )(7034491 bytes )

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE  
BEFORE THE TRADEMARK TRIAL AND APPEAL BOARD

EMBARCADERO TECHNOLOGIES, INC.,

Opposer,

v.

RSTUDIO, INC.

Applicant.

Opposition No. 91193335

Applications S.N.

77/691980

77/691984

77/697987

**APPLICANT'S NOTICE OF FILING OF TRIAL TESTIMONY**

Please take notice that pursuant to Trademark Rule 2.125(c), Applicant RStudio, Inc. is hereby filing electronically via the ESTTA system the deposition transcript of the trial testimony of Applicant's witness Joseph Allaire, taken on April 15, 2011 together with all associated exhibits.

Respectfully submitted,

RSTUDIO, INC.,

/Anthony E. Rufo/

---

Julia Huston  
Charles E. Weinstein  
Joshua S. Jarvis  
Anthony E. Rufo  
Foley Hoag LLP  
155 Seaport Boulevard  
Boston, MA 02210  
Tel. 617/832-1000  
[jhuston@foleyhoag.com](mailto:jhuston@foleyhoag.com)  
[cweinstein@foleyhoag.com](mailto:cweinstein@foleyhoag.com)  
[jjarvis@foleyhoag.com](mailto:jjarvis@foleyhoag.com)  
[arufo@foleyhoag.com](mailto:arufo@foleyhoag.com)

Dated: June 24, 2011

Attorneys for Applicant

CERTIFICATE OF SERVICE

I hereby certify that I have this day served a true copy of the above-identified Notice of Filing of Trial Testimony upon Opposer's attorneys of record:

Martin R. Greenstein  
Mariela P. Vidolova  
TechMark A Law Corporation  
4820 Harwood Road, 2<sup>nd</sup> Floor  
San Jose, CA 95124-5273

via First-Class Mail and e-mail to [MRG@TechMark.com](mailto:MRG@TechMark.com) and [MPV@TechMark.com](mailto:MPV@TechMark.com).

/Anthony E. Rufo/  
Anthony E. Rufo

# **Exhibit 8**

## **Part II**

**Experts****Oracle**

Volunteer

Answers to thousands of questions

27

[More Oracle Answers](#)[Question Library](#)[Ask a question about](#)[Oracle](#)[Volunteer](#)[Experts of the Month](#)[Expert Login](#)[Awards](#)[About Us](#)[Tell friends](#)[Link to Us](#)[Disclaimer](#)**About Amal****Expertise**

Can answer SQL queries related questions, exporting, importing, DML commands, net easy configuration, Oracle enterprise manager, client side oracle installation, connecting client to server

**Experience**

Can answer SQL queries related questions, exporting, importing, DML commands, net easy configuration, Oracle enterprise manager, client side oracle installation, connecting client to server. But I am a stranger to Web related issues like XML, JAVA.  
3 years

You are here: [Experts](#) > [Computing/Technology](#) > [Oracle](#) > [Oracle](#) > [Entity Relationship Diagrammer](#)

**Oracle - Entity Relationship Diagrammer**Expert: [Amal](#) - 3/2/2004**Question**

Hello Amal,

I would like to know why it is so important to create first an Entity Relationship Diagram before you start creating tables. Why can't you just create tables at once?

What is the exact function and the use of the Entity Relationship Diagrammer?

Thank you very much for your help!

Do you perhaps also know some sites where I can find objective information about Oracle Designer in stead of all those sales talk?

Kind regards, An

**Answer**

Hi An,

The ER diagram helps in the following ways,  
You know what fields are contained in which table  
You get to know which table can be related to which table to get multitable information.

To help you avoid having duplication information in multiple tables.

To help you identify which column should be the primary key

Regarding the info about web sites, I am not aware. google may help you out.

Regards,

Amal

Advertisement

Chrome fast.

[Add to this Answer](#) [Ask a Question](#)

RS828

## Related Articles

- [View the Relationship Diagram](#)
- [Creating Relationships \(cont&#39;d\)](#)
- [Creating a Simple Query in Microsoft Access](#)
- [Table Tennis Forums - Discussion Forums for Table Tennis/Ping-Pong](#)
- [Before You Buy Marriage Books](#)

[User Agreement](#) | [Privacy Policy](#) | [Kids' Privacy Policy](#) | [Help](#)

Copyright © 2008 About, Inc. AllExperts, AllExperts.com, and About.com are registered trademarks of About, Inc. All rights reserved.

PETER PIN-SHAN CHEN\*

*Graduate School of Management, University of California, Los Angeles, California 90024*

---

## ABSTRACT

In many information system projects, information requirements are initially documented in English, and then database designers convert these English descriptions into database schemas in terms of entity-relationship (ER) diagrams (or other similar representations). This paper studies the correspondence between English sentence structure and ER diagrams, and proposes eleven rules for translation. The basic constructs of English, such as noun, verb, adjective, adverb, gerund, and clause, are found to have counterparts in the ER diagrammatic technique. Finally, an example is used to demonstrate the applicability of these rules in database design.

---

## 1. INTRODUCTION

The entity-relationship (ER) diagrammatic technique [4] is a graphic way of displaying entity types, relationship types, and attributes. Many people have found the technique useful for modeling user information requirements. One of the reasons often cited is that the ER diagram is easy to understand not only for systems analysts and database designers but also for managers and users. Therefore, the ER diagram can serve as a good communication tool between the systems people and the users during the process of identifying user information requirements.

In order to construct a database using the ER diagram, the database designer not only has to interview users but also must study the documentation of the old system (if there is one) and the functional specifications of the new system. Since most of this documentation or specification material is in English (or other natural languages), it is difficult to decipher the contents of these documents into database schemas as defined by ER diagrams. There is a critical need for devising rules or guidelines for converting English descriptions into ER diagrams. This motivates our research into the correspondence between English sentence structure and entity-relationship diagrams.

---

\*Author's current address: Department of Computer Science, Louisiana State University, Baton Rouge, LA 70803.

The ER diagram was formally proposed in [4]. Figure 1(a) is an example of a simple ER diagram. The rectangular-shaped boxes represent entity types, and the diamond-shaped boxes represent relationship types. For example, in Figure 1(a), "EMP" (EMPLOYEE) and "PROJ" (PROJECT) are entity types, and "WORKS-FOR" is a relationship type. The "M" and "N" in the diagram indicate that the relationship "WORKS-FOR" is many-to-many. In other words, an employee may work for several projects, and a project may have several employees. For those relationships which are one-to-one or one-to-many, we will indicate that property accordingly in the ER diagrams. Figure 1(a) displays only entity and relationship types.

In certain situations, we need to display the properties of entities and relationships in terms of attributes and value types. Figure 1(b) is an example of an ER diagram with attributes and value types. The value types are represented by circles, and the attributes are represented by the lines connecting the entity and relationship types to the circles. In Figure 1(b), EMP#, EMP-NAME, and AGE are attributes of EMPLOYEE entities. NUMBER, NAME, and NUMBER-OF-YEARS are the corresponding value types for these attributes. Relationships may have

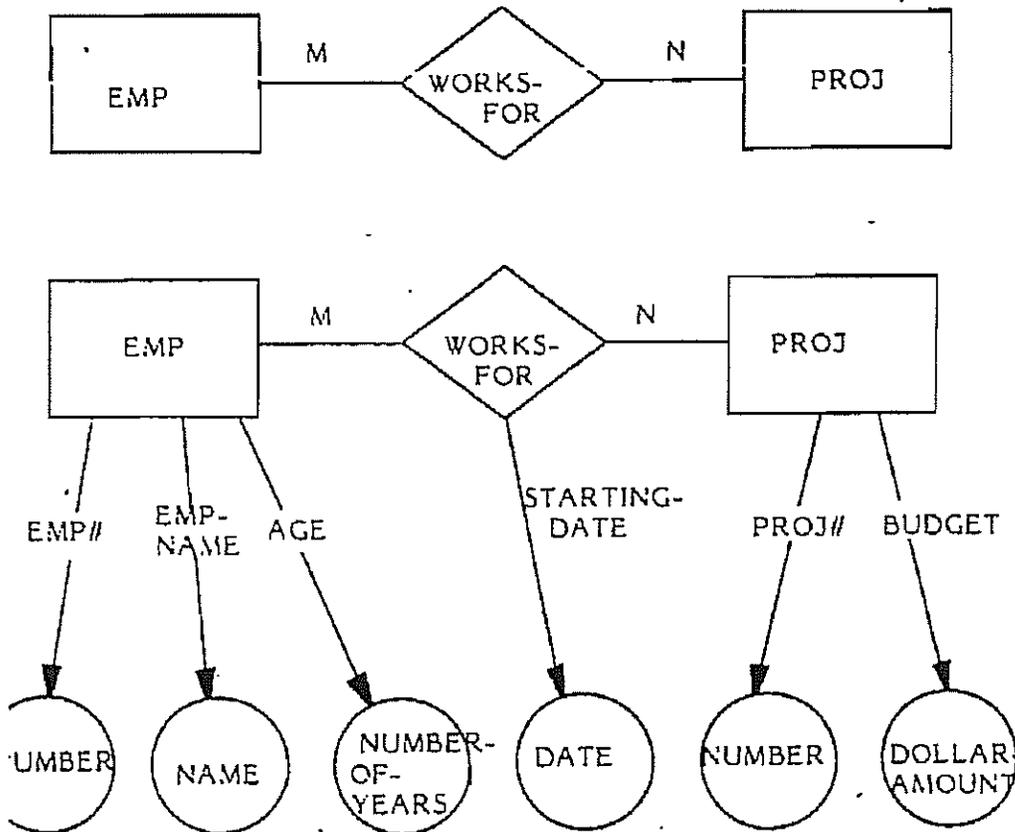


Fig. 1. An entity-relationship (ER) diagram: (a) without attributes and value types, (b) with attributes and value types.

attributes, too. For example, STARTING-DATE is an attribute of a "WORKS-FOR" relationship, since it describes the starting date of a particular employee on a particular project.

The above is a short introduction to ER diagrams. The reader may refer to [5, 7-9, 11-13, 17] for more detailed discussions of the ER diagrammatic technique and its applications.

There are many variations of ER diagrams proposed by different people or used by different organizations. Figure 2 illustrates several versions of ER diagrams. The last version in Figure 2 is the version being considered as a possible standard by the International Standards Organization (ISO). A detailed discussion of different forms of ER diagrams and models can be found in [6]. For the purpose of this paper, we will use the version illustrated in Figure 1(a) and (b). Readers may extend the ideas discussed in this paper to the version of the ER diagram they prefer to use.

After the formal introduction in 1976, there have been many proposals to extend the ER diagrammatic technique. For example, Lee and Gerritsen [10] propose a technique for modeling the fact that an entity type is a subset of

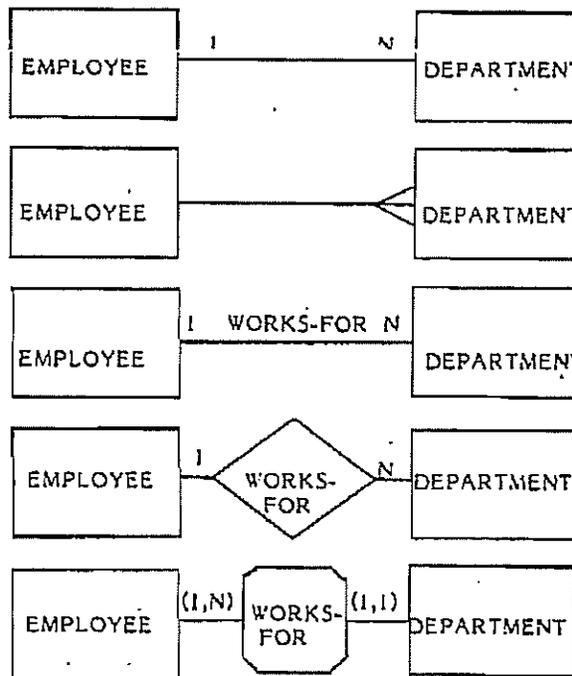


Fig. 2. Different forms of entity-relationship (ER) diagrams.

another entity type and that an entity type can be decomposed into several entity types by the range of values of a particular attribute. Schiffner and Scheuermann [15] introduce the abstracting capabilities of the ER diagram: it allows the abstraction of a group of low-level interlinked entity and relationship types into a high-level entity type. Santos, Neuhold, and Furtado [14] propose to use algebraic operators (such as the Cartesian product) to form new entity types. Batini [1] proposes a set of rules for the decomposition of ER diagrams: starting with only one entity type in the diagram, a comprehensive ER diagram can be derived by systematically applying the decomposition rules. Webre [18] proposes many different variations of relationship types. There are many other extensions which we do not have room to discuss here. The purpose of this paper is to consolidate their proposals, to develop further the ER diagrammatic technique, and, more importantly, to relate the ER diagrammatic technique to English sentence structure.

The rest of the paper is divided into three sections. Section 2 describes the rules for translating English statements into ER diagrams. Section 3 describes an example of using these rules in translating an English description of information requirements into an ER diagram. Section 4 is the conclusion.

## 2. TRANSLATION RULES

What we intend to do here is to classify certain patterns of English usage and to identify the counterpart components in ER diagrams. In this section, we will present eleven rules for translating English sentences into ER diagrams. Although we call them "rules," they might better be viewed as "guidelines," since it is possible to find counterexamples to them. The following are the detailed explanations of the translation rules:

**RULE 1.** A *common noun* (such as "person," "chair") in English corresponds to an *entity type* in an ER diagram.

**RULE 2.** A *transitive verb* in English corresponds to a *relationship type* in an ER diagram.

### EXAMPLE A.

*English statement:* A person may own a car and may belong to a political party.

*Analysis:* Note that "person," "car," and "political party" are nouns and therefore correspond to entity types. Note also that "own" and "belong to" are transitive verbs (or verb phrases) and therefore correspond to relationship types.

*ER diagram:* The corresponding ER diagram is shown in Figure 3.

**RULE 3.** An *adjective* in English corresponds to an *attribute of an entity* in an ER diagram.

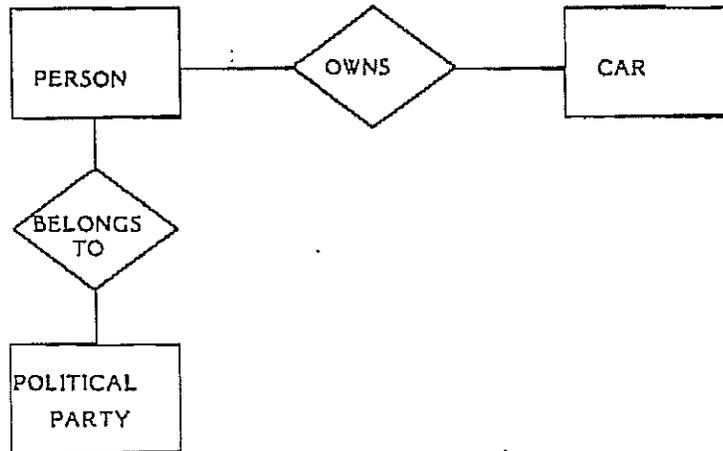


Fig. 3. An ER diagram for Example A.

**RULE 4.** An *adverb* in English corresponds to an *attribute of a relationship* in an ER diagram.

**EXAMPLE B.**

*English statement:* A 40-year-old person works on a project with project number 2175 for 20% of his time.

*Analysis:* "Person" and "project" are nouns and can be considered as entity types. Since "40-year-old" is an "adjective" modifying the noun "person", we can consider "number of years old" (or "age") as an attribute of person entities. Similarly, since "with Project number 2175" is an adjective phrase modifying the noun "project," we can view "project number" as an attribute of "project" entities. "Works on" is a transitive verb phrase and therefore corresponds to a relationship type. Since "for 20% of his time" is an adverb phrase used to modify the verb phrase "works on," we can consider "percentage of time" as an attribute of "works on" relationships.

*ER diagram:* The corresponding ER diagram is shown in Figure 4.

**RULE 5.** If the sentence has the form: "There are ... *X* in *Y*," we can convert it into the equivalent form "*Y* has ... *X*."

**EXPLANATION.** Since we are interested in the semantics of each sentence, we may use an equivalent form of the sentence in order to derive the corresponding ER diagram. In its deep semantics the sentence "there are ... *X* in *Y*" is equivalent to the sentence "*Y* has ... *X*," which can be easily translated into an ER diagram.

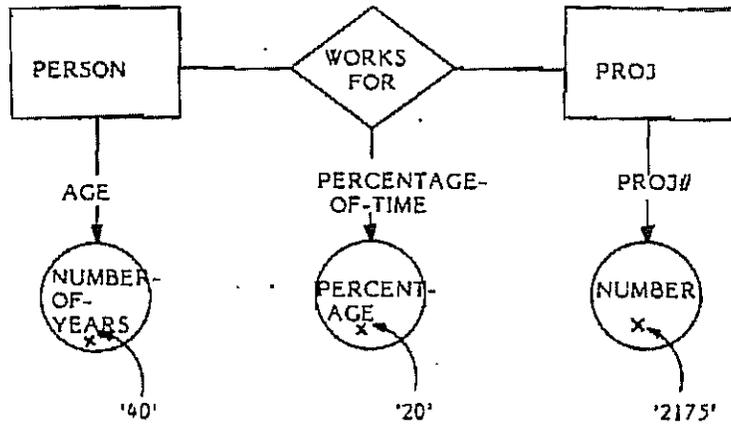


Fig. 4. An ER diagram for Example B.

EXAMPLE C.

*English statement:* There are 200 employees in this department.

*Analysis:* The equivalent form of the sentence is: "The department has 200 employees."

*ER diagram:* The corresponding ER diagram is shown in Figure 5.

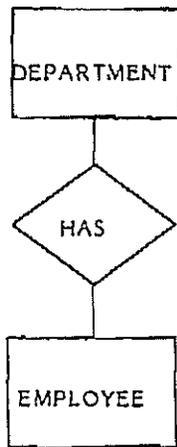


Fig. 5. An ER diagram for Example C.

RULE 6. If the English sentence has the form "The *X* of *Y* is *Z*" and if *Z* is a proper noun, we may treat *X* as a relationship between *Y* and *Z*. In this case, both *Y* and *Z* represent entities.

RULE 7. If the English sentence has the form "The *X* of *Y* is *Z*" and if *Z* is not a proper noun, we may treat *X* as an attribute of *Y*. In this case, *Y* represents an entity (or a group of entities), and *Z* represents a value.

EXPLANATION. When the sentence pattern is "The *X* of *Y* is *Z*," we may say *Y* represent an entity (or a group of entities). However, we do not know whether *Z* represent an entity or not, and neither do we know whether *X* represent a relationship or not. What we can say is that there is some kind of association between *Y* and *Z*, but we cannot tell whether this association is one of relationship or attribute. It seems that whether *X* is an attribute or relationship depends primarily on what *Z* represents. If *Z* is a proper noun (such as "John Kennedy," "The United Kingdom"), then *Z* implicitly refers to an entity although *Z* itself is considered as a value of a certain value type. For example, "John Kennedy" could refer to a "person" entity, although "John Kennedy" itself is an instance of the value type "name". If *Z* is a proper noun, we may treat *X* as a relationship between *Y* and *Z*. If *Z* is not a proper noun, we can say that *Z* is an instance of a pure value type, that is, it does not represent an entity. Therefore, we may treat *X* as an attribute of *Y*.

EXAMPLE D.

English statement: The color of the desk is blue.

Analysis: Since "blue" is not a proper noun, we may infer that "color" is an "attribute" of "desk" entities.

ER diagram: The corresponding ER diagram is shown in Figure 6.

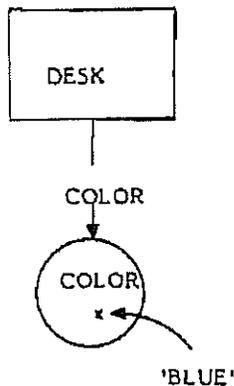


Fig. 6. An ER diagram for Example D.

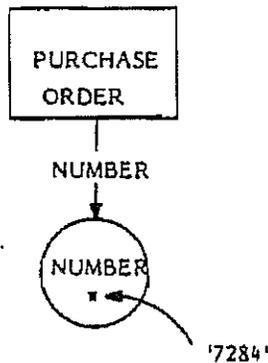


Fig. 7. An ER diagram for Example E.

#### EXAMPLE E.

*English statement:* The number of the purchase order is 7284.

*Analysis:* Since "7284" is a numeric and is not a proper noun, we may infer that "number" is an attribute of "purchase-order" entities.

*ER diagram:* The corresponding ER diagram is shown in Figure 7

#### EXAMPLE F.

*English statement:* The father of James Smith is Robert Smith.

*Analysis:* Since both "James Smith" and "Robert Smith" are proper nouns, we can say that both of them refer to entities and that "father" is a *relationship* between these two entities. If we assume that both "James Smith" and "Robert Smith" refer to entities of the "person" entity types, we may say that "father" is a relationship between "person" entities.

*ER diagram:* The corresponding ER diagram is shown in Figure 8.

**RULE 8.** The *objects* of algebraic or numeric operations can be considered as *attributes*.

#### EXAMPLE G.

*English statement:* The average salary is \$20,000, and the maximum credit limit is \$500.

*Analysis:* Since both "average" and "maximum" are algebraic operations, we may infer that "salary" and "credit limit" are attributes (of implicit employee entities). Actually, this rule can be derived from Rule 7 if we add the missing components to the original sentence. For example, the sentence "the average salary is \$20,000" can be changed into its equivalent "the average salary of employees is \$20,000". Using Rule 7, we can derive that "salary" is an attribute of "employee" entities.

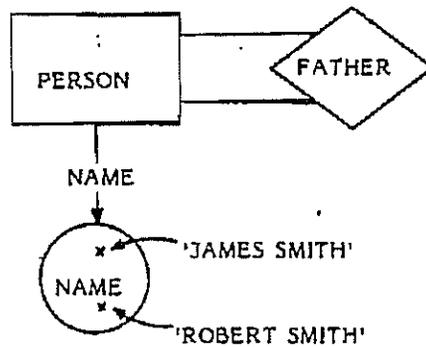


Fig. 8. An ER diagram for Example F.

**RULE 9.** A *gerund* in English corresponds to a *relationship-converted entity type* in ER diagrams.

**EXPLANATION.** In Rule 2, a transitive verb corresponds to a relationship type. In Rule #1, a noun corresponds to an entity type. Since a gerund is a noun converted from a verb, we may say that it corresponds to an entity type converted from a relationship type.

**EXAMPLE H.**

*English statement:* Products are shipped to customers, and the shipping is performed by clerks.

*Analysis:* Both "product" and "customer" are entity types, and "shipped to" is a relationship type between them. The verb "ship" is then converted to a gerund "shipping" in order to become the subject of the second clause. In other words, the relationship type "shipped to" has been converted into the entity type "shipping." The relationship type "performed by" is defined on the entity type "shipping" and the entity type "clerk."

*ER diagrams:* Figure 9 is the ER diagram representing the first clause: "Products are shipped to customers." Figure 10 is the ER diagram representing the entire sentence. Note that we use a special symbol, a rectangular-shaped box



Fig. 9. An ER diagram for Example H (part 1).

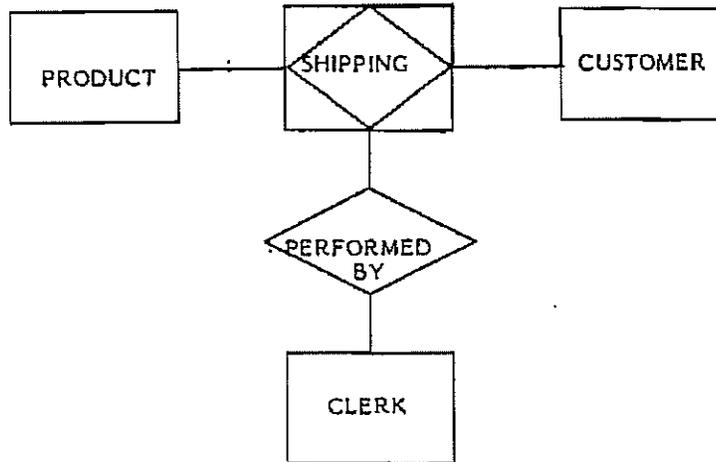


Fig. 10. An ER diagram for Example H (part 2).

on top of a diamond-shaped box, to represent a relationship-converted entity type.

**RULE 10.** A *clause* in English is a high-level entity type abstracted from a group of interconnected low-level entity and relationship types in ER diagrams.

**EXPLANATION.** The clause is a major building block in English. A clause can be used to build another clause. On the other hand, a clause may be decomposed further into subclauses.

#### EXAMPLE I.

**English statement:** The managers decide which machine is assigned to which employee.

**Analysis:** "Which machine is assigned to which employee" is a noun clause used as the object of the verb "decide." Inside this clause, "machine" and "employee" are entity types, and "assigned to" is a relationship type defined between "employee" and "machine." We could view the entire clause as an equivalent to a high-level entity called *assignment*.

**ER diagram:** Figure 11 is the corresponding ER diagram. Note that the clause is represented by a high-level rectangular-shaped box, which encloses a group of interconnected low-level entity and relationship types.

**RULE 11.** A *sentence* in English corresponds to one or more entity types connected by a relationship type, in which each entity type can be decomposed (recursively) into low-level entity types interconnected by relationship types.

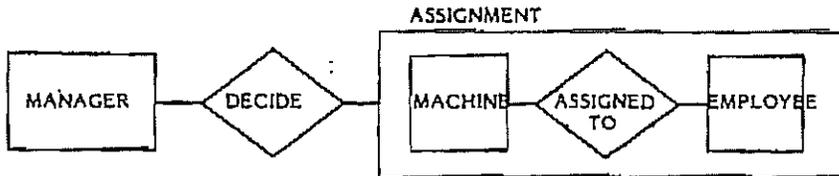


Fig. 11. An ER diagram for Example I.

**EXPLANATION.** Each sentence has one or more nouns, which correspond to entity types. In addition, each sentence has one verb, which corresponds to a relationship type. Since a sentence may be decomposed into clauses, which in turn may be decomposed into subclauses, the corresponding entity types may be decomposed (recursively) into low-level entity types interconnected by a relationship type.

Certainly, these are the basic rules. We expect that more rules will be developed in the future.

### 3. AN EXAMPLE

In this section, we will describe an example of translating a short English description of information requirements into an ER diagram. The example, which was a revision of a case study originally proposed in [3], is taken from Teorey and Fry's paper [16]. It is a description of the information requirements of an information system for labor and employee management in a manufacturing firm. Teorey and Fry [16] converted this description into an ER diagram, but they did not explain in detail how the ER diagram was derived. In this section we apply the rules described in the previous section to the translation of the English description into an ER diagram, and then compare our result with Teorey and Fry's.

#### 3.1. ENGLISH DESCRIPTION OF INFORMATION REQUIREMENTS

As mentioned, a manufacturing firm wishes to develop a computer-based information system for labor and employee management. The following is the English description of the information requirements of the proposed information system:

The company has 50 plants located in 40 states and approximately 100,000 employees. Each plant is divided into departments and further subdivided into work stations. There are 100 departments and 500 work stations in the company. In each department there is an on-line time clock at which employees report their arrival and departure. A work task is associated with one

of 20 different job types. Each of the job types can be performed at each of the plants. During a given day an employee may perform more than one work task, each associated with a different job type, and each can be performed at a different work station. Each work station has an on-line data entry device at which an employee reports activity on a work task. There are five worker unions represented in the company, and every employee belongs to exactly one union. Although the size of the company remains stable, about 20 percent of the employees leave each year and are replaced by new personnel. ([16, p. 191], with minor modifications.)

### 3.2. TRANSLATION

Figure 12 is the ER diagram derived by Teorey and Fry from the above English description, but there are several unanswered questions. For example, how can anyone (without previous experience) derive Figure 12 from the English description given? Is Figure 12 the only representation of this English description? In the following, we are going to translate the English description into an ER diagram using the rules described in the previous section. We will then compare our result with Teorey's result. Although we do not have clear-cut answers to the questions raised above, we do believe that the following exercise will stimulate more research into this area so that a more rigorous methodology will be developed in the future.

In the following, we will analyze the English description one sentence at a time.

**STATEMENT 1.** The company has 50 plants located in 40 states and approximately 100,000 employees.

**ANALYSIS AND TRANSLATION.** This sentence can be decomposed into three sentences (clauses): (1) the company has 50 plants; (2) the 50 plants are located in 40 states; (3) the company has approximately 100,000 employees. Applying Rules 1 and 2, we get the ER diagram in Figure 13. Although the verb "has" is used in the original sentence, we use "HAS-1" and "HAS-2" in Figure 13 in order to make each relationship type name unique.

**STATEMENT 2.** Each plant is divided into departments and further subdivided into work stations.

**ANALYSIS AND TRANSLATION.** Applying Rules 1 and 2, we get the ER diagram in Figure 14.

**STATEMENT 3.** There are 100 departments and 500 work stations in the company.

**ANALYSIS AND TRANSLATION.** Applying Rule 5, we have the equivalent sentence: "The company has 100 departments and 500 work stations." However, this sentence has no new information on entity types or relationship types, and it contains only supplemental information to the previous two statements.

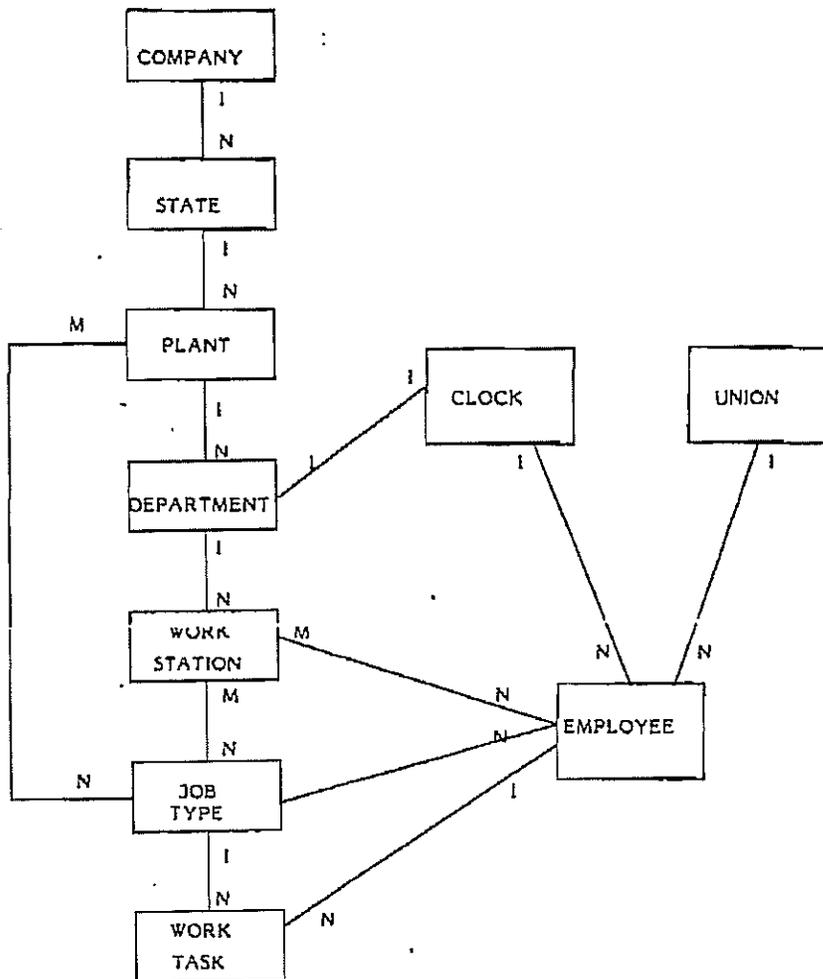


Fig. 12. The ER diagram taken from Teorey and Fry's paper [16, p. 193], with minor modifications.

For example, we now know the average value of  $M$  in Figure 14 is 2 (100 departments divided by 50 plants), and the average value of  $N$  in Figure 14 is 5 (500 work stations divided by 100 departments). This information is useful in the design of physical data structures but is not essential for the conceptualization of ER diagrams.

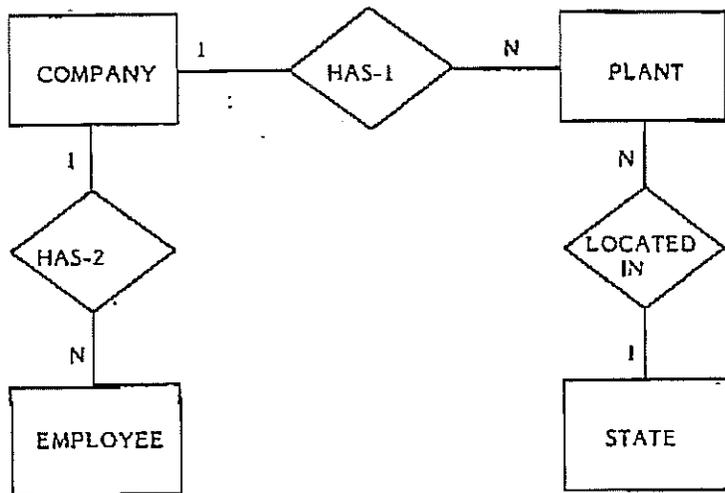


Fig. 13. An ER diagram for Statement 1.

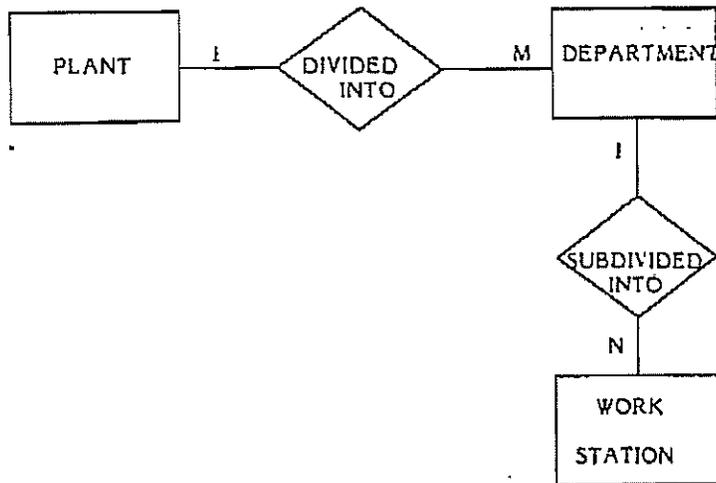


Fig. 14. An ER diagram for Statement 2.

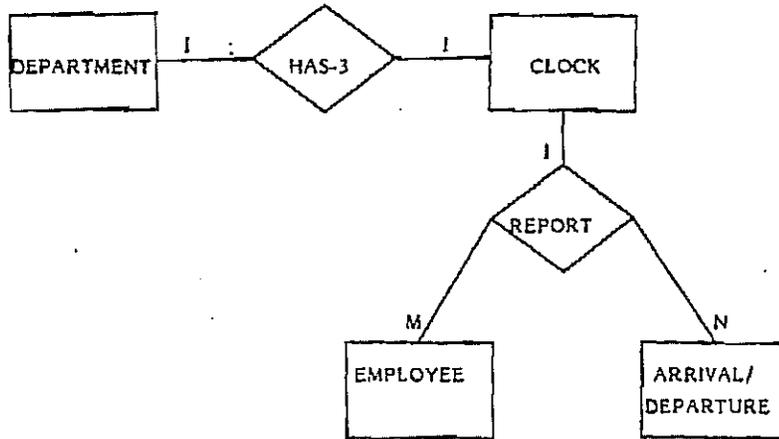


Fig. 15. An ER diagram for Statement 4.

STATEMENT 4. In each department there is an on-line time clock at which employees report their arrival and departure.

ANALYSIS AND TRANSLATION. Strictly speaking, this statement is concerned with how certain data are collected, and therefore is irrelevant to the construction of the ER diagram representing the database. However, for practice we will analyze this statement and derive the corresponding ER diagram.

This statement indicates that the relationship between "department" entities and "clock" entities is one-to-one. In addition, "employee," "clock," and "arrival/departure" are entities involved in a "report" relationship. The corresponding ER diagram is shown in Figure 15.

STATEMENT 5. A work task is associated with one of 20 different job types.

ANALYSIS AND TRANSLATION. This indicates that the relationship between "job type" entities and "work task" entities is one-to-many. The corresponding ER diagram is shown in Figure 16.



Fig. 16. An ER diagram for Statement 5.



Fig. 17. An ER diagram for Statement 6.

STATEMENT 6. Each of these job types can be performed at each of the plants.

ANALYSIS AND TRANSLATION. This indicates that the relationship between "job type" entities and "plant" entities is many-to-many. The corresponding ER diagram is shown in Figure 17.

STATEMENT 7. During a given day an employee may perform more than one work task, each associated with a different job type, and each can be performed at a different work station.

ANALYSIS AND TRANSLATION. This indicates that the relationship between "employee" entities and "work task" entities is one-to-many. Also, the relationship between "work task" entities and "job type" entities is many-to-one. It also implies that the relationship between "employee" entities and "work station" entities is many-to-many. The corresponding ER diagram is shown in Figure 18.

STATEMENT 8. Each work station has an on-line data entry device at which an employee reports activity on a work task.

ANALYSIS AND TRANSLATION. Similarly to Statement 4, this statement is concerned with data collection and is irrelevant to the construction of the ER diagram representing the database. For the benefit of those interested in the correspondence between ER diagrams and English statements, we perform the following analysis.

The relationship between "work station" entities and "(on-line) data entry device" entities is one-to-one. In addition, "employee," "work task activities," and "data entry device" are involved in a "report-2" relationship (The name "report-2" is used to distinguish it from the "report" relationship in Figure 15). The corresponding ER diagram is shown in Figure 19.

STATEMENT 9. There are five worker unions represented in the company, and every employee belongs to exactly one union.

ANALYSIS AND TRANSLATION. The relationship between "company" entities and "(worker) union" entities is one-to-many. In addition, the relationship between "employee" entities and "union" entities is many-to-one. The corresponding ER diagram is shown in Figure 20.

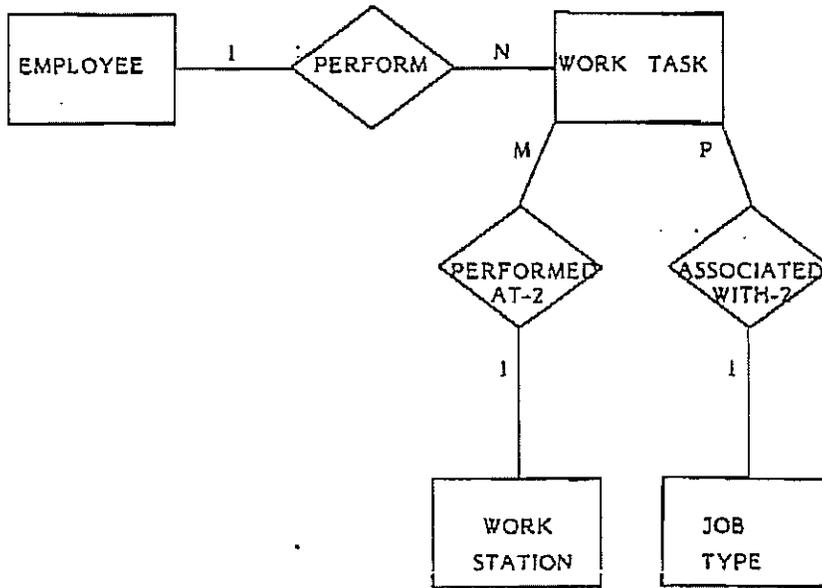


Fig. 18. An ER diagram for Statement 7.

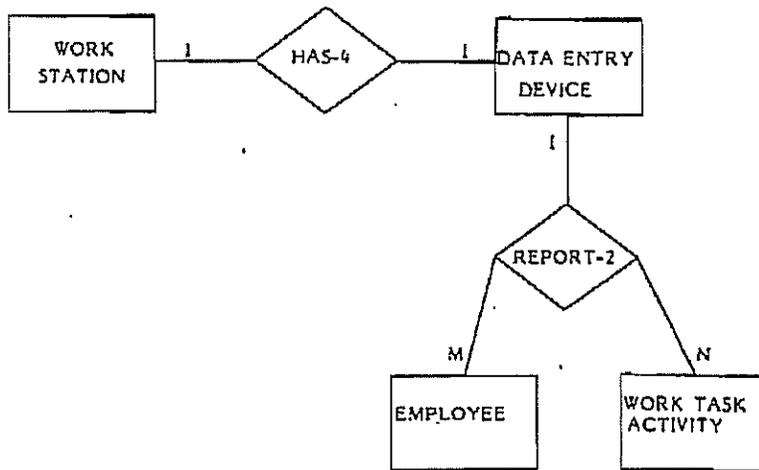


Fig. 19. An ER diagram for Statement 8.

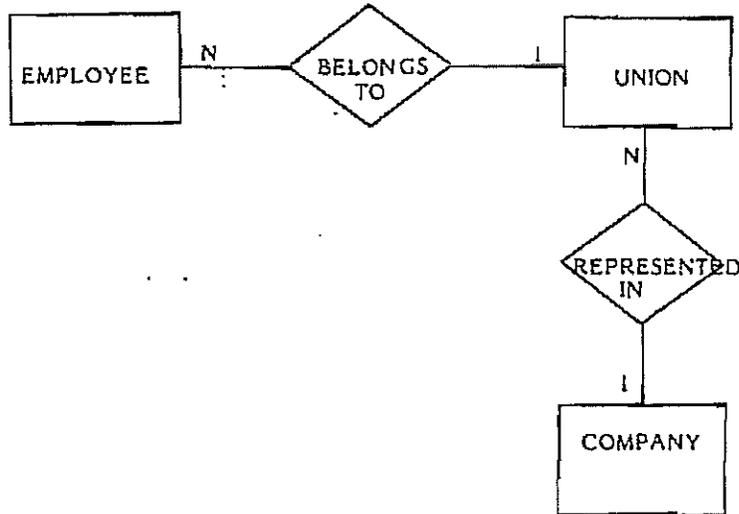


Fig. 20. An ER diagram for Statement 9.

**STATEMENT 10.** Although the size of the company remains stable, about 20 percent of the employees leave each year and are replaced by new personnel.

**ANALYSIS AND TRANSLATION.** The relationship between "employee" entities and "company" entities has been shown in Figure 13. This statement gives only the frequency of changing (updating) the actual relationship instances between "company" entities and "employee" entities; it does not provide information on new entity or relationship types.

In the preceding, we have analyzed each statement and derived a corresponding ER diagram. These individual diagrams can be merged to form an overall ER diagram. Figure 21 is such a diagram, formed by merging Figures 13 to 20 (without Figures 15 and 19). Since there is no attribute for any relationship in Figure 21, we may use a simple convention in drawing ER diagrams (using straight lines instead of diamond-shaped boxes to represent relationship types and no explicit names for relationship types). Figure 22 is the resulting version of the diagram. Also, since we are modeling the information requirements for a specific company only, we can delete the "company" entity type, and then Figure 23 becomes the final ER diagram.

### 3.3. COMPARISON OF RESULTS

What are the differences between the ER diagram derived from the translation rules and the ER diagram derived by Teorey and Fry? Comparing Figure

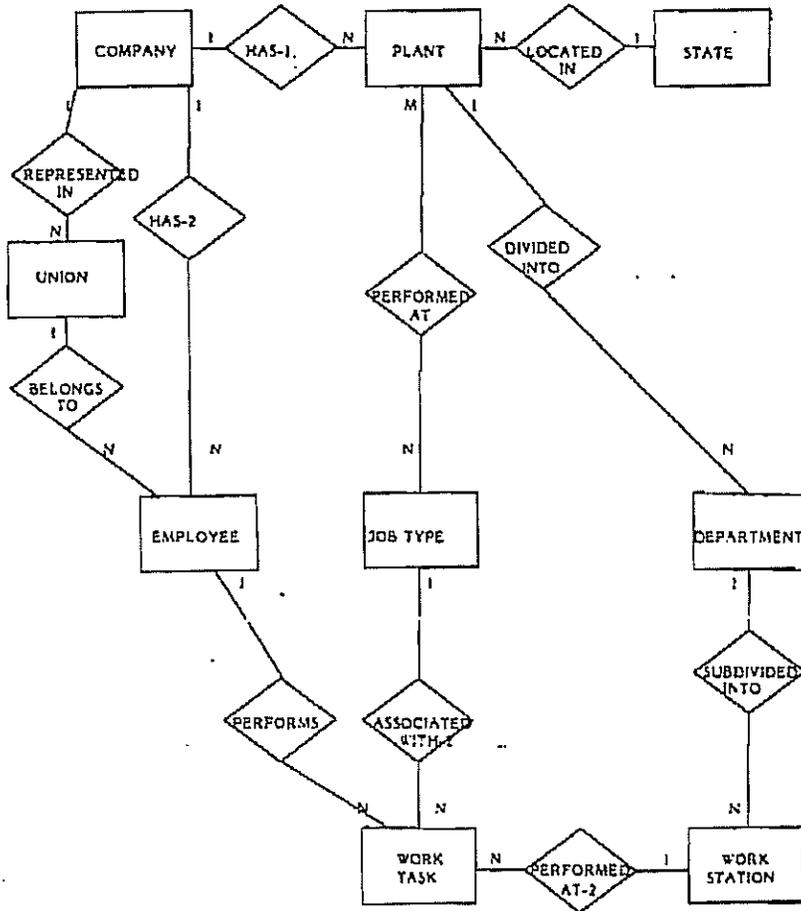


Fig. 21. An overall ER diagram.

23 with Figure 12, we can see that in Figure 12, there are two more entity types (COMPANY and CLOCK)<sup>1</sup> and three more relationship types (EMPLOYEE-WORK-STATION, EMPLOYEE-JOB-TYPE, and WORK-STATION-JOB-TYPE). In addition, the relationship type WORK-STATION-WORK-TASK is missing in Figure 23. Are these differences significant or superficial? Which diagram is more faithful to the original English description? In the following, we will attempt to address these questions.

<sup>1</sup>In the figures, we use capital letters (instead of quotation marks) to denote entity or relationship type names.

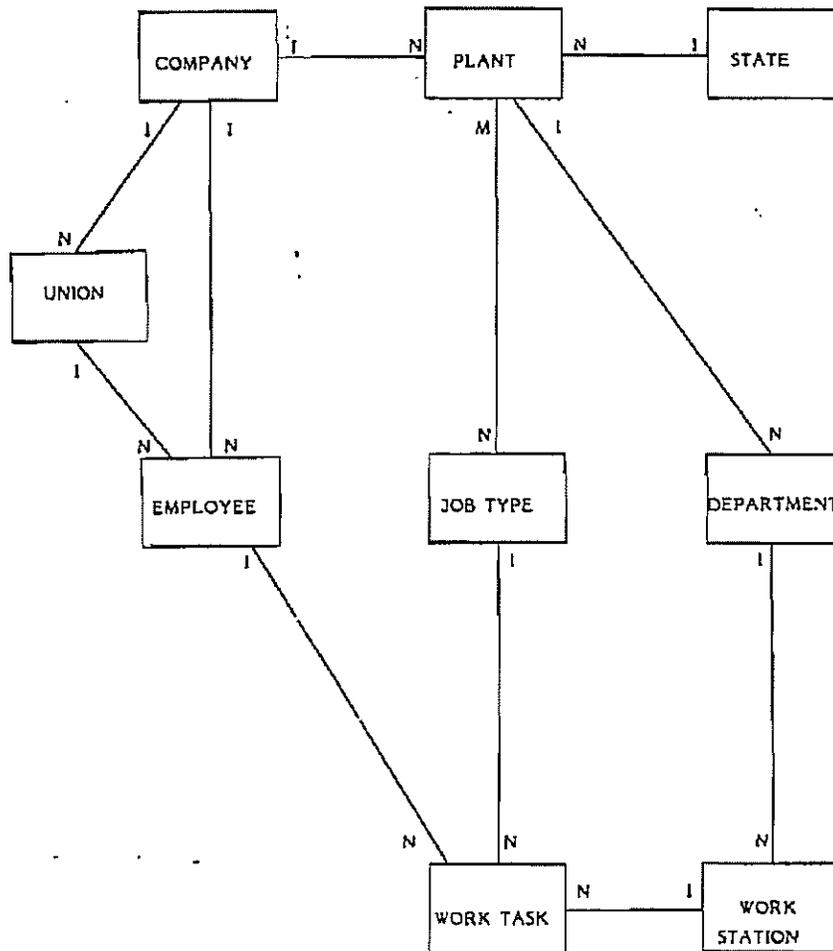


Fig. 22. A revised ER diagram corresponding to Figure 21.

Let us first examine the extra entity types. In Figure 23, we do not have the *COMPANY* entity type, since we said that we were modeling a specific company. If we were to accept *COMPANY* as an entity type, we probably should have two additional relationship types as shown in Figure 22: *COMPANY-EMPLOYEE* and *COMPANY-PLANT*. In addition, the relationship type *COMPANY-STATE* in Figure 12 is not directly derivable from the original English description. The *CLOCK* entity type in Figure 12 is not included in Figure 23, since we said that it was

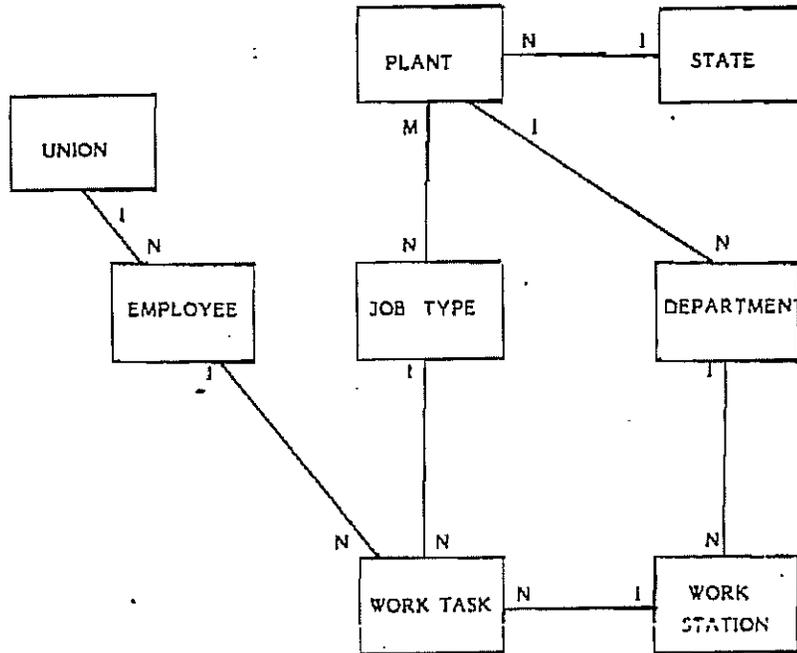


Fig. 23. The final ER diagram.

related to the data collection activity and was not relevant to the construction of the database itself. If we were to include the **CLOCK** entity type, we should also include the **DATA-ENTRY-DEVICE** (see Figure 19) for the sake of consistency.

Now, let us examine the extra relationship types. We think that these extra relationship types may be derivable from the existing relationship types in Figure 23. For example, the **WORK-STATION-JOB-TYPE** relationship in Figure 12 may be inferred by combining the two relationship types in Figure 23: **JOB-TYPE-WORK-TASK** and **WORK-TASK-WORK-STATION**. The extra relationship **EMPLOYEE-JOB-TYPE** may be derived by combining the two relationship types **EMPLOYEE-WORK-TASK** and **WORK-TASK-JOB-TYPE**. Similarly, we can construct the extra relationship type **EMPLOYEE-WORK-STATION** through other relationship types. Therefore, the extra relationship types in Figure 12 do not contain significant extra information, and they are derivable from the existing relationship types in Figure 23.

Finally, let us discuss the relationship type **WORK-TASK-WORK-STATION**, which is missing in Figure 12. It seems that this relationship type cannot be derived completely from other relationship types, since we cannot decide

whether this relationship type is one-to-one, one-to-many, or many-to-many. We think that a reason should be given why this relationship is deleted in Figure 12.

In conclusion, Figure 23 and Figure 12 are very similar. However, Figure 23 is a more faithful representation of the original English description of the information requirements. It seems that additional information, which is based on the database designer's own knowledge of the system, was incorporated into the construction of the ER diagram in Figure 12. However, this additional information was not documented, and it is difficult for any person to derive Figure 12 based solely on the original English descriptions. It is not our intention to claim that Figure 23 is better than Figure 12. What we try to do here is to demonstrate that by systematically applying the translation rules proposed in Section 2, a more faithful representation can be obtained. We believe that by following the translation rules we can rely less on the database designers' intuition and more on structured methods. We hope that these translation rules can stimulate more research on a rigorous and comprehensive logical database design methodology.

#### 4. CONCLUSION

In this paper, we have proposed eleven basic rules for translation between English sentences and ER diagrams. These translation rules can be used in the conversion of an English language description of information requirements into ER diagrams. Using an example, we have demonstrated that an ER diagram more faithful to the original description can be derived by systematically applying the translation rules instead of relying solely on database designers' intuition.

Certainly, the rules provided in this paper are not complete and may have exceptions. However, we hope this paper will stimulate more research in this area so that a set of more complete and accurate rules may be defined in the near future and that a more rigorous methodology for information requirements analysis and logical database design can be developed.

*The author would like to thank Dennis Perry and Ilchoo Chung for their contribution of ideas for the translation rules.*

#### REFERENCES

1. C. Batini, Top-down design of entity-relationship models, in *Entity-Relationship Approach to Systems Analysis and Design* (P. Chen, Ed.), North-Holland, Amsterdam, 1980.
2. R. L. Bennetworth et al., The implementation of GERM, an entity-relationship data base

- management system, in *Proceedings of the Seventh Very Large Data Base Conference*, Cannes, France, Sept. 1981.
3. J. Bubenko, S. Bertid, E. Lindencrona-Ohlin, and S. A. Nachmens, From information requirements to DBTG data structures, in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1977.
  4. P. P. Chen, The entity-relationship model: Toward a unified view of data, *ACM Trans. Database Systems* 1, No. 1 (Mar. 1976).
  5. P. P. Chen, The entity-relationship model: A basis for the enterprise view of data, in *Proceedings of the 1977 National Computer Conference*, Dallas, June 1977.
  6. P. P. Chen, A preliminary framework for entity-relationship model, in: *Entity-Relationship Approach to Information Modeling and Analysis* (P. Chen, Ed.), ER Institute, P.O. Box 617, Saugus, CA 91350, 1981.
  7. T. C. Chiang and R. F. Bergeron, A data base management system with an E-R conceptual model," in *Entity-Relationship Approach to Systems Analysis and Design* (P. Chen, Ed.), North-Holland, Amsterdam, 1980.
  8. R. Elmasri and G. Wiederhold, GORDAS: A formal high-level query language for the entity-relationship model," in *Entity-Relationship Approach to Information Modeling and Analysis* (P. Chen, Ed.), ER Institute, P.O. Box 617, Saugus, CA 91350, 1981.
  9. E. Y. Lien, The design of entity-relationship distributed data base systems, in *Proceedings of the 3rd IEEE Computer Society Compsac Conference*, Chicago, 1978.
  10. D. Lee and R. Gerritsen, A hybrid entity-relationship model, Working Paper, Wharton School, Univ. of Pennsylvania, Philadelphia, 1977.
  11. E. L. Lusk, R. A. Overbeek, and B. Parrello, The design of IMS databases from entity-relationship model, in *Proceedings of the 1980 ACM SIGMOD International Conference on Management of Data*, Los Angeles, 1980.
  12. P. Ng, Formal definitions of the entity-relationship model, *IEEE Trans. Software Engrg.*, 1980.
  13. H. Sakai, On the optimization of the entity-relationship model, in *Proceedings of the 3rd USA-Japan Computer Conference*, AFIPS Press, N.J., 1978.
  14. C. S. dos Santos, E. J. Neuhold, and A. L. Furtado, A data type approach to the entity-relationship model, in *Entity-Relationship Approach to Systems Analysis and Design* (P. Chen, Ed.), North-Holland, Amsterdam, 1980.
  15. G. Schiffner and P. Scheuermann, Multiple view and abstractions with an extended entity-relationship model, *J. Comput. Languages* 4:139-154 (1980).
  16. T. J. Teorey and J. P. Fry, The logical record access approach to database design, *ACM Comput. Surveys* 12:2 (June 1980).
  17. J. D. Ullman, *Principles of Database Systems*, Computer Science Press, 1980.
  18. N. Webre, An extended entity-relationship model and its use on a defense project, in *Entity-Relationship Approach to Information Modeling and Analysis* (P. Chen, Ed.), ER Institute, P.O. Box 617, Saugus, CA 91350, 1981.

*Note: Entity-Relationship Approach to Information Modeling and Analysis* (P. Chen, Editor) has been republished by North-Holland Publishing Company, 1983.



some aspect of the real world which can be distinguished from other aspects of the real world.<sup>[3]</sup>

Two related entities

An entity may be a physical object such as a house or a car, an event such as a house sale or a car service, or a concept such as a customer transaction or order. Although the term entity is the one most commonly used, following Chen we should really distinguish between an entity and an entity-type. An entity-type is a category. An entity, strictly speaking, is an instance of a given entity-type. There are usually many instances of an entity-type. Because the term entity-type is somewhat cumbersome, most people tend to use the term entity as a synonym for this term.

Entities can be thought of as nouns. Examples: a computer, an employee, a song, a mathematical theorem.

A relationship captures how two or more entities are related to one another. Relationships can be thought of as verbs, linking two or more nouns. Examples: an *owns* relationship between a company and a computer, a *supervises* relationship between an employee and a department, a *performs* relationship between an artist and a song, a *proved* relationship between a mathematician and a theorem.

The model's linguistic aspect described above is utilized in the declarative database query language ERROL, which mimics natural language constructs.

Entities and relationships can both have attributes. Examples: an *employee* entity might have a *Social Security Number* (SSN) attribute; the *proved* relationship may have a *date* attribute.

Every entity (unless it is a weak entity) must have a minimal set of uniquely identifying attributes, which is called the entity's primary key.

Entity-relationship diagrams don't show single entities or single instances of relations. Rather, they show entity sets and relationship sets. Example: a particular *song* is an entity. The collection of all songs in a database is an entity set. The *eaten* relationship between a child and her lunch is a single relationship. The set of all such child-lunch relationships in a database is a relationship set. In other words, a relationship set corresponds to a relation in mathematics, while a relationship corresponds to a member of the relation.

Certain cardinality constraints on relationship sets may be indicated as well.

## Diagramming conventions

Entity sets are drawn as rectangles, relationship sets as diamonds. If an entity set participates in a relationship set, they are connected with a line.

Attributes are drawn as ovals and are connected with a line to exactly one entity or relationship set.

Cardinality constraints are expressed as follows:

- a double line indicates a *participation constraint*, totality or surjectivity: all entities in the entity set must participate in *at least one* relationship in the relationship set;
- an arrow from entity set to relationship set indicates a key constraint, i.e. injectivity: each entity of the entity set can participate in *at most one* relationship in the relationship set;
- a thick line indicates both, i.e. bijectivity: each entity in the entity set is involved in *exactly one* relationship.
- an underlined name of an attribute indicates that it is a key: two different entities or relationships with this attribute always have different values for this attribute.

Attributes are often omitted as they can clutter up a diagram; other diagram techniques often list entity attributes within the rectangles drawn for entity sets.

Chen's notation for entity-relationship modeling uses rectangles to represent entities, and diamonds to represent

relationships appropriate for first-class objects: they can have attributes and relationships of their own.

Related diagramming convention techniques:

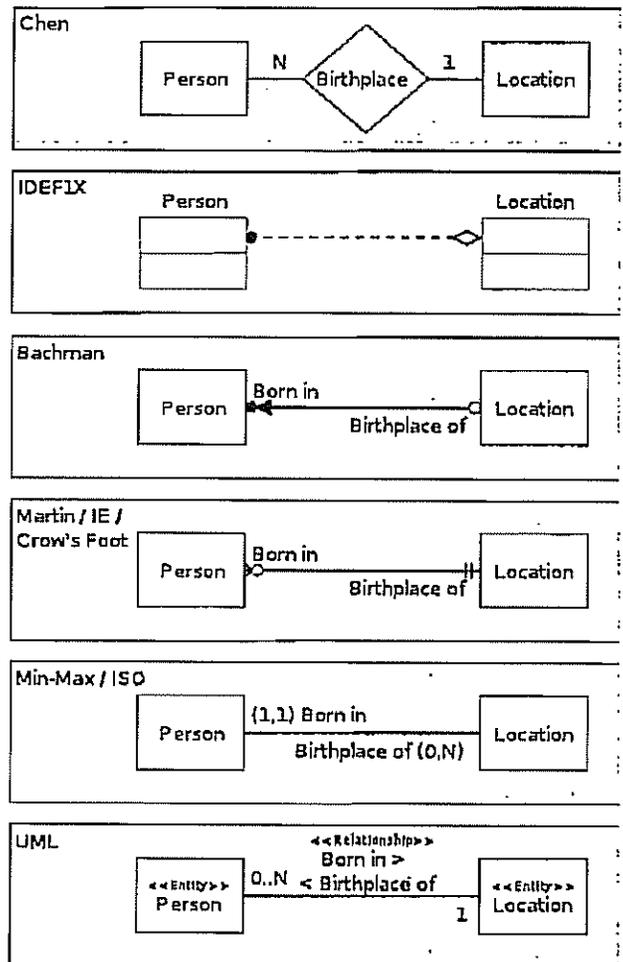
- Bachman notation
- EXPRESS
- IDEF1X<sup>[4]</sup>
- Martin notation
- (min, max)-notation of Jean-Raymond Abrial in 1974
- UML class diagrams

## Crow's Foot Notation

Crow's Foot notation is used in Barker's Notation, SSADM and Information Engineering. Crow's Foot diagrams represent entities as boxes, and relationships as lines between the boxes. The ends of these lines are shaped to represent the cardinality of the relationship.

Usage of Chen notation is more prevalent in the United States, while usage of Crow's Foot notation was used primarily in the UK. Crow's Foot notation was used in the 1980s by the consultancy practice CACI. Many of the consultants at CACI (including Barker) subsequently moved to Oracle UK, where they developed the early versions of Oracle's CASE tools, introducing the notation to a wider audience. Crow's Foot notation is used by these tools: ARIS, System Architect, Visio, PowerDesigner, Toad Data Modeler, DeZign for Databases, Devgems Data Modeler, OmniGraffle, and MySQL

Workbench. CA's ICASE tool, CA Gen aka Information\_Engineering\_Facility also uses this notation.



## ER diagramming tools

There are many ER diagramming tools. Some free software ER diagramming tools that can interpret and generate ER models, SQL and do database analysis are MySQL Workbench and DBDesigner (open-source). A freeware ER tool that can generate database and application layer code (webservices) is the RISE Editor.

Some of the proprietary ER diagramming tools are ARIS, Avolution, dbForge Studio for MySQL, DeZign for Databases, ER/Studio, Devgems Data Modeler, ERwin, MEGA International, OmniGraffle, Oracle Designer, PowerDesigner, Rational Rose, Sparx Enterprise Architect, SQLyog, System Architect, Toad Data Modeler, SQL Maestro, Microsoft Visio, Visible Analyst, and Visual Paradigm.

Some free software diagram tools just draw the shapes without having any knowledge of what they mean, nor do they generate SQL. These include Kivio and Dia. DIA diagrams, however, can be translated with tedia2sql.

## See also

- Database design
- Associative entity
- Data structure diagram
- Enhanced Entity-Relationship Model

- Object Role Modeling
- Three schema approach
- Unified Modeling Language
- Value range structure diagrams

## References

1. ^ "The Entity Relationship Model: Toward a Unified View of Data".
2. ^ A.P.G. Brown, "Modelling a Real-World System and Designing a Schema to Represent It", in Douque and Nijssen (eds.), *Data Base Description*, North-Holland, 1975, ISBN 0-7204-2833-5.
3. ^ Paul Beynon-Davies (2004). *Database Systems*. Houndmills, Basingstoke, UK: Palgrave
4. ^ IDEF1X

## Further reading

- Richard Barker (1990). *CASE Method: Tasks and Deliverables*. Wokingham, England: Addison-Wesley.
- Paul Beynon-Davies (2004). *Database Systems*. Houndmills, Basingstoke, UK: Palgrave
- Peter Chen, Peter Pin-Shan (March 1976). "The Entity-Relationship Model - Toward a Unified View of Data". *ACM Transactions on Database Systems* 1 (1): 9–36. doi:10.1145/320434.320440.

## External links

- Entity Relationship Modeling - Article from *Development Cycles*
- Entity Relationship Modelling
- An Entity Relationship Diagram Example. Demonstrates the crow's feet notation by way of an example.
- "Entity-Relationship Modeling: Historical Events, Future Trends, and Lessons Learned" by Peter Chen.
- "English, Chinese and ER diagrams" by Peter Chen.
- Case study: E-R diagram for Acme Fashion Supplies by Mark H. Ridley.
- Logical Data Structures (LDSs) - Getting started by Tony Drewry.
- Introduction to Data Modeling
- Lecture by Prof.Dr.Muhittin GÖKMEN, Department of Computer Engineering, Istanbul Technical University.
- ER-Diagram Convention
- Crow's Foot Notation
- "Articulated Entity Relationship (AER) Diagram for Complete Automation of Relational Database Normalization"P. S. Dhabe, Dr. M. S. Patwardhan, Asavari A. Deshpande.

Retrieved from "[http://en.wikipedia.org/wiki/Entity-relationship\\_model](http://en.wikipedia.org/wiki/Entity-relationship_model)"

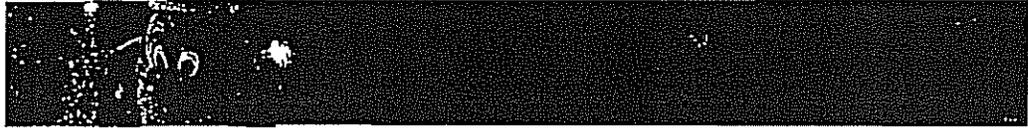
Categories: Data modeling diagrams

---

- This page was last modified on 15 September 2010 at 09:32.
  - Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. See Terms of Use for details.
- Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.



► Become a member of a new community of IT evaluators.



- CodeWalker Forums
- Tutorials
- Database Articles
- Miscellaneous
- Navigation Usability
- PEAR Articles
- Programming Basics
- Server Administration
- XML Tutorials
- Reviews
- Database Book Reviews
- Linux Book Reviews
- Miscellaneous Reviews
- PHP Book Reviews
- PHP Software Reviews
- Server Admin Reviews
- SQL Tool Reviews
- Code Gallery
- Content Management Code
- Content Code
- Counters Code
- Database Code
- Date Time Code
- Discussion Board Code
- Email Code
- File Manipulation Code
- GUI Code
- Link Form Code
- Miscellaneous Code
- Search Code
- Site Navigation Code
- User Management Code
- Mobile Linux
- Case Studies
- iPad Development

## DATABASE CODE

### Relationships, Entities and Database Design

By: [O'Reilly Media](#)  
 Rating: ★★★★★ / 13  
 2007-11-29

[Search For More Articles!](#)  
[Disclaimer](#)  
[Author Teams](#)

#### Table of Contents:

- Relationships, Entities and Database Design
- Intermediate Entities
- Entity Relationship Modeling Examples
- The University Database

Rate this Article: Poor      Best  Rate

#### ADD THIS ARTICLE TO:

- Delicious
- Digg
- Blink
- Simple
- Google
- Spurl
- MyWeb
- Furl

- Email Me Similar Content When Posted
- Add Developer Shed Article Feed To Your Site
- Email Article To Friend
- Print Version Of Article
- PDF Version Of Article



ADVERTISEMENT

### Relationships, Entities and Database Design - Entity Relationship Modeling Examples (Page 3 of 4)

Earlier in this chapter, we showed you how to design a database and understand an Entity Relationship (ER) diagram. This section explains the requirements for our three example databases—music, university, and flight—and shows you their Entity Relationship diagrams:

- The music database is designed to store details of a music collection, including the albums in the collection, the artists who made them, the tracks on the albums, and when each track was last played.
- The university database captures the details of students, courses, and grades for a university.
- The flight database stores an airline timetable of flight routes, times, and the plane types.

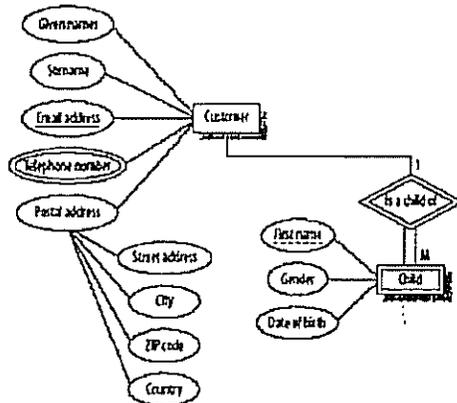


Figure 4-9. The ER diagram representation of a weak entity

### Developers: See Also...

- Code-Free Database-Driven SQL Apps in SharePoint - Free trial (Sponsor)
- Examples and Tools for Database Design (2007-12-06)
- Modeling and Designing Databases (2007-11-11)
- An Introduction to Database Normalization (2004-02-26)
- Entity Relationship Modeling (2004-10-05)
- Plan a Good Database (2010-06-02)
- MapPoint Web Service Find APIs (2008-02-03)
- The Basics (2005-03-30)



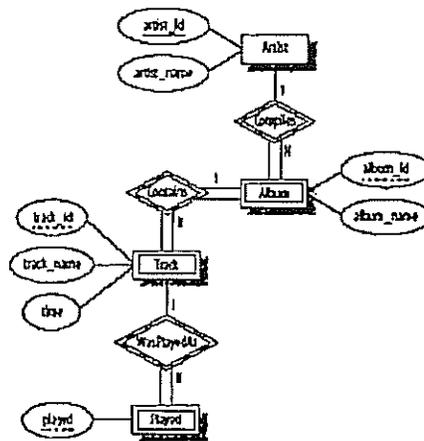


Figure 4-11. The ER diagram of the music database

**What it doesn't do**

We've kept the music database simple because adding extra features doesn't help you learn anything new, it just makes the explanations longer. If you wanted to use the music database in practice, then you might consider adding the following features:

- Support for compilations or various-artists albums, where each track may be by a different artist and may then have its own associated album-like details such as a recording date and time. Under this model, the album would be a strong entity, with many-to-many relationships between artists and albums.
- Playlists, a user-controlled collection of tracks. For example, you might create a playlist of your favorite tracks from an artist.
- Track ratings, to record your opinion on how good a track is.
- Source details, such as when you bought an album, what media it came on, how much you paid, and so on.
- Album details, such as when and where it was recorded, the producer and label, the band members or sidemen who played on the album, and even its artwork.
- Smarter track management, such as modeling that allows the same track to appear on many albums.

*Next: The University Database >>*

*More Database Code Articles  
More by O'Reilly Media*

**Comments On: Relationships, Entities and Database Design**

- This article is an excerpt from the book "Learning MySQL," published by O'Reilly. We...
- Nice examples. There are a few things I didn't understand very well though;...

>>> [Post your comment now!](#)

**Buy this book...**



This article is excerpted from chapter 4 of the book *Learning MySQL*, written by Seyed M.M. "Saled" Tahaghoghi and Hugh E. Williams (O'Reilly, 2006; ISBN: 0596008643). Check it out today at your favorite bookstore. Buy this book now.

**DATABASE CODE ARTICLES**

- Converting CSV Files to MySQL Insert Queries...
- Examples and Tools for Database Design
- Relationships, Entities and Database Design
- Modeling and Designing Databases
- Data extract to Excel
- Oracle database class 0.76
- The opposite of mysql\_fetch\_assoc
- On line Thermal Transmittance Calculation
- pjtTextBase
- PHP Object Generator
- FastMySQL
- RC4PHP
- SQL function with integrated sprintf()
- DB Interaction Classes v1.1

- deenMySQLParser

[Find More Database Code Articles](#)



Shop All Departments

Search

Books

31

Books

Advanced Search

Browse Subjects

New Releases

Bestsellers

The New York Times® Bestsellers

Libros En Español

Bargain Books

Textbooks

Entity-Relationship Approach - ER '93 and over 670,000 other books are available for Amazon Kindle - Amazon's new wireless reading device. Learn more

Click to LOOK INSIDE!



Entity-Relationship Approach - ER '93: 12th International Conference on the Entity-Relationship Approach, Arlington, Texas, USA, December 15 - 17, 1993. Proceedings (Lecture Notes in Computer Science) [Paperback] Ramez A. Elmasri (Editor), Vram Kouramajian (Editor), Bernhard Thalheim (Editor)

★★★★☆ (1 customer review)

Share

List Price: ~~\$98.00~~

Price: \$88.20 & this item ships for FREE with Super Saver Shipping. Details

You Save: \$9.80 (10%)

In Stock.

Ships from and sold by Amazon.com. Gift-wrap available.

Want it delivered Wednesday, September 22? Order it in the next 8 hours and 55 minutes, and choose One-Day Shipping at checkout. Details

10 new from \$80.68 8 used from \$22.81

Quantity: 1

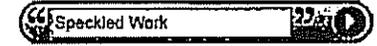
Add to Cart

Sign in to turn on 1-Click ordering

Amazon Prime Free Trial required. Sign up when you check out. Learn More

Add to Wish List

Express Checkout with PayPhrase



What's this? | Create PayPhrase

More Buying Choices

18 used & new from \$22.81

Have one to sell? Sell yours here.

Share your own customer images

Search inside this book



Start reading Entity-Relationship Approach - ER '93 on your Kindle in under a minute.

Don't have a Kindle? Get your Kindle here, or download a FREE Kindle Reading App.



Join Amazon Student and get FREE Two-Day Shipping for one year with Amazon Prime shipping benefits.

Formats	Amazon Price	Now from	Used from
Kindle Edition	\$78.40	-	-
Paperback	\$88.20	\$80.68	\$22.81

Show 1 more format

Customers Viewing This Page May Be Interested in These Sponsored Links (What's this?)

- ERD Design Tool [www.Embarcadero.com](http://www.Embarcadero.com) - Draw Entity Relationship Diagrams With ER/Studio. Try It Free!

See a problem with these advertisements? Let us know

Advertise on Amazon

Editorial Reviews

Product Description

This volume contains the proceedings of the 12th International Conference of the Entity-Relationship Approach, held in

Arlington, Texas in December 1993; it contains the revised versions of 42 papers selected for presentation at the conference from a total of 87 submissions.

The volume presents many of the most important results on the ERA published since the predecessor conference ER '92. It is organized in sections on object-oriented models, query languages, applications of the ER model, knowledge-based modeling, data modeling, schema integration, reuse and reengineering, integrating ER and object-orientation, conceptual clustering, modelling time and data semantics.

---

## Product Details

**Paperback:** 531 pages

**Publisher:** Springer; 1 edition (August 26, 1994)

**Language:** English

**ISBN-10:** 3540582177

**ISBN-13:** 978-3540582175

**Product Dimensions:** 9 x 6.1 x 1.3 inches

**Shipping Weight:** 1.6 pounds (View shipping rates and policies)

**Average Customer Review:** ★★★★★  (1 customer review)

**Amazon Bestsellers Rank:** #7,311,176 in Books (See [Top 100 in Books](#))

Would you like to update product info, give feedback on images, or tell us about a lower price?

---

## More About the Author



Discover books, learn about writers, read author blogs, and more.

> Visit Amazon's Ramez Elmasri Page

---

## Inside This Book (learn more)

### First Sentence:

Object-oriented databases offer a number of features for modeling objects and their associated, encapsulated methods. Read the first page

### Key Phrases - Statistically Improbable Phrases (SIPs): (learn more)

binary cardinality, nonnumerical domain, binary imposition, fuzzy query condition, binary cardinalities, derivable data, database schema components, logical extensibility, repository meta model, planned capacity usage, domain concept hierarchy, major entity types, ternary cardinality, system modelling techniques, conceptual normalization, schema evolution operations, generic meta model, strong entity type, data structure extraction, users decision tree, executable schema, conceptual query language, binary collection, complementary schemas, integration assertions

### Key Phrases - Capitalized Phrases (CAPs): (learn more)

International Conference, Data Engineering, Computing Surveys, Management of Data, San Mateo, Naive Business Model, New York, Computer Society Press, Prentice Hall, Very Large Databases, Information Sciences, Redwood City, Commonsense Business Reasoner, Englewood Cliffs, Int'l Conf, Linguistic Base, Morgan Kaufmann, University of Alberta, Collection Model, The Netherlands, Candidate Base, Elsevier Science Publishers, Los Angeles, North Holland, Explicit Binary Permission

### New!

Books on Related Topics | Concordance | Text Stats

### Browse Sample Pages:

Front Cover | Table of Contents | First Pages | Index | Back Cover | Surprise Me!

### Search Inside This Book:



---

## Citations (learn more)

This book cites 77 books:

- Database Systems: Concepts, Languages & Architectures by Paolo Atzeni on 9 pages
- Relational Databases by Chao-Chih Yang on 7 pages
- Fundamentals of Database Systems (5th Edition) by Ramez Elmasri on 6 pages
- Temporal Databases by Richard Snodgrass on 5 pages
- Conceptual Schema and Relational Database Design by Terry Halpin on 5 pages

See all 77 books this book cites

Books on Related Topics *(learn more)*



**Database Modeling and Design, Third Edition** by Toby J. Teorey

- Discusses:**
- ternary cardinality
  - binary relationships
  - Computing Surveys



**Database Systems** by Thomas M. Connolly

- Discusses:**
- binary cardinality
  - binary cardinalities
  - strong entity type



**Readings in Object-Oriented Database Systems** by Stanley B. Zdonik

- Discusses:**
- schema change operations
  - International Conference
  - Computing Surveys



**Advanced Database Technology and Design** by Mario Plattini

- Discusses:**
- logical extensibility
  - Data Engineering
  - Computing Surveys

Tag this product *(What's this?)*

Think of a tag as a keyword or label you consider is strongly related to this product. Tags will help all customers organize and find favorite items.

> Explore product tags

Search Products Tagged with

> See most popular tags

Customer Reviews

1 Review

- 5 star: (0)
- 4 star: (1)
- 3 star: (0)
- 2 star: (0)
- 1 star: (0)

**Average Customer Review**  
 ★★★★★ (1 customer review)

Share your thoughts with other customers:

[Create your own review](#)

Most Helpful Customer Reviews

0 of 2 people found the following review helpful:

★★★★☆ **profundo y bastante completo**, November 24, 1998

By A Customer

This review is from: *Entity-Relationship Approach-Er '93: 12th International Conference on the Entity-Relationship Approach Arlington, Texas, Usa, December 15-17, 1993*. (Lecture Notes in Computer Science) (Paperback)

Cubre los aspectos más importantes acerca de las Bases de Datos. En la sección 3.3 los alumnos suelen encontrar los conceptos un poco ambiguos. Lo mismo sucede con al exposición de la notación del DER, donde a veces utiliza extensiones de la notación antes de presentarla. A pesar de todo puede decirse que, en general, es muy claro en su exposición y agradable su lectura.

[Report this](#) , [Permalink](#)

ADVERTISEMENT

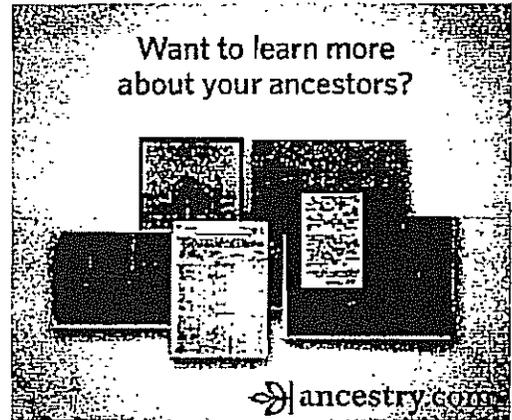
Help other customers find the most helpful reviews

Was this review helpful to you?  Yes  No

Comment

Share your thoughts with other customers: [Create your own review](#)

> [See the customer review...](#)



[Ad feedback](#)

Search Customer Reviews

Only search this product's reviews

---

## Customer Discussions

---

## Listmania!

---

## So You'd Like to...

---

## Look for Similar Items by Category

[Books](#) > [Computers & Internet](#) > [Computer Science](#) > [Artificial Intelligence](#)

[Books](#) > [Computers & Internet](#) > [Databases](#)

[Books](#) > [Computers & Internet](#) > [Programming](#) > [Software Design, Testing & Engineering](#) > [Object-Oriented Design](#)

[Books](#) > [New & Used Textbooks](#) > [Computer Science](#)

[Books](#) > [Science](#) > [Mathematics](#)

---

## Look for Similar Items by Subject

### Search Books by subject:

- [Computing: Professional & Programming](#)
- [Computing & information technology](#)
- [Database software](#)
- [Computers - General Information](#)
- [Computer Science](#)
- [Computers / Artificial Intelligence](#)
- [Computers / Computer Science](#)
- [Computers / Data Processing / Storage & Retrieval](#)
- [Computers / Database Management / General](#)
- [Computers / Information Technology](#)

- Computers : Computer Science
- Computers : Database Management - General
- Database Management - General
- Information Storage & Retrieval
- Mathematics / Logic
- System Administration - Storage & Retrieval
- Database design
- Entity-relationship modeling

Find books matching ALL checked subjects

i.e., each book must be in subject 1 AND subject 2 AND ...

ADVERTISEMENT

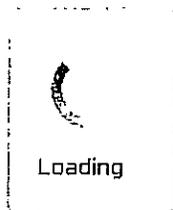


[Ad feedback](#)

### Feedback

- ▶ If you need help or have a question for Customer Service, **contact us**.
- ▶ Would you like to **update product info**, **give feedback on images**, or **tell us about a lower price**?
- ▶ Is there any other feedback you would like to provide? **Click here**

### Your Recent History (What's this?)



Get to Know Us  
 Careers  
 Investor Relations  
 Press Releases  
 Amazon and Our Planet

Make Money with Us  
 Sell on Amazon  
 Join Associates  
 Advertise Your Products  
 Self-publish with Us  
 › See all

Let Us Help You  
 Shipping Rates & Policies  
 Amazon Prime  
 Returns  
 Help

amazon.com®

Canada China France Germany Japan United Kingdom

[AmazonWireless](#)  
Cellphones &  
Wireless Plans

[Askville](#)  
Community  
Answers

[Audible](#)  
Download  
Audio Books

[DPReview](#)  
Digital  
Photography

[Endless](#)  
Shoes  
& More

[Fabric](#)  
Sewing, Quilting  
& Knitting

[IMDb](#)  
Movies, TV  
& Celebrities

[Shopbop](#)  
Designer  
Fashion Brands

[Small Parts](#)  
Industrial  
Supplies

[Warehouse Deals](#)  
Open Box  
Discounts

[Zappos](#)  
Shoes &  
Clothing

[Conditions of Use](#) [Privacy Notice](#) © 1996-2010, Amazon.com, Inc. or its affiliates

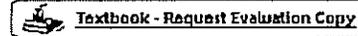
## Database Design Using Entity-Relationship Diagrams

Sikha Bagui, *University of West Florida, Pensacola, Florida, USA*; Richard Earp, *University of West Florida, Pensacola, USA*  
Series: Foundations of Database Design



Price: \$83.95  
Cat. #: AU1548  
ISBN: 9780849315480  
ISBN 10: 0849315484  
Publication Date: June 27, 2003  
Number of Pages: 242  
Availability: In Stock  
Binding(s): Hardback | Available in e-book!

Email this title to a friend



### Related Titles

- **Networking Systems Design and Development**  
Lee Chao, *University of Houston-Victoria, Sugar Land, Texas, USA*  
Publication Date: December 21, 2009  
Price: \$89.95
- **Database Development and Management**  
Lee Chao, *University of Houston-Victoria, Sugar Land, Texas, USA*  
Publication Date: January 13, 2006  
Price: \$106.95
- **Software Testing: A Craftsman's Approach, Third Edition**  
Paul C. Jorgensen, *Grand Valley State University, Allendale, Michigan, USA*  
Publication Date: February 15, 2008  
Price: \$99.95
- **Physical Database Design Using Oracle**  
Donald K. Burleson, *Burleson Consulting, Kiltrell, North Carolina, USA*  
Publication Date: July 27, 2004  
Price: \$98.95
- **Data Mining: Technologies, Techniques, Tools, and Trends**  
Bhavani Thuraisingham, *The University of Texas at Dallas, Richardson, USA*  
Publication Date: December 18, 1998  
Price: \$109.95
- **XML Databases and the Semantic Web**  
Bhavani Thuraisingham, *The University of Texas at Dallas, Richardson, USA*  
Publication Date: March 27, 2002  
Price: \$109.95
- **Grid Database Design**  
April J. Wells, *Oracle, Austin, Texas, USA*  
Publication Date: May 26, 2005  
Price: \$88.95
- **Managing and Mining Multimedia Databases**  
Bhavani Thuraisingham, *The University of Texas at Dallas, Richardson, USA*  
Publication Date: June 28, 2001  
Price: \$109.95
- **Database and Applications Security: Integrating Information Security and Data Management**  
Bhavani Thuraisingham, *The University of Texas at Dallas, Richardson, USA*  
Publication Date: May 26, 2005  
Price: \$87.95
- **Oracle 11i E-Business Suite from the Front Lines**  
April J. Wells, *Oracle, Austin, Texas, USA*  
Publication Date: December 29, 2003  
Price: \$87.95
- **Building Web Applications with C# and .NET: A Complete Reference**  
Dudley W. Gill, *Gill Associates, Inc., Belmar, New Jersey, USA*  
Publication Date: December 02, 2002  
Price: \$92.95

Description | **Table of Contents**

#### Features

- Demonstrates testing of a newly-constructed database via the theory of normal forms and referential integrity constraints
- Provides a data modeling schema that defines entities, relationships, attributes
- Discusses structural constraints in relationships
- Includes corresponding grammar and mapping rules
- Explores generalizations and specializations
- Illustrates a reverse mapping design for mapping a relational database backward to an ER diagram—performed when database is in use but no diagram exists

#### Summary

Entity-relationship (E-R) diagrams are time-tested models for database development well-known for their usefulness in mapping out clear database designs. Also commonly known is how difficult it is to master them. With this comprehensive guide, database designers and developers can quickly learn all the ins and outs of E-R diagramming to become expert database designers. Because E-R diagrams are so fundamental to database design, this book is also an indispensable text for teaching computer science students the basics of database development.

Database Design Using Entity-Relationship Diagrams clarifies E-R diagramming by defining it in terms of requirements (end user requests) and specifications (designer feedback to those requests). The book explains how open communication between designers and end users is critical to developing usable, easy-to-understand E-R diagrams that model both requirements and specifications.

The authors explain, in an intuitive, informal manner, how to develop an E-R diagram, how to map it to a database, and how the resulting database can be tested. This definitive guide is a basic component for any database course, and is also an invaluable reference that database professionals can use throughout their careers.

introbu>Features

- > Logical Database Design Principles  
 John Garmany, *Burleson Enterprises, Inc., Colorado, USA*; Jeff Walker, *Largo, Florida, USA*; Terry Clark, *Las Vegas, Nevada, USA*  
 Publication Date: May 12, 2005  
 Price: \$98.95
- > Strategic Data Warehousing: Achieving Alignment with Business  
 Neera Bhansali, *IMEMS Corp.*  
 Publication Date: July 29, 2009  
 Price: \$69.95
- > Programming Languages for Business Problem Solving  
 Shouhong Wang, *University of Massachusetts Dartmouth, USA*; Hal Wang, *Saint Mary's University, Halifax, Nova Scotia, Canada*  
 Publication Date: November 08, 2007  
 Price: \$95.95
- > The Complete Book of Middleware  
 Judith M. Myerson, *IT Consultant, Philadelphia, Pennsylvania, USA*  
 Publication Date: March 05, 2002  
 Price: \$92.95
- > Modeling Software Behavior: A Craftsman's Approach  
 Paul C. Jorgensen, *Grand Valley State University, Allendale, Michigan, USA*  
 Publication Date: July 21, 2009  
 Price: \$89.95
- > Implementing Enterprise Content Management Systems  
 Azad Adam  
 Publication Date: July 01, 2011  
 Price: \$69.95
- > Implementing Electronic Document and Record Management Systems  
 Azad Adam, *Independent Consultant, London, UK*  
 Publication Date: August 24, 2007  
 Price: \$98.95

Copyright © 2010 Taylor & Francis LLC. All Rights Reserved

Get your MySQL data into MS Access for local off-line analysis and reporting.

This program is an efficient and easy to use database converter.



### Full Convert MySQL Edition 3.2

➤ Spectral Core GmbH

Full Convert is the best application to convert structure and data of databases.



### MySQL Sybase Anywhere Import, Export & Convert Software 7.0

➤ Sobolsoft

## Other Convert Mysql Into er Diagram Downloads:



### MSSQL2MySQL Pro 1.4

➤ DMSoft Technologies

MSSQL to MySQL Pro is designed to convert databases from Microsoft SQL to MySQL and vice versa. This...



### Full Convert MySQL Edition Trial 2.0

➤ Spectral Core GmbH



### Convert Mysql to Oracle 1.2

➤ www.convert.cc



### MySQL Export Table To XML File Software 7.0

➤ Sobolsoft

Convert MySQL data in one table to an XML file.



### Excel to MySQL Import, Export & Convert 1.1

➤ Sobolsoft



### MySQL PostgreSQL Import, Export & Convert Software 7.0

➤ Sobolsoft

Transfer tables to and from MySQL and PostgreSQL databases.



### MySQL IBM DB2 Import, Export & Convert Software 7.0

➤ Sobolsoft

Transfer tables to and from MySQL and IBM DB2 databases.



### MySQL Oracle Import, Export & Convert Software 7.0

➤ Sobolsoft

Transfer tables to and from MySQL and Oracle databases.



### MySQL Sybase SQL Anywhere Import, Export & Convert Software 7.0

➤ Sobolsoft



### MySQL Sybase ASE Import, Export & Convert Software 7.0

➤ Sobolsoft

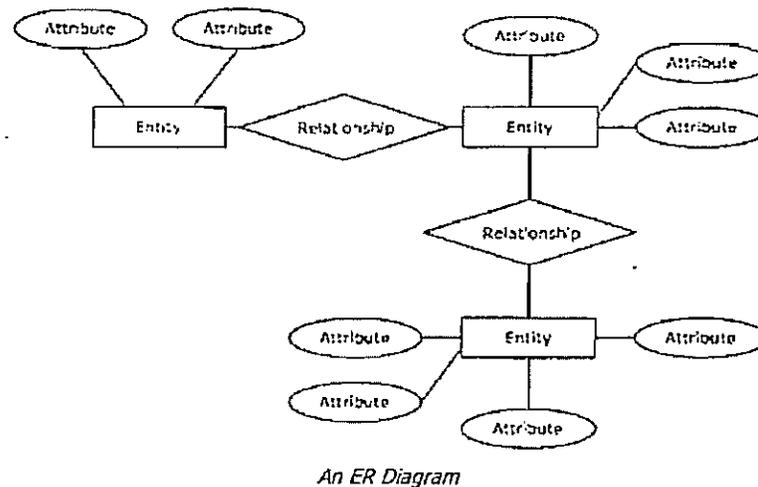


### Convert Mysql to Access 4.0

➤ www.5stardatabasesoftware.com

## What are Entity Relationship Diagrams?

Entity Relationship Diagrams (ERDs) illustrate the logical structure of databases.



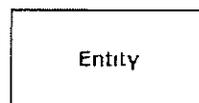
### Entity Relationship Diagram Notations

Peter Chen developed ERDs in 1976. Since then Charles Bachman and James Martin have added some slight refinements to the basic ERD principles.

#### Entity

An entity is an object or concept about which you want to store information.

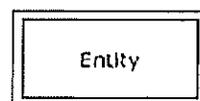
Learn how to edit text on an entity.



#### Weak Entity

A weak entity is an entity that must be defined by a foreign key relationship with another entity as it cannot be uniquely identified by its own attributes alone.

Learn how to edit text on this object.



#### Key attribute

A key attribute is the unique, distinguishing characteristic of the entity. For example, an employee's social security number might be the employee's key attribute.



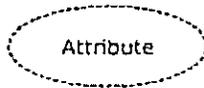
### Multivalued attribute

A multivalued attribute can have more than one value. For example, an employee entity can have multiple skill values.



### Derived attribute

A derived attribute is based on another attribute. For example, an employee's monthly salary is based on the employee's annual salary.

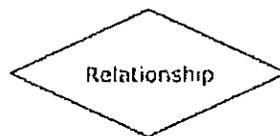


### Relationships

Relationships illustrate how two entities share information in the database structure.

Learn how to draw relationships:

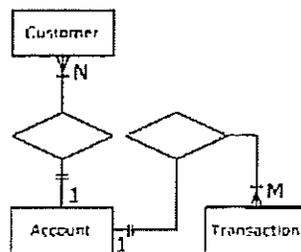
First, connect the two entities, then drop the relationship notation on the line.



### Cardinality

Cardinality specifies how many instances of an entity relate to one instance of another entity.

Ordinality is also closely linked to cardinality. While cardinality specifies the occurrences of a relationship, ordinality describes the relationship as either mandatory or optional. In other words, cardinality specifies the maximum number of relationships and ordinality specifies the absolute minimum number of relationships.

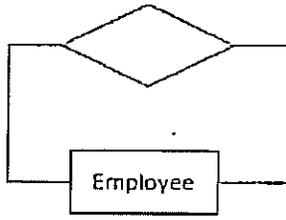


[Click here for more cardinality notations](#)

To learn how to express cardinality in SmartDraw, [click here](#).

### Recursive relationship

In some cases, entities can be self-linked. For example, employees can supervise other employees.



# Non-spatial Database Models

34

by Thomas H. Meyer, Mapping Sciences Laboratory, Texas A&M University, USA

This unit is part of the *NCGLA Core Curriculum in Geographic Information Science*. These materials may be used for study, research, and education, but please credit the author, Thomas H. Meyer, and the project, *NCGLA Core Curriculum in GIScience*. All commercial rights reserved. Copyright 1997 by Thomas H. Meyer.

Your comments on these materials are welcome. A link to an evaluation form is provided at the end of this document.

---

## Advanced Organizer

### Topics covered in this unit

- This unit introduces the terms and concepts needed to understand non-spatial databases and their underlying data models, including:
  - a motivation of the need for database management systems
  - an overview of database terminology
  - a description of non-spatial data models

### Intended Learning Outcomes

- After learning the material covered in this unit, students should be able to:
  - explain the purpose of a database management system
  - list the major non-spatial data models and their features
  - identify the primary distinctions between the major non-spatial data models

### Instructors' Notes

### Full Table of Contents

### Metadata and Revision History

---

# Non-spatial Database Models

## 1. Motivation: Why database management systems?

- Database management systems (DBMSs) are very good at organizing and managing large collections of persistent data.
  - We use DBMSs to help cope with large amounts of data because, when problems get big, they get hard.
    - Consider the task of finding a particular book in a typical university library.
    - Now, reconsider that same task if the library doesn't keep the books arranged in any particular order or if the library has no indexes.
  - Using a big collection of unorganized things is practically impossible. Structure turns data into information.
  - *Persistence* means that the data exist permanently; they do not disappear when the computer is shut off.

- DBMSs are like suitcases: they are somewhere to put stuff so that it's all in one place and easy to get to.
  - DBMSs help protect data from unauthorized access.
  - DBMSs help protect data from accidental corruption or loss due to:
    - hardware failures such as power outages and computer crashes
    - software failures such as operating system crashes
  - DBMSs allow concurrent access, meaning that a single data set can be accessed by more than one user at a time
    - virtually all commercial database applications require the data entry staff to have access to the database simultaneously.
      - For example, an airline reservation system cannot restrict access to the database to a single travel agent.
    - concurrent data access introduces unwanted problems caused by two users manipulating exactly the same data at exactly the same time.
      - These problems can cause the database to be corrupted or for a user's interface program to never complete its query.
      - These problems are analogous to road intersections: if there are no traffic lights or stop signs, havoc will ensue.
    - DBMSs provide mechanisms to prevent concurrent access problems; these mechanisms are collectively called *concurrency control*.
  - A *distributed* DBMS allows a single database to be split apart such that its pieces reside at geographically separated sites.
    - this can provide performance improvements by eliminating transmitting the data across a relatively slow long distance communication channel (it's a lot faster to have the database on your hard drive than to access it across an Ethernet or via a modem)
    - this can reduce concurrency control bottlenecks by giving each user that part of the database which they need rather than having all the users compete for access to the whole database
  - DBMSs are not necessarily meant for data analysis; that is more the job of a spread sheet or some other special-purpose analysis tool.
    - DBMSs are general-purpose tools. It is basically irrelevant to the DBMS what is stored within it. Software design principles suggest de-coupling domain specific analysis packages from the DBMS to keep the division of labor clear.
    - DBMSs are very good at retrieving a relatively small portion of the database and passing it along for detailed analysis by a tool designed for that purpose.
    - DBMSs often allow integrity constraints to be imposed on the data to insure validity and consistency. These rules can interfere with ad-hoc analysis in which the user manipulates the data without any preconceived ideas of how the data should relate to each other.
    - DBMSs often do not have adequate facilities to perform complicated calculations; some have no such facilities whatsoever.
- 

## 2. Fundamental Concepts and Terminology

- This section presents a few common database concepts and terms.

### 2.1. Data

- Data are facts. Some facts are more important to us than others. Some facts are important enough to warrant keeping track of them in a formal, organized way.
- Important data are like the valuables we keep in a bank. They are a small subset of our total possessions but they are so important that we protect them by putting them in a special, safe place.

- "Data" is a plural. The singular of "data" is "datum".
- "Data" is a broad concept that can include things such as pictures (binary images), programs, and rules. Informally, *data* are the things you want to store in a *database*.

## 2.2. Spatial vs. Non-spatial Data

- Spatial data includes location, shape, size, and orientation.
  - For example, consider a particular square:
    - its center (the intersection of its diagonals) specifies its location
    - its shape is a square
    - the length of one of its sides specifies its size
    - the angle its diagonals make with, say, the *x*-axis specifies its orientation.
- Spatial data includes spatial relationships. For example, the arrangement of ten bowling pins is spatial data.
- Non-spatial data (also called *attribute* or *characteristic* data) is that information which is independent of all geometric considerations.
  - For example, a person's height, mass, and age are non-spatial data because they are independent of the person's location.
  - It's interesting to note that, while mass is non-spatial data, weight is spatial data in the sense that something's weight is very much dependent on its location!
- It is possible to ignore the distinction between spatial and non-spatial data. However, there are fundamental differences between them:
  - spatial data are generally multi-dimensional and autocorrelated.
  - non-spatial data are generally one-dimensional and independent.
- These distinctions put spatial and non-spatial data into different philosophical camps with far-reaching implications for conceptual, processing, and storage issues.
  - For example, sorting is perhaps the most common and important non-spatial data processing function that is performed.
  - It is not obvious how to even sort locational data such that all points end up "nearby" their nearest neighbors.
- These distinctions justify a separate consideration of spatial and non-spatial data models. This unit limits its attention to the latter unless otherwise specified.

## 2.3. Database

- A database is a collection of facts, a set of data.
  - It is like the contents of a bank's vault.
- The information in a phone book is an example of a database.
  - Pay carefully attention to the fact that the book itself is not the database.
  - Rather, the database is the information stored on the pages of the book, not the pieces of paper with ink on them.

## 2.4. Repository

- A repository is a structure that stores and protects data.
- Repositories provide the following functionality:
  - add (insert) data to the repository
  - retrieve (find, select) data in the repository
  - delete data from the repository
- Some repositories allow data to be changed, to be updated.

RS875

- This is not strictly necessary because an update can be accomplished by retrieving a copy of the datum from the repository, updating the copy, deleting the old datum from the repository, and inserting the updated datum into storage.
- Repositories are like a bank vault. They exist mainly to protect their contents from theft and accidental destruction.
  - Security: repositories are typically password protected, many have much more elaborate security mechanisms.
  - Robustness: Accidental data loss is safeguarded against via the *transaction* mechanism.
    - A *transaction* is a sequence of database manipulation operations.
    - Transactions have the property that, if they are interrupted before they complete, the database will be restored to a self-consistent state, usually the one before the transaction began.
    - If the transaction completes, the database will be in a self-consistent state.
    - Transactions protect the data from power failures, system crashes, and concurrent user interference.
- An example of a commercially available repository is Kala (Simmel and Godard 1991).

## 2.5. Database Management System (DBMS)

- A *database management system* is a data repository along with a user interface providing for the manipulation and administration of a database. A phone book is an example of a DBMS.
- Unless specified otherwise, a DBMS will hereafter be understood to be a software system, a program (or suite of programs) that is run on a digital computer. A few examples of commercially available DBMSs include Gemstone, O<sub>2</sub>, Versant, Mattise, Codasyl, Sybase, Oracle, DB2, Access, and dBase.
- A DBMS is like a full-service bank, providing many features and services missing from the comparatively Spartan repository.

## 2.6. Queries

- Many DBMSs provide a user interface consisting of some sort of formal language.
- A *data definition language* (DDL) is used to specify which data will be stored in the database and how they are related.
- A *data manipulation language* (DML) is used to add, retrieve, update, and delete data in the DBMS.
- A *query* is often taken as a statement or group of statements in either a DDL or a DML or both. Some researchers view queries as read-only operations, no data modifications are allowed (Codd 1990, p. 21).
- A *query language* is a formal language that implements a DDL, a DML, or both. Examples of query languages include SQL (Structured Query Language), QUEL, ISBL, and Query-by-Example.

## 2.7. Data Models

- A *data model* is mathematical formalism consisting of two parts (Ullman 1988, p.32):
  - A notation for describing data, and
  - A set of operations used to manipulate that data.
- A data model is a way of organizing a collection of facts pertaining to a system under investigation.
- Data models provide a way of thinking about the world, a way of organizing the phenomena that interest us.
- They can be thought of as an abstract language, a collection of words along with a grammar by which we describe our subject.
  - By choosing a language, we pay the price of being constrained to form expressions whose words are limited to those in the language and whose sentence structure is governed by the language's grammar.
  - We are not free to use random collections of symbols for words nor can we put the words together in any *ad hoc* fashion.

- A major benefit we receive by following a data model stems from the theoretical foundation of the model.
    - From the theory emerges the power of analysis, the ability to extract inferences and to create deductions that emerge from the raw data.
  - Different models provide different conceptualizations of the world; they have different outlooks and different perspectives.
    - There is no universally agreed upon best data model so this unit presents the most common ones.
  - DBMSs are seen to be composed of three *levels of abstraction*:
    - *physical*: this is the implementation of the database in a digital computer. It is concerned with things like storage structures and access method data structures.
    - *conceptual*: this is the expression of the database designer's model of the real world in the language of the data model.
    - *view*: different user groups can be given access to different portions of the database. A user groups portion of the database is called their *view*.
  - This unit is concerned mostly with the conceptual level.
- 

### 3. Common Data Models

- This section presents an overview of the most common data models

#### 3.1. Entity-Relationship Model

- The Entity-Relationship (ER) model is generally attributed to (Chen 1976).
- The ER model envisions the world as comprised of *entities* that are associated with each other by *relationships*. All of the entities of a particular type are collected together into *entity sets*.
- Entity sets and relationships can be depicted graphically in an ER-diagram.

##### 3.1.1. Entities

- Entities are distinguishable "real-world" objects such as employees, maps, airplanes, or bus schedules.
  - "Distinguishable" means that all entities can be uniquely identified.
  - Entities have common attributes that define what it means to be such an entity.
  - Any particular real-world object does not necessarily have a single or best representation as an entity.
    - For any given real-world object, different modelers can choose different sets of attributes of the object that are of interest to their particular situation.
    - This results in the same object being modeled differently.
- Entities are collected into *entity sets*.
  - Entity sets are depicted as rectangles in *ER diagrams*.
  - Their attributes are depicted as ellipses attached to the rectangles by lines.

##### 3.1.2. Relationships

- A relationship is a list of entity sets.
  - Notation: two entity sets *A* and *B* that stand in relationship *r* is written  $A \text{ r } B$ . See the next bullet for examples.
- Types of relationships (see Figure 1.):
  - aggregating relationships:

RS877

- one-one: if  $A \text{ r } B$  and  $r$  is one-one then each entity of  $B$  is in relationship with at most one entity of  $A$  and vice-versa.
    - For example, if *CAPTAIN* commands *VESSEL* and *commands* is one-one then, in our model, each vessel has at most one captain and each captain commands at most one vessel at a time.
  - many-one: if  $A \text{ r } B$  and  $r$  is many-one then each entity of  $A$  is in relationship with at most one entity of  $B$  but not vice-versa.
    - For example, if *CREW assigned-to VESSEL* and *assigned-to* is many-one then, in our model, a vessel has many crew members but a crew member is assigned to only one vessel.
  - many-many: if  $A \text{ r } B$  and  $r$  is many-many then each entity of  $A$  can be in relationship with any number of  $B$  entities and vice-versa.
    - For example, if *VESSEL patrols REGION* and *patrols* is many-many then, in our model, a vessel patrols many regions and a region is patrolled by many ships.
- **isa** (read "is a") relationships: if  $A \text{ isa } B$  then  $A$  is a specialization of  $B$ , or, conversely,  $B$  is a generalization of  $A$ .
    - For example, if *CAPTAIN isa CREW* then, in our model, captains have all the attributes of crew members but not vice versa.
    - The *isa* relationship allows hierarchies to be established among entity sets.
- A Relationship is depicted by a lozenge with lines connecting it to the relevant entity sets.
  - The Entity-Relationship model lacks an underlying formalism and is, therefore, used more for general conceptualization than for creating physical models
    - (indeed, some authors do not acknowledge the ER model as a data model at all).
    - It is not uncommon for a conceptual design to be expressed in the ER model and then "translated" into another model for implementation.

### 3.2. The Network Model

- The network data model is based upon the concept of a *structure* such as is found in programming languages like C or Pascal.
  - ER entities can be modeled as structures with the entity's attributes corresponding to the structure's fields.
  - Entities are distinguished by their location, *i.e.*, the "physical" address of the structure that is holding them. Thus, two structures of identical value represent two separate entities.
- Entity sets can be implemented as files whose records match the structures.
- Relationships are created with explicit linkages (*viz.* pointers) from structure to structure.
- Codasyl is an example of a DBMS based on the network model (Olle 1978).
- The network model has no formal semantics nor a high-level query language. Database manipulation was done via custom programs often written in COBOL.
- Network model databases are hand-coded and, therefore, can be very efficient in their space utilization and query execution times; all the relationships are "hardwired" or precomputed and built into the structure of the database itself.
- The price for such performance is inflexibility and great difficulty of use (among many other things).

### 3.3. The Relational Model

- The relational model was introduced by Codd (1970) and has been the inspiration of an entire generation of database management systems that are based on the concept of a *relation* which is a set of *tuples*.

### 3.3.1. Tuples

- A tuple is a set of facts that are related to each other in some way (perhaps only by the fact they've been put together in a set).
- Each fact in a tuple is a datum whose value comes from a specified domain (e.g., the domain of all integers, the domain of all character strings of length 255 or less, etc.)
- Formally, let  $D_1, \dots, D_n$  be  $n$  sets of values constituting  $n$  domains ( $n$  is usually greater than zero but that is not strictly necessary). A **tuple**  $t$  is a set of values  $t = \{d_1, \dots, d_n\}$ , such that  $d_1$  is an element of  $D_1, \dots$ , and  $d_n$  is an element of  $D_n$ . The domains are called **attributes**.

### 3.3.2. Relations

- Formally, let  $D_1, \dots, D_n$  be  $n$  domains. A **relation**  $R$  is a set of tuples over the Cartesian product  $D_1 \times \dots \times D_n$ .
- In English, a relation is a (possibly complete) subset of all the possible tuples formed by the Cartesian product of the domains.
- Since tuples are sets (of values) and a relation is also a set (of tuples), relations are sets of sets.
  - a file is a list of records
  - a table is a list of rows
  - a relation is a set of tuples
- Relations are naturally represented as tables.
  - Tables are not relations because relations cannot have duplicate tuples and there is no such stricture on tables. However, it is perhaps convenient to think about relations as tables so long as the distinction remains clear.
  - Most (if not all) commercial "relational" DBMSs violate this principle: they allow duplicate tuples.
- The use of relations as a data modeling tool becomes apparent when we have a relation, say, "OUR\_DEM" with fields  $\{quadname, zone\_code, mappingcenter\}$ .
  - It happens that the USGS has a digital elevation model named "PLACITAS NM" in UTM zone 13 that was created by the Forest Service Mapping Center.
  - Then, the presence of a tuple in the OUR\_DEM relation whose
    - *quadname* attribute has the value "PLACITAS NM" and
    - *zone\_code* attribute has the value "13", and
    - *mappingcenter* attribute has the value "FS",
  - indicates that we have the Placitas DEM in our possession.

### 3.3.3. Tuples, Relations and Keys

- Relations are *sets* of tuples; consequently, no two tuples that are elements of the same relation can have identical values for all their attributes. That is to say, there are no duplicate tuples in a relation.
- All tuples in a relation can be distinguished by the values of their attributes.
  - Any set of attributes whose values *necessarily* uniquely identify a tuple are said to be a **key**.
- Database designers choose some attribute set to be a key for their database's relations.
  - This key is known as the **primary key**.
- If the primary key of one table appears as an attribute of a different relation, the key is known as a **foreign key** in the other relation.
- A key uniquely identifies its tuple. Therefore, a tuple's key is often used as a surrogate for the entire tuple.

### 3.3.4. Relationships

RS879

- Not surprisingly, the relational model represents relationships with relations.
- Figure 2 depicts a relational database that was designed from the ER-diagram developed above.
  - Key attributes are denoted in bold face.
- If you wish to work with these examples, you can download either:
  1. the Microsoft Access97 .mdb file by **SAVING** the files at:  
<http://ncgia.ucsb.edu/giscc/units/u045/data/samples.mdb>
    - **NOTE: you can do this by right clicking (PC and Unix computers only) on the link above**
  2. ASCII text for the tables by clicking their names below.  
 CAPTAIN CREW PATROLS  
 REGION SOTL VESSEL
    - The attribute names are on the first row, character strings are delimited with double quotes (") and the fields are comma delimited.
- Aggregating relationships are represented by embedding the primary key of one relation into another relation as a foreign key:
  - *one-one*: if  $A \text{ r } B$  and  $r$  is one-one then the primary key of  $A$  can be embedded in  $B$  or vice versa or both.
    - For example, suppose *CAPTAIN* commands *VESSEL* and that *commands* is one-one.
    - Suppose further that *cptn\_name* is the primary key of *CAPTAIN* and *vessel\_name* is the primary key of *VESSEL*.
    - Then, *CAPTAIN* could have an attribute *commands* whose value is that of *vessel\_name* for the vessel that captain commands.
    - It is equally reasonable to have an attribute *commanded\_by* in *VESSEL* whose value is that of name for the captain commanding the vessel.
  - *many-one*: if  $A \text{ r } B$  and  $r$  is many-one then the primary key of  $B$  can be embedded in  $A$  but not vice versa.
    - For example, suppose *CREW* assigned-to *VESSEL* and assigned-to is many-one.
    - Suppose further that *crew\_name* is the primary key of *CREW* and *vessel\_name* is the primary key of *VESSEL*.
    - Then, *CREW* could have an attribute *assigned\_to* whose value is that of *vessel\_name* for the vessel this crew member serves on.
    - However, *VESSEL* cannot have an attribute *roster* because *roster* would have to be a set (many crew members per vessel) and the relational model stipulates that all domains are atomic; no collections.
  - *many-many*: if  $A \text{ r } B$  and  $r$  is many-many then neither primary key can be embedded the other table. Again, the difficult lies in the atomicity rule for domains. So, for a many-many relationship, we must create a separate relation whose attributes include but are not limited to the primary keys from  $A$  and  $B$ .
    - For example, if *VESSEL* patrols *REGION* and *patrols* is many-many.
    - Suppose further that *vessel\_name* is the primary key of *VESSEL* and *region\_name* is the primary key of *REGION*. Then we have a third relation *PATROLS* with attributes *vessel\_name* and *region\_name*.
    - such relations are sometimes called *join supports*
    - such relations are no different in any way from any other relation
- *isa* relationships are handled as the other relationships:
  - *one-one*: Suppose *CAPTAIN* isa *CREW*.
    - Then there is a one-one relationship between *CAPTAIN* and *CREW* so the primary key of *CREW* can be used as the key in *CAPTAIN*.
    - The one-one nature of this relationship indicates that the two tuples really give details of the same entity; they are sort of like a single tuple that has been split in two.

- many-one: Suppose we are modeling WWII combat vessels, known collectively as "ship(s) of the line" (*SOTL*). It happens that a ship design can be used as the plan for many individual vessels (obviously).
  - The design is known as a "class" and the vessels made to that design are said to belong to that class.
    - For example, the USS Missouri belongs to the Iowa class of battleships.
  - We model this relationship with a relation *SOTL* which has a single tuple for each class of warship. Thus, *VESSEL* *isa* *SOTL*.
  - The *SOTL* relation has attributes that are common to all ships of the line. For WWII vessels, this might include attributes such as the number of primary guns, size of the primary guns, *etc.*
    - The tuple in *SOTL* for the Iowa battleships gives information that is common to all Iowa class battleships (*e.g.*, nine 16-inch guns, *etc.*).
    - The tuple in *VESSEL* for the USS Missouri holds the information specific to that vessel including the fact it belongs in the Iowa class.
    - Therefore, the primary key of *SOTL* is embedded in *VESSEL*, not vice versa.
  - Compare many-one *isa* relationships with one-one *isa* relationships.
  - Ullman restricts relationships to be one-one (Ullman 1988, p. 35).

### 3.3.5. Query Languages

- Codd invented two early languages for dealing with relations: one was algebraic and the other was based on first-order predicate logic (Codd 1971). These languages have the same expressive power.
  - *Relational Algebra*
    - "[an] algebraic notation" where queries are expressed by applying specialized operators to relations (Ullman 1988, p. 53)
    - see (Codd 1990, pp. 61-144) for a presentation of the relational algebra.
  - *Relational Calculus*
    - "[a] logical notation" where queries are expressed by writing logical formulas that the tuples in the answer must satisfy (Ullman 1988, p. 53)
    - see (Ullman 1988, pp. 145-160) for a presentation of the relational calculus.
- The most common commercial query language is the *Structured Query Language*, or *SQL*.
  - Despite its reputation as a relational query language, SQL does not fully support the relational model (it includes things that are not in the model and omits things that are. See (Codd 1988) and (Date 1987)).

### 3.3.6. Relational Database Management System (RDBMS)

- A *relational database management system* is a DBMS based on the relational model as defined by (Codd 1990).
- There is no commercially available DBMS that fully implements the relational model as defined by (Codd 1990). Some are coming closer. Not everyone agrees that this strict lack of conformance is a Bad Thing.

### 3.3.7. Advantages of the Relational Model

- Codd (1990, pp. 431-439) presents many advantages of the relational model. Some of them are highlighted below:
- The relational model is truly a mathematically complete data model. This solid theoretical underpinning is responsible for
  - *ad hoc* query languages whose queries can be automatically compiled, executed, and optimized without resorting to programming
  - correctness: the semantics of the relational algebra are sound and complete

- predictable: the consistent semantics enables users to easily anticipate the result of a given query
- Adaptability: making a change in the structure of the tables in the network model requires programmatic making changes to all the database's queries. As a result, the network model is inflexible in the extreme.
  - The relational model cleanly separates the logical from the physical model and this decoupling mitigates or eliminates these problems.
  - Also, the relational model's integrity constraints are very helpful in ensuring that structural changes did not adversely effect the meaning of the database.
- Multiple views: it is straightforward to present different user groups different views of the same database.
- Concurrency: a full theory of transaction concurrency control exists which depends upon the theoretical formalisms of the relational model.
  - This theory guarantees the correct execution of concurrent queries (indeed, it defines what "correct" means!)

### 3.4. The Object Model

#### 3.4.1. What is the Object Model?

- The word "object" is similar to the Entity-Relationship concept of an "entity" although "object" is more general.
  - I recommend taking "object" in the spirit of "objects in the physical world."
  - Objects are things but they are not limited to physical, tangible things. For example, data structures (e.g., a hash table) can be objects.
  - All objects are distinct and, like the network model, are made distinct by an identifying attribute, the *object ID*.
- Like the other models, the object model assumes that objects can *conceptually* be collected together into meaningful groups. These groups are called *classes*.
- An object grouping is meaningful because objects of the same class must have common attributes, behaviors, and relationships with other objects.
- Unlike entity sets and relations, classes do not actually hold the objects of that class.
  - Classes are purely conceptual.
  - There is nothing in the object model that is equivalent to either a entity set or a relation (there could be but it's not required by the model).
- Like the network model, the relationships among objects are specified via a "physical" link (pointer) between objects.
- According to Rumbaugh *et al.* (1991), "The object model describes the structure of objects in a system " their identity, their relationships to other objects, their attributes, and their operations."
- The DARPA Open OODB project proposes the following as the essential features of the OO data model (Blakeley 1991) and (Rao 1994, p.72):
  - *Object identity*: the ability of the system to distinguish between two different objects that have the same state. The state of an object can be shared by several objects via object identity.
  - *Encapsulation*: a kind of abstraction that enforces a clean separation between the external interface (behavior) of an object and its internal implementation. Encapsulation requires that all access (or interaction) with objects be done by invoking the services provided by their external interface.
  - *Complex state*: the ability to define data types whose implementation has a nested structure. The state of an object could be built from records of primitive types, other objects, or [collections] of objects.
  - *Type extensibility*: the ability to define new data types from previously defined types by enhancing or changing the structure or behavior of the types. Type inheritance is a mechanism used to define new

types by enhancing already existing behavior.

- o *Genericity*: The types of the object data model with which the object query language collaborates must be generic. That is, as a new type is added to the system, it must be queryable.
- There is no universally agreed upon object data model but *The Object-Oriented Database System Manifesto* (Atkinson, *et al.* 1989) gives a framework being considered from which to derive a standard.
- According to Rao (1994), □The object-oriented database (OODB) paradigm is the combination of object-oriented programming language (OOPL) systems and persistent systems. The power of the OODB comes from the seamless treatment of both persistent data, as found in databases, and transient data, as found in executing programs.□
  - o Note that the emphasis with OODB, like the network model, is towards programmers, not end users.
  - o This point is further emphasized by the primary interface to OODBs being OOPLs.
- I suggest (Booch 1994) for a good introduction to object-oriented design and analysis.

### 3.4.2. Inheritance (isa) Relationships and Typing

- Many object-oriented models take classes to be a typing mechanism (for example, Eiffel (Meyer 1997) and C++ (Stroustrup 1997)).
- The *type* of an object is its *class*; an object is an *instance* of its class.
  - o For example, the number 2.3 is an instance of the class of rational numbers.
- Interpreting classes to be types implies the inherent ability of users to create their own data domains.
- Inheritance can be viewed from two perspectives (Cusack 1991):
  - o *incremental*: the process of adding attributes and functions to an existing class (the *base class*).
    - new attributes/functions can added to the new class that were not in the base class.
    - this is a technique for code reuse.
    - no typing information is implied by this relationship.
    - for example, suppose that there is a class *PERSISTENT* that has the functionality of automatically storing its objects in a database. Any class that inherits *PERSISTENT* □magically□ gains the ability to do likewise.
  - o *subtyping*: a technique for arranging class definitions in a hierarchy satisfying the condition that members of the subclass are also members of the superclass.
    - subtyping constitutes the *isa* relationship.
    - old attributes/functions can change type so long as the new type is more specific (it inherited either directly or indirectly) than the original base class.
    - old attributes/functions cannot be removed.
    - old functions can be provided with new implementations so long as the interface to the function remains unchanged (or is changed via specialization as indicated above)
- Various object models span the gambit of inheritance relationships:
  - o full repeated multiple inheritance (Eiffel, C++)
  - o single inheritance (Java)
  - o no inheritance (Actor, Ada)

### 3.4.3. Encapsulation

- Objects *encapsulate* their attributes and the behaviors. This implies:
  - o there is no interaction with an object that does not go through a publicly published interface
  - o objects manipulate their own state; the definition of class includes the object's behavior manifested as functions and procedures.

- an object's state *cannot* be manipulated by anything external to them (at least, not without permission).
- For example, in a non object-oriented language such as C, let's say a programmer writes a procedure to change the values of a structure holding the position of a graphics primitive.
  - In an object-oriented language, the programmer creates a graphics primitive class that has its positional information along with an internal procedure that changes its own position.
  - The programmer □sends a message□ to the object requesting it to change its own position.
- The advantage of encapsulation is that the implementation of any behavior can be changed without effecting any other class in the system. This helps de-couple the classes and reduces the complexity of the system.

#### 3.4.4. Comparison to the Relational Model

- The object model differs from the relational model in (at least) the following ways:
  - The object model allows complex objects to be attribute domains; this is prohibited in the relational model.
  - The only complex type available in the relational model is the relation.
    - The object model restricts all system entities to be objects which is a more general concept than a relation (relations can be objects but not all objects are relations).
  - The relational model allows no duplicate tuples and, consequently, entities are identified by their attribute values.
    - The object model assumes the existence of an object ID which uniquely identifies its object and is, possibly, invisible to the user.
  - Objects are instances of classes and classes constitute the typing system of the model.
    - There is no concept of class-level typing in the relational model; everything is a relation.
    - The relational model supports user-defined domains but this is applied at the attribute level whereas, with the object model, the class is also a type.
    - The equivalent in the relational world would be for relations to constitute types, as well.
  - There is no generally accepted formal object model.
    - The relational model is well-defined, sound, and complete.
  - Relations hold all tuples. There is no equivalent for objects; there is no set or anything else that contains all the objects of a class.
  - There are many higher-order, non-programming query languages for the relational model. There are few equivalents for the object model (UniSQL is an example).
  - The object model is aimed more at programmers than at end users; the reverse is true of the relational model.

## 4. Summary

- This unit has presented many of the more common reasons why people need and use database management systems.
  - A DBMS provides for the storage, retrieval, removal, and analysis of large quantities of data.
  - A DBMS provides safety and security from accidental loss or theft of data.
  - A DBMS is analogous to a full-service bank:
    - data are like valuables,
    - a database is the collection of all the valuables stored in the bank,
    - a repository is like a vault,
    - a DBMS is like the entire, full-service bank.
- The most common data models
  - Entity-Relationship (ER)
    - real-world things are modeled by *entities*.

- all entities of the same type are collected together into an *entity set*.
  - the relationships between entity sets are represented by *relationships*.
  - Network
    - essentially a programmer's database model
    - efficient but inflexible and hard to understand
  - Relational
    - its only complex data type is the *relation*
    - it is the only complete data model
    - aimed at users instead of programmers
    - relational query languages are easier to use than full-blown programming languages
    - rich underlying theory
    - separation of implementation and design
  - Object-Oriented
    - an extension of object-oriented programming
    - no generally agreed upon formal data model
    - great freedom regarding complex data structures
    - inheritance
    - user-defined types
    - encapsulation
- 

## 5. Review and Study Questions

### 5.1. Essay and Short Answer Questions

- What is the difference between a spread sheet program and a database management system? When would you use one or the other?
- Why bother with data modeling? Is there anything "wrong" with just putting data into the database in whatever way seems good at the moment?
- What are the strengths and weaknesses of the Entity-Relationship data model?
- What are the strengths and weaknesses of the Network data model?
- What are the strengths and weaknesses of the Relational data model?
- What are the strengths and weaknesses of the Object-Oriented data model?
- The relational model uses "value" identity, meaning that entities can be distinguished by examining the values of their attributes. Neither the network nor the object model follow this approach.
  - First: do you feel that this distinction is significant? Why or why not?
  - Second: What are the advantages and disadvantages of each approach?
- Explain the distinction between aggregate and inheritance relationships.
- What is a "complex" object?
- It is always possible to find a primary key for *any* relation. Why?
- Why are tables not relations?

### 5.2. Multiple-choice questions

- What relationship does a particular, individual book stand in with respect to libraries?
  1. one-one
  2. many-one
  3. one-many
  4. many-many
- What relationship does a book title stand in with respect to libraries?
  1. one-one
  2. many-one

3. one-many
4. many-many

Choose the best or most appropriate answer(s) to the question.

---

## 6. Bibliography

- Atkinson, M., Bancilhon, F., DeWitt, D., Dittrich, K., Maier, D., and Zdonik, S. (1989). The Object-Oriented Database System Manifesto. In *Proceedings DOOD '89*, Kyoto, December.
  - Blakeley, J.A. (1991) DARPA Open Object-Oriented Database Preliminary Module Specification: Object Query Module. Texas Instruments, Inc., Version 3, Nov. 25.
  - Booch, G. (1994) *Object-Oriented Analysis and Design with Applications, 2nd ed.* Benjamin/Cummings Publishing Company.
  - Chen, P.P. (1976) The Entity-Relationship Model □ Toward a Unified View of Data. *ACM TODS*, 1:1.
  - Codd, E.F. (1970) A Relational Model of Data for Large Shared Data Banks. *Comm. ACM* 13:6.
  - Codd, E.F. (1971) ALPHA: A Data Base Sublanguage Founded on the Relational Calculus. In *Proc. 1971 ACM SIGFIDET Workshop* (San Diego, Nov. pp. 11-12).
  - Codd, E.F. (1988) Fatal Flaws in SQL (Both the IBM and the ANSI Versions). *Datamation*, August and September.
  - Codd, E.F. (1990) *The Relational Model for Database Management, version 2.* Addison-Wesley Publishing Company, New York.
  - Cusack, E. (1991) Inheritance in Object-Oriented Z. In Pierre America (ed.), *Lecture Notes in Computer Science, ECOOP '91, European Conference on Object-Oriented Programming*, Springer-Verlag.
  - Date, C.J. (1987) Where SQL Falls Short. (abridged) *Datamation*, May 1. Unabridged version, What is Wrong with SQL, available from The Relational Institute, San Jose.
  - Meyer, B. (1997) *Object-Oriented Software Construction, 2nd ed.* Prentice Hall.
  - Olle, T.W. (1978) *The Codd Approach to Data Base Management.* John Wiley & Sons, New York.
  - Rao, B.R. (1994) *Object-Oriented Databases: Technology, Applications, and Products.* McGraw-Hill, Inc., New York.
  - Rumbaugh, J., Blaha M., Premerlani, W., Eddy, F., and Lorenzen, W. (1991) *Object-Oriented Modeling and Design.* Prentice Hall, New Jersey.
  - Simmel, S.S. and Godard, I. (1991) The KALA BASKET: A Semantic Primitive Unifying Object Transactions, Access Control, Versions and Configurations. In *Proceedings of OOPSLA '91: Conference on Object-Oriented Programming Systems, Languages and Applications*, Nov., pp. 230-246.
  - Stroustrup, B. (1997) *The C++ Programming Language.* Addison-Wesley Publishing Company.
  - Ullman, J.D. (1988) *Principles of Database and Knowledge-Base Systems, volumes I and II.* Computer Science Press, Inc.
- 

## 7. Reference Materials

### 7.1. Print References

Bancilhon, F., Delobel, C., and Kanellakis, P. (1992) *Building an Object-Oriented Database System: The Story of O2.* Morgan Kaufman Publishers. O2 is an important and powerful object-oriented database management system. This book provides many interesting insights into O2 in particular and into one approach to OODBMSs in general.

Booch, G. (1994) *Object-Oriented Analysis and Design with Applications, 2nd ed.* Benjamin/Cummings

Publishing Company. There are many excellent titles covering object-oriented design and analysis; this is just one of them. I believe it provides an outstanding introduction to the topic for both seasoned programmers and object newcomers alike.

Chen, P. (1977) *The Entity-Relationship Approach to Logical Data Base Design*. QED Publishing Co. This is a good introductory book by the man who is generally attributed as having developing the E-R modeling approach. It is out of print and, therefore, can be hard to find.

Codd, E.F. (1990) *The Relational Model for Database Management, version 2*. Addison-Wesley Publishing Co., New York. This work is not an introductory text and might not be suited for beginners. It is, however, a must for anyone who is seriously interested in the relational model either from an implementation or from a theoretical point of view.

Date, C.J. (1994) *An Introduction to Database Systems*. Addison-Wesley Publishing Co., New York. A comprehensive and approachable classic that covers a broad spectrum of database issues. Date covers both relational and object-oriented databases with material that is suitable for experts and beginners alike. This book covers the essential core of the relational model with special attention to practical implementation issues, however, it is not specifically aimed at designing databases.

Hernandez, M.J. (1997) *Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design*. Addison-Wesley Publishing Co., New York. This book is aimed, for the most part, at beginning relational database designers. It presents a platform independant approach while avoiding lots of jargon and theory that is irrelevant in the early stages.

Kim, W. (1990) *Introduction to Object-Oriented Databases*. MIT Press. Dr. Kim is a widely respected researcher and developer of object-oriented theory and systems. He is a pioneer in that part of database theory which is trying to bridge the gap between the object-oriented world and the relational world.

Maier, D. (1983) *Theory of Relational Databases*. Computer Science Press. For those interested in the formal underpinnings of the relational database model, this is the book.

Meyer, B. (1997) *Object-Oriented Software Construction, 2nd ed.* Prentice Hall. The Eiffel programming language is notable for its small size, expressive completeness, simplicity, elegance, and practicality. This book presents the Eiffel language as a vehicle with which to discuss software engineering.

Stroustrup, B. (1997) *The C++ Programming Language, 3rd ed.* Addison-Wesley Publishing Company. The C++ programming language is almost ubiquitous and this book is the C++ Bible. Bjarne Stroustrup created C++ and this book represents the latest in a long line of texts he has written on the subject.

Ullman, J.D. (1988) *Principles of Database and Knowledge-Base Systems, volumes I and II*. Computer Science Press, Inc. If I were constrained to own only one title about database systems, these two volumes would be it. They cover every aspect of the topic from the theory to the implementation along with the connections from databases to knowledge-bases and the artificial intelligence community.

---

## Evaluation

We are very interested in your comments and suggestions for improving this material. Please follow the link above to the evaluation form if you would like to contribute in this manner to this evolving project..

## Citation

To reference this material use the appropriate variation of the following format:

Meyer, Thomas H. (1997) Non-spatial Database Models, *NCGIA Core Curriculum in GIScience*,  
<http://www.ncgia.ucsb.edu/giscc/units/u045/u045.html>, posted November 10, 1997.

---

The correct URL for this page is: [http://www.ncgia.ucsb.edu/giscc/units/u045/u045\\_f.html](http://www.ncgia.ucsb.edu/giscc/units/u045/u045_f.html).

Last revised: November 19, 1997.

---

Gateway to the Core Curriculum

---



# techdictionary.com

Ads by Google

**Sharp Solar Experience**  
Over 50 years history of research & development in solar technology  
sharp-solar.com

- Home
- Search
- Search Tips
- Domains
- Emoticons
- Chat - IM
- File Extensions
- ASCII/HEX
- Security
- Job Search - *NEW!*
- Games *HOT!*
- Free Music - *COOL!*
- Tech Stores - *NEW!*
- Tech & Virus News
- Suggest A Term
- OS X Widget
- Trivia/Links
- Add To Your Site
- About TD/Site Map

## TechDictionary™ Search Results

Term:	Definition:
Entity-Relationship modeling	A Discipline for examining and representing the components and interrelationships in a database system. Also known as E-R modeling, this discipline factors a database system into entities, attributes, and relationships.

**SEARCH AGAIN**

Ads by Google

- Computer Terms
- Speak Chat
- Computer Games
- Chat Emoticons
- Text Chat

**CREATE YOUR MINI ME!**




**Zwinky**

**START NOW IT'S FREE!**

**Find A New Job**

Job Nurse, IT, Sales

Where: City, State

Search

careerbuilder.com



- ADVERTISE ON TD
- Take Yellow.com
  - Technical Translation
  - Website Translation

**The new PC Mall**

Computers and Electronics for Home and Business

**PC Mall**

[CLICK HERE](#)

**Open Box Outlet**

**PC Mall** [Click Here](#)

**iPod Nano** comes in 7 colors

**T&T** GET MORE. PAY LESS.

**WEB HOSTING**

FROM \$2.99/mo

**Buy.com**

**\$10 off \$200**

**FREE SHIPPING**

**freecycle**

Code Amber Alerts Powered by GTX Corp -- Presented by Code Art

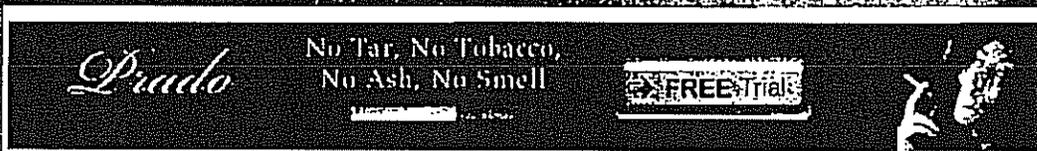
We created smart processors that deliver extra performance when you need it.



Smarter at Tomorrow



RS889



My Home

TOPICS

People

Companies

Jobs

White Paper Library

Search

JOIN NOW

SIGN IN

Toolbox for IT &gt; Topics &gt; Wiki

[Edit](#) > [Discuss](#) > [History](#) > [Invite Peers](#) > [Connect \(355\)](#)

 Parent Categories: [Entity Modeling](#) | [Data Model](#) | [Database](#) | [Computer](#) | [Software Documentation](#)

 Major Editors: [phill runciman](#) [stevetuf](#) [akshaya bhalla](#)
[All Editors](#)

## Entity Relationship Diagram

Updated Mar 2 2010 17:01 (101271 views)

### Entity Relationship Diagram (ERD) [\[edit\]](#)

An Entity Relationship Diagram (ERD) is a snapshot of data structures. ERDs show entities in a database and relationships between tables within that database. It is essential to have one of these if you want to create a good database design. The patterns help focus on how the database actually works with all of the interactions and data flows, although another useful tool is a Data Flow Diagram (DFD) which more directly describes this.

### Entity Relation Model [\[edit\]](#)

We use an Entity Relation Model (ERM) to create a data model of a system and its requirements in a top-down approach. This is frequently the approach utilized in database design. The diagrams which emerge from this methodology are called ER diagrams.

Structured data is represented by an Entity Relationship Model (ERM). Entity-Relationship Modeling is the model-generating process. The end-product of the modeling process is an entity-relationship diagram (ERD) or ER diagram, a type of Conceptual Data Model or Semantic Data Model.

### History [\[edit\]](#)

ER Diagrams were first introduced by Charles Bachman. "Bachman Diagrams" described data structures, however he did not go further and recognised the need to model at a higher level of abstraction.

\* . . . entity to mean a particular object being considered, the term entity class will mean an entire group of entities which are sufficiently similar, in terms of attributes that describe them, to be considered collectively. . . . entity set . . . associates a group of entities in one entity class with one entity of a different entity class in a subordinate relationship" page 4 of "Bachman, C. W. (1969) Data Structure Diagrams. DATA BASE 1(2) 4-10"

See a fuller discussion of the whole topic of Bachman and Chen diagramming in "The Entity Relationship Model And Practical Data Modelling By Steve Hitchman 2004, Journal of Conceptual Modeling, April 2004"

ER diagrams were popularised by Dr. Pin-Shan (Peter) Chen (朱敏生) in 1976, but he drew on the previous work of his colleagues, one of whom was Charles Bachman.

Mention should also be made of J. Barrie Leigh. "Mr. Leigh began his interest in E-R modeling techniques as a systems engineer for IBM in the UK, developing his first E-R diagram in 1971 for an annuity system of Royal Insurance. He was a recognized leader of transactional and database systems design for IBM in the insurance industry. He evangelized the use of distributed intelligence and distributed database architectures throughout the 1970's." See meeting notice for DAMA Philadelphia/Delaware Valley 10th January 2007. While at CACI, London he worked with a number of key professionals including Ian Palmer, Richard Barker and Keith Short all of whom made significant contributions to this field.

\*Note\*

In database context, an ERD facilitates visualizing relationship amongst tables by establishing linkages through a key in particular table pointing to specific records in the related tables.


[New to Toolbox?](#)
[Ask a Question](#)
[Join](#)

A Community of 1.7 Million Professionals



Lotus knows

#### Popular Articles In Computer

- > Basics of Excel VBA: Cell Navigation
- > Sharepoint Interview questions
- > How do I escape single quotes in SQL queries?
- > How do I update multiple fields with single update statement using subqueries?
- > Interview Questions with Answers for Oracle, DBA, and developer candidates

RSS90

Conceptually while defining ERD in context of data modeling, its application is not restricted to the traditional database design process only. For instance, it can also be applied as a process design tool where each process can be defined as an entity and relationship to another process can be established based on the attributes associated to each individual entity.

Also, Object oriented design and analysis can also be helped through an ERD, where each object can be defined as an entity along with respective attributes. An ERD can also facilitate codification of messaging by use of an object-oriented programming (OOP) language for interaction of objects in a system. In object oriented design context, an entity can be visualized as an individual object or a group of similar objects instantiated from a class of objects (also referred to as instances of an object)

JOIN NOW

### Related White Papers and Webcasts

- Unified Data Management: A Collaboration of Data Disciplines and Business Strategies
  - Data Quality Strategy: A Step-by-Step Approach
  - Master Data Management: Extracting Value from Your Most Important Intangible Asset
- [Show more White Papers](#)

SIGN IN

### Related Content

- Data Flow Diagrams (DFDs) *(Blogs)*
- ER\_Diagrams *(Wiki)*
- Data Modeling Software Vendors *(Groups)*

Disclaimer: IT Wiki is a service that allows content to be created and edited by anyone in the community. Content posted to this site is not reviewed for correctness and is not supported by Toolbox.com or any of its partners. If you feel a wiki article is inappropriate, you can either correct it by clicking "Edit" above or click [here](#) to notify Toolbox.com.

### Sponsored Links

- Innowera Process Runner - SAP and Excel Made Easy - Join a web demo and win an Apple iPad**  
Process Runner is the ultimate SAP Up/Download tool that requires no coding. Free Trial at [www.innowera.com](http://www.innowera.com)
- Quickly Create HRIS Operational Reports for PeopleSoft/Oracle Applications**  
Easily write simple and complex queries, distribute them securely and run them efficiently with [Rwiz](#).
- Big Security for Small Businesses from ZyXEL**  
Secure, Affordable and Green Firewall VPN & UTM Devices. Save up to 80% in energy cost.
- Have a question?**  
[Ask your question at Toolbox.com IT Groups](#)
- Buy a link now**





My Home

Training

People

Companies

Jobs

White Paper Library

Search

JOIN NOW  
SIGN IN

Toolbox for IT Topics - Wiki

Edit Discuss History Invite Peers Connect (355)

Parent Categories: Entity Modeling | Data Model | Database | Computer | Software Documentation

Major Editors: phil runciman stoveluf akshaya bhalia

All Editors

### Entity Relationship Diagram

updated Mar 2 2010 7:06 am | 101,878 views

#### Entity Relationship Diagram (ERD) [edit]

An Entity Relationship Diagram (ERD) is a snapshot of data structures. ERDs show entities in a database and relationships between tables within that database. It is essential to have one of these if you want to create a good database design. The patterns help focus on how the database actually works with all of the interactions and data flows, although another useful tool is a Data Flow Diagram (DFD) which more directly describes this.

#### Entity Relation Model [edit]

We use an Entity Relation Model (ERM) to create a data model of a system and its requirements in a top-down approach. This is frequently the approach utilized in database design. The diagrams which emerge from this methodology are called ER diagrams.

Structured data is represented by an Entity Relationship Model (ERM). Entity-Relationship Modeling is the model-generating process. The end-product of the modeling process is an entity-relationship diagram (ERD) or ER diagram, a type of Conceptual Data Model or Semantic Data Model.

#### History [edit]

ER Diagrams were first introduced by Charles Bachman. "Bachman Diagrams" described data structures, however, he did however go further and recognised the need to model at a higher level of abstraction.

entity to mean a particular object being considered; the term entity class will mean an entire group of entities which are sufficiently similar, in terms of attributes that describe them, to be considered collectively. entity set associates a group of entities in one entity class with one entity of a different entity class in a subordinate relationship" page 4 of "Bachman, C. W. (1969) Data Structure Diagrams. DATA BASE 1(2): 4-10"

See a fuller discussion of the whole topic of Bachman and Chen diagramming in "The Entity Relationship Model And Practical Data Modelling By Steve Hitchman 2004, Journal of Conceptual Modeling, April 2004"

ER diagrams were popularised by Dr. Pin-Shan (Peter) Chen (陈品山) in 1976, but he drew on the previous work of his colleagues, one of whom was Charles Bachman.

Mention should also be made of J. Barrie Leigh "Mr. Leigh began his interest in E-R modeling techniques as a systems engineer for IBM in the UK, developing his first E-R diagram in 1971 for an annuity system of Royal Insurance. He was a recognized leader of transactional and database systems design for IBM, in the insurance industry. He evangelized the use of distributed intelligence and distributed database architectures throughout the 1970's." See meeting notice for DAMA Philadelphia/Delaware Valley 10th January, 2007. While at CACI, London he worked with a number of key professionals including Ian Palmer, Richard Barker and Keith Short all of whom made significant contributions to this field.

\*Note \*

In database context, an ERD facilitates visualizing relationship amongst tables by establishing linkages through a key in particular table pointing to specific records in the related tables.



New to Toolbox?

Ask a Question

Join

A Community of 1.7 Million Professionals



GET STARTED

Lotus knows collaboration works better in the cloud.

Get complete cloud-based email and collaboration for just \$10/user/month.

TRY OUR OFFER

#### Popular Articles in Computer

- Basics of Excel VBA: Cell Navigation
- Sharepoint Interview questions
- How do I escape single quotes in SQL queries?
- How do I update multiple fields with single update statement using subqueries?
- Interview Questions with Answers for Oracle, DBA, and developer candidates

RS892

Conceptually while defining ERD in context of data modeling, its application is not restricted to the traditional database design process only. For instance, it can also be applied as a process design tool where each process can be defined as an entity and relationship to another process can be established based on the attributes associated to each individual entity.

Also, Object oriented design and analysis can also be helped through an ERD, where each object can be defined as an entity along with respective attributes. An ERD can also facilitate codification of messaging by use of an object-oriented programming (OOP) language for interaction of objects in a system. In object oriented design context, an entity can be visualized as an individual object or a group of similar objects instantiated from a class of objects (also referred to as instances of an object)

JOIN NOW

#### Jobs by indeed

- expressor Community Manager - Burlington MA
- Senior Solution Architect - Network Architecture - Framingham MA
- Senior IT Security Architect - Framingham MA

Search more jobs...

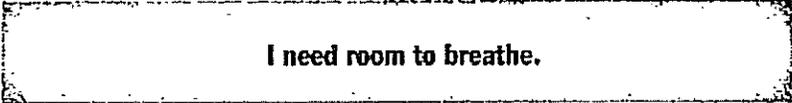
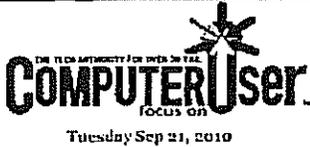
SIGN IN

Disclaimer: IT Wiki is a service that allows content to be created and edited by anyone in the community. Content posted to this site is not reviewed for correctness and is not supported by Toolbox.com or any of its partners. If you feel a wiki article is inappropriate, you can either correct it by clicking "Edit" above or click [here](#) to notify Toolbox.com

#### Sponsored Links

- > Big Security for Small Businesses from ZyxEL  
Secure, Affordable and Green Firewall-VPN & UTM devices. Save up to 60% on energy cost
- > Quickly Create HRIS Operational Reports for PeopleSoft/Oracle Applications  
Easily write, compile and run the queries, distribute them securely and run them efficiently with PeopleSoft
- Innovera Process Runner - SAP and Excel Made Easy - Join a web demo and win an Apple iPad  
ProcessRunner is the ultimate SAP Upload tool that requires no coding. Free Trial at [www.innovera.com](http://www.innovera.com)
- > Have a question?  
Ask your peers at Toolbox for IT Groups
- > Buy a link now





Home Articles User TV Press Release Education Dictionary SMDZone Resources Forum Blogs Classifieds Links Directory Green SEO Apps/Review

Ads by Google
Entity Relationship Model
Download Now!
Windows, Linux, OSX
Diagram Databases
and ER Models
www.aunafold.com

Become A Personal Trainer
Get ISSA Certified At Your Own Pace Turn Your Passion Into Your Career!
www.ISSAOnline.com/Person

Instructional Designer
Find Flexible & Accredited Education Schools Near You!
FindTheRightSchool.com/Edu

Learn Med. Coding Online
Take Online Medical Coding Cert Programs. Get Free Info!
www.elearners.com/Medical

Top Technology Degrees
Earn Your Certificate in Technology Online. Accredited Schools Only.
EducationDegreeSource.com

- CATEGORIES LIST
Y2K Terms
Emoticons
Chat Stuff
HTML Tags
File Types
Domains
Virus Terms
Communications
Electronics
Hardware
Networking Security
Software
Programming
Opensource
Database

WIT'S ONLINE
We have 664 guests online

Search for Dictionary terms (regular expression allowed)
search...
Begins with Contains Exact term Sounds like
All | 0-9 | & | ( | - | . | @ | | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | -

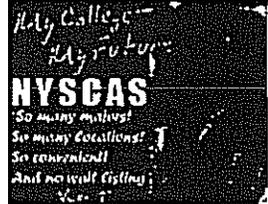
All
Word Explanation
ER model (Entity Relationship Model) ER model is a conceptual data model that views the real world as entities and relationships. A basic component of the model is the Entity-Relationship diagram which is used to visually represents data objects. Dr. Peter Chen's original paper on the Entity-Relationship model (ER model) is one of the most cited papers in the computer software field.

- Categories List
1) Y2K Terms 2) Emoticons 3) Chat Stuff
4) HTML Tags 5) File Types 6) Domains
7) Virus Terms 8) Communications 9) Electronics
10) Hardware 11) Networking/Security 12) Software
13) Programming 14) Opensource 15) Database
16) Internet 17) Wireless 18) Mobile Computing
19) Graphics 20) Multimedia 21) Gadgets
22) Others 23) VOIP 24) Adobe
25) Apple Mac 26) Autodesk 27) Cisco
28) Citrix 29) Google 30) IBM
31) Microsoft 32) Oracle 33) PHP
34) SUN 35) Imae

Disclaimer: This section is for information purposes only posted by the public. If you feel the information is incorrect, please send us an email to webmaster@computeruser dot com.

Explore Content
Entity Relationship Technology Reviews Electronics Reviews
Entity-Relationship Model Technology Videos Er Model

Online Certification Flexible Certificate Programs To Fit Your Schedule. Start Today. Phoenix.edu
Networking Certification MCP, A+, and Network+ Certification Local 12 Month Program. Start Now! www.PorterChester.com/Computer.html
Common Data Model Provide a Common Data Model for SOA-Based Applications. web.progress.com/dataxtend
Become A Personal Trainer



Sign Up for Our Newsletter!
Name
E-mail
Subscribe

Common Data Model Provide a Common Data Model for SOA-Based Applications. web.progress.com/dataxtend

- Ads by Google
FREE antivirus download
IT Support London
Freeware
HP Ink Cartridges
SQL training courses
Classifieds fr
Online payday loans by Advantecum.net
spy software
Add URL Directory - add links & articles
tradeshaw giveaways

I Want.



PORSCHE

I need a car I can commute in.



Seats four of the most comfortable and utterly speechless passengers in its class.



PORSCHE



PORSCHE

TODAY'S TECH WORD

- **.CAB**  
Microsoft compressed format
- **ACD Canvas**  
ACD Canvas is a graphics, publishing, and imaging product for personal computers. Canvas provides to Readmore..

FEATURED CATEGORIES

- Emoticons
- Chat Stuff
- File Types
- Domains
- Recent words

RECENT WORDS

- **Blu-ray** :  
Blu-ray is a DVD format for storing high-definition video. Bl...
- **Air Link Interface** :  
The network interface between a MIES and the CDPD service provi...
- **APCO 25** :  
Association of Public Safety Communications Officials A set o...
- **IXC** :  
InterExchange Carrier. Another name for a long distance telep...
- **MAE** :  
Metropolitan Area Exchange or Metropolitan Area Ethernet. A ma...

Want to Share?

Do you have an article or IT helpful tips you would want to include in our site? send it to [Contact us](#) and we'll review it for inclusion. Thank you.

RS895

[About us](#) | [Terms of use](#) | [Privacy Policy](#) | [Legal](#) | [Awards](#) | [Advertise](#) | [Writer guidelines](#) | [Sitemap](#) | [Contact](#) | [FAQs](#) | [Feedback](#) | [Link to us](#) | [Tell a friend](#)

Copyright © 1994-2010 ComputerUser, Inc., All Rights Reserved. "ComputerUser" is a registered trademark. Reproduction in whole or in part in any form or medium without express written permission of ComputerUser, Inc. is prohibited. ALL VIOLATORS/VIOLATIONS WILL BE PROSECUTED TO THE FULLEST EXTENT OF THE LAW.

Compliance: All banners and ads including downloads, text links, banners are paid advertisers. As a computer news site we encourage everybody to post their products and services for FREE. All the premium properties on the site are reserved for paid advertisers. If you have any questions and are interested in knowing more about our services, please contact webmaster at computeruser dot com.

Here are the topics we cover computer certification careers IT training games consulting data recovery data security digital entertainment emerging technology gadget reviews handheld hardware reviews home automation home networks home office how-to advice Internet Linux companies news local profiles articles blogs and press releases classifieds buy sell CU marketplace business channels smbcnre goodcause.

I want a break

Partner

sites: [www.computeruser.com](http://www.computeruser.com)  
[www.computeruser.com](http://www.computeruser.com)



**FALL 2010  
TRENDS  
ARE HERE!**  
THE RUNWAY LOOKS  
THAT ABSOLUTELY MUST  
FIND THEIR WAY INTO  
YOUR CLOSET...  
▶ FROM \$12

NEWPORTNEWS.COM

Recommended

College for your personal career path  
**FOR ALL COMPUTER TRAINING CLASSES**  
Self-paced Computer training programs  
**Choose your online Degree and more**  
Computer, Accounting, and Certificate in A+ MCSA,  
Microsoft Office  
**Professional, Trade, and Vocational  
Colleges and Training**  
All your IT training and technical degrees  
**Computer Training, IT Certification and  
Career Oriented Tech Schools.**  
A+ Certification, CISPE, MCSE, MCAS certification  
and Learn Info  
**Custom IT training for your company,  
online, offsite, onsite**  
Register for classes - IT for you courses 2009  
**Learn SharePoint from entry level to  
Professional**  
Learn Microsoft SQL Server courses here  
**Discover snap services to expand your  
Internet business**

RS896



ER

Related Searches

- Reichenbach falls
- Er cast
- Er tv show
- Er episode guide
- Tambo colorado
- Er spoilers
- Er episode summaries
- Er cast members
- Er nbc spoiler
- Er finale spoiler
- Er tv series
- Er original cast

Commonly Confused

- err
- Ur
- ur-

Nearby Words

- equus kiang
- equus quagga
- equus zebra zebra
- er
- er bentley
- er bloor
- er bukwer I lytton

Learn more about our Privacy Policy

At Dictionary.com, we respect your privacy. [Learn more.](#)

Did you know? It's one thing to read drugs, but can you think of the two words drug is short for?

ER - 31 dictionary results

New Cisco X2-10GB-ER  
 Fortune 50 Trusted - 3 Yr Warranty & In  
 Stock - Special \$2389.00 USD  
 AdvantageOptics.com

TeamHealth  
 Leader in emergency medicine and  
 hospitalist staffing and management  
 www.teamhealth.com

Collaborative Fusion, Inc  
 Leading provider of ESAR-VHP and incident management solutions.  
 www.CollaborativeFusion.com Sponsored Results

er 

{uh, er}  Show IPA

**-Interjection**  
(used to express or represent a pause, hesitation, uncertainty, etc.).

**-Can be confused:** er, err, ur-, Ur.

 Related Words for: ER  
 emergency room  
 View more related words »

 VISUALTHESAURUS

ER  
 Explore the Visual Thesaurus »

ED Documentation Systems  
 Electronic (EDIS) & paper charting for  
 emergency physicians & nurses  
 www.tsystem.com

Online Health Videos  
 Learn About the Early Indicators of  
 Arthritis & Other Conditions Today!  
 HealthNation.com/Arthritis Sponsored Results

ER

1. efficiency report.
2. emergency room.

Er

Symbol, Chemistry  
 erbium.

-er<sup>1</sup>

1. ...

Today's Word Picks on Dictionary.com

- inflammable
- little
- postprandial
- haberdash

object of their occupation or labor (*hatter; tiler; tinner; moonshiner*), or from their place of origin or abode (*Icelander; southerner; villager*), or designating either persons or things from some special characteristic or circumstance (*six-footer; three-master; teetotaler; fiver;*

2. *tenner* ).  
a suffix serving as the regular English formative of agent nouns, being attached to verbs of any origin (*bearer; creeper; employer; harvester; teacher; theorizer* ).

Compare *-ier<sup>1</sup>*, *-yer*.

**Origin:**

ME *-er* ( *e* ), a coalescence of OE *-ere* agentive suffix (c. OHG *-āri*, Goth *-areis* < Gmc *\*-arjaz* (> Slav *\*-ari*) < L *-ārius* *-ary*) and OE *-ware* forming nouns of ethnic or residential orig. (as *Rōmware* Romans), c. OHG *-āri* < Gmc *\*-warjaz* people

**-er<sup>2</sup>**

a noun suffix occurring in loanwords from French in the Middle English period, most often names of occupations (*archer; butcher; butler; carpenter; grocer; mariner; officer*), but also other nouns (*corner; danger; primer*). Some historical instances of this suffix, as in *banker* or *gardener*, where the base is a recognizable modern English word, are now indistinguishable from denominal formations with *-er<sup>1</sup>*, as *millor* or *potter*.

**Origin:**

ME < AF *-er*, equiv. to OF *-er*, *-ier* < L *-ārius*, *-ārium*. Compare *-ary*, *-eer*, *-ier<sup>2</sup>*

**-er<sup>3</sup>**

a termination of nouns denoting action or process: *dinner; rejoinder; remainder; trover*.

**Origin:**

< F, orig. inf. suffix *-er*, *-re*

**-er<sup>4</sup>**

a suffix regularly used in forming the comparative degree of adjectives: *harder; smaller*.

**Origin:**

ME *-er* ( *e* ), *-re*, OE *-ra*, *-re*; c. G *-er*

**-er<sup>5</sup>**

a suffix regularly used in forming the comparative degree of adverbs: *faster*.

**Origin:**

ME *-er* ( *e* ), *-re*, OE *-or*; c. OHG *-or*, G *-er*

**-er<sup>6</sup>**

a formal element appearing in verbs having frequentative meaning: *licker; flutter; shiver; shudder*.

**Origin:**

ME; OE *-r-*; c. G - ( *e* ) *r-*

**-er<sup>7</sup>**

a suffix that creates informal or jocular mutations of more neutral words, which are typically clipped to a single syllable if polysyllabic, before application of the suffix, and which sometimes undergo other phonetic alterations: *bed-sitter; footer; fresher; rigger*. Most words formed thus have been limited to English public-school and university slang; few, if any, have become current in North America, with the exception of *soccer*, which has also lost its earlier informal character.

Compare *-ers*.

**Origin:**

prob. modeled on nonagentive uses of *-er<sup>1</sup>*; said to have first become current in University College, Oxford, 1975-80

## E.R.

1. East Riding (Yorkshire).
2. East River (New York City).
3. King Edward. *Origin:*  
< NL *Eduardus Rex*
4. Queen Elizabeth. *Origin:*  
< NL *Elizabeth Regina*
5. emergency room.

## emergency room

### -noun

a hospital area equipped and staffed for the prompt treatment of acute illness, trauma, or other Medical emergencies. *Abbreviation:* ER

Dictionary.com Unabridged  
Based on the Random House Dictionary © Random House, Inc. 2010  
Cite This Source

## World English Dictionary

er<sup>1</sup> (v, ɜː) /ɜː/

### -interj

a sound made when hesitating in speech

er<sup>2</sup>

- the Internet domain name for Eritrea

Er

- the chemical symbol for erbium

ER

- abbreviation for

1. (in the US) Emergency Room (in hospitals)
2. Elizabeth Regina

3. Eduardus Rex  
[Latin: Queen Elizabeth]

Collins English Dictionary - Complete & Unabridged 10th Edition  
2009 © William Collins Sons & Co. Ltd. 1979, 1986 © HarperCollins  
Publishers 1998, 2000, 2003, 2005, 2006, 2007, 2009  
Cite This Source

## Word Origin & History

-er

suffix used to make jocular or familiar formations from common or proper names (soccer being one), first attested 1860s, English schoolboy slang, "Introduced from Rugby School into Oxford University slang, orig. at University College, in Michaelmas Term, 1875" [OED, with unusual precision].

er

as a sound of hesitation or uncertainty, attested from mid-19c.

Online Etymology Dictionary © 2010 Douglas Harper  
Cite This Source

## Medical Dictionary

### emergency room definition

Function: *n*

: a hospital room or area staffed and equipped for the reception and treatment of persons with conditions (as illness or trauma) requiring immediate medical care

Er definition

Function: *symbol*  
erbium

**ER definition**  
Function: *abbreviation*  
emergency room

Merriam-Webster's Medical Dictionary, © 2007 Merriam-Webster, Inc.  
Cite This Source

**emergency room** *n.*  
The section of a health care facility intended to provide rapid treatment for victims of sudden illness or trauma.

**Er**  
The symbol for the element erbium .

**ER abbr.**  
endoplasmic reticulum

The American Heritage® Student's Medical Dictionary  
Copyright © 2007, 2001, 1996 by Houghton Mifflin Company. Published by Houghton Mifflin Company.  
Cite This Source

## Science Dictionary

**Er**  
The symbol for erbium .

**erbium**  (ĕr'bi-əm) **Pronunciation Key**

**Symbol** Er

A soft, silvery, metallic element of the lanthanide series. It is used as a neutron absorber in nuclear technology and in light amplification for fiber-optic telecommunications. Atomic number 68; atomic weight 167.26; melting point 1,497°C; boiling point 2,900°C; specific gravity 9.051; valence 3. See [Periodic Table](#) .

The American Heritage® Science Dictionary  
Copyright © 2007. Published by Houghton Mifflin. All rights reserved.  
Cite This Source

## Computing Dictionary

**ER definition**

Entity-Relationship

**er definition**

**networking**

The country\_code for Eritrea.  
(1999-D1-27)

The Free On-Line Dictionary of Computing, © Denis Howe 2010 <http://fodoc.org>  
Cite This Source

## Abbreviations & Acronyms

**Er**  
erbium

- ER**
1. emergency room
  2. endoplasmic reticulum
  3. Eritrea (international vehicle ID)

The American Heritage® Abbreviations Dictionary, Third Edition  
Copyright © 2006 by Houghton Mifflin Company.  
Published by Houghton Mifflin Company. All rights reserved.  
Cite This Source

## Encyclopedia

**Er.**

(Er), chemical element, rare-earth metal of the lanthanoid series of the periodic table. Erbium is a grayish silver element that also occurs as a series of pink compounds. It had limited commercial uses until the age of fibre-optic telecommunications, when it became an important constituent of the signal repeaters in long-distance telephone cables.

Learn more about **Er** with a free trial on [Britannica.com](#).

Encyclopedia Britannica. 2008. [encyclopedia.britannica.com](http://encyclopedia.britannica.com)  
Cite This Source

## Famous Quotations

**Er**

"This Nicholas was risen for to plisse,  
And thoughte..."

"Someday our grandchildren will look up at us and say, "..."

RS900

"God is subtle, but he is not malicious.  
[Raffiner...]"  
"When that the firste cok hath crowe, anon  
Up rist..."

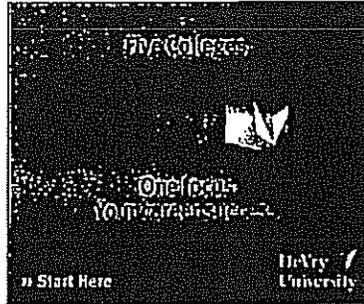
"At breakfast this Saturday morning, the Indian ... aske..."

More Quotes

Popular Subjects:

Friendship Funny Inspirational Life Love Proverbs

Search another word or see ER on Thesaurus | Reference



New Cisco X2-10GB-ER  
Fortune 50 Trusted - 3 Yr Warranty & In  
Stock - Special \$2389.00 USD  
AdvantageOptics.com

TeamHealth  
Leader in emergency medicine and  
hospitalist staffing and management  
www.teamhealth.com

Collaborative Fusion, Inc  
Leading provider of ESAR-VHP and Incident management solutions.  
www.CollaborativeFusion.com



ER

Search

## Data Modeling

# Entity Relationship Diagram

Entity Relationship Diagram is a specialized graphic that illustrates the interrelationships between entities in a database.

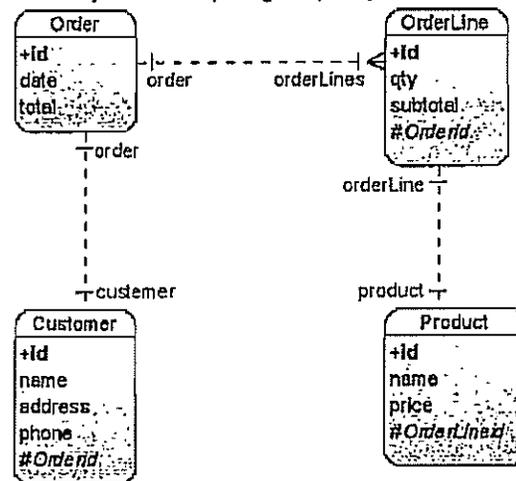
[ Entity Relationship Diagram | One-to-one Relationship ]  
 [ One-to-many Relationship | Many-to-many Relationship ]  
 [ Copy SQL Statements from Tables ]  
 [ Copy SQL Statements from Relationships ]  
 [ Synchronization to Class Diagram ]

[ Tutorial | Entity Relationship Diagram Demo ]

[Gallery Home](#)
[Visual Modeling](#)
[UML 2 Diagrams](#)
[Use Case Modeling](#)
[Requirements Capturing](#)
[Data Modeling](#)
[Object Relational Mapping](#)
[Documentation Generation](#)
[Code Engineering](#)
[Interoperability](#)
[User Interface](#)
[Cross-Platform](#)
[SDE for Visual Studio](#)
[SDE for Eclipse](#)
[SDE for NetBeans](#)
[SDE for JBuilder](#)

### Entity Relationship Diagram Sample

The Entity Relationship Diagram (ERD) illustrates the logical structure of the databases.



[Top](#)

[Home](#) | [Products](#) | [Shop](#) | [Download](#) | [Product Support](#) | [Resources](#) | [Company](#) | [Contact Us](#) | [Site Map](#)

Patents pending. All rights reserved.

[Legal Privacy Statement](#)

[Skip to Navigation](#)[Skip to Content](#)**OPPapers.com**

Research Papers and Essays for All

[Join](#)[Login](#)[Writing Service](#)[Upload](#)[Blog](#)[Follow Us](#)[Join](#)[Search 200,000 Essays](#) · [Search](#)**Get Better Grades Today By Joining OPPapers.com and Accessing Over 200,000 Articles and Essays!****Journal Entry Reversal Entity Relationship Diagram Explanation**

When developing a database design, there are many elements that are confusing and difficult to decipher. That is where an entity-relationship diagram (ERD) comes in. ERD's are used to describe the data requirements in a database system. They are used to describe the relationships between the entities (Entity Relationship, paragraph 1). Entities are anything which an organization needs to store data. A diagram is put into a picture for of the database design. Understanding a database is easier by looking at a diagram. This tool serves as a communication source between designers and users (Data Model, paragraph 2).

The Journal Entry Adjustment is the relationship of the entity. A database has more tables, which represent entities. Links among database tables represent the relationship with other tables. In this database, the relationship of a journal entry adjustment is with the entity "accounts to be adjusted."

The entity of this database is "accounts to be adjusted." Because; the entity is the characteristic of the database, a description of the entity. The accounts which need to be adjusted are the main source of a database table. The accounts needed to be adjusted will be related in other database tables. For example, to build a report of adjustments made for a period that could be found in different reports. Payroll and open accounts payable are examples of accounts that might need to be adjusted.

"Cardinalities are a notation showing the nature of a relationship among entities" (Moscove, Simkin, and Bagranoff, 2003, p.201). The relationship among accounts being adjusted is, one-to-many. Normally, the accounts being adjusted are the same ones for the end of an accounting period; one account, many times adjusted (1,N). The amount adjusted and date of adjustment, is the foreign key. The foreign key is a data field; combines information from tables to produce reports. In this case, when building a report with all the adjustment made for a certain accounting...

[Read Full Essay](#)[Already a Member? Login Now »](#)

This essay and over 200,000 other essays are available now on OPPapers.com.

Submitted by: magpiejones

Date Submitted: 10/10/2006 09:57 AM

Category: Accounting

Length: 3 pages (687 words)

Views: 1812

Rank: 20279

[Report this Essay](#)[Save Paper](#)**Related Essays**[Journal Entry Reversal...](#)[Journal Entry Of A...](#)[Journal Entry For A...](#)

RS903

Journal Entry  
Journal Entries Of "For...  
The Lottery: Examples Of...  
Journal Entry About My...  
Enders Game Journal Entries  
Catch-22 Journal Entry  
Journal Entry Of An Arab...  
Journal Entry Of A...  
Journal Entry  
Journal Entry Of A Subordinate  
Journal Entry Of A...  
Journal Entry On Eminem  
Journal Entry Of A Choctaw...  
Journal Entry  
Journal Entry  
Journal Entry African American  
Journal Entry: An Irish...  
Journal Entry Subordinate  
Journal Entry Of A...  
Journal Entry Of A...  
Journal Entry Of A...  
Raw - Journal Entries  
Journal Entry  
Assignment: Journal Entry...  
Journal Entry Of A...  
Journal Entry Of A...  
Journal Entry  
Read Full Essay  
Already a Member? [Login Now »](#)  
RSS ©2010 OPPapers.com

#### **Help**

[About Us](#)

[F.A.Q.](#)

[Contact Us](#)

#### **Site Stats**

[Top Donators](#)

[Top Colleges](#)

[Top Visitors](#)

[Popular Topics](#)

[Newest Members](#)

[Newest Papers](#)

#### **Legal**

[Terms of Service](#)

[Privacy Policy](#)

[Copyright Note](#)

[Sitemap](#)

#### **Saved Papers**

Save papers so you can find them more easily!

#### **Join Now**

Get instant access to over 200,000 papers.

[Join Now](#)

#### **Recent Topics**

[Holiday Essay](#)

[Identify The Five...](#)

RS904

Science & Technology  
Income Statement...  
Accountant...  
Types Of Electronic...

RS905



Level 1

- Foundations of Computing
- Computer Systems
- Models of Computation

Level 2

- Computation and Algorithms
- Professional Practice
- Software Engineering
- Interactive Systems and HCI
- Database Systems

- Introduction
- Database and File Based Systems
- Database Management System (DBMS)
- ANSI SPARC 3 Level Database Architecture
- Mappings and Data Independence
- Relational Database Principles and Terminology
- Relational Algebra, Calculus and Operators
- Database Design / Application Lifecycle
- Logical and Conceptual Data Models
- Entities and Entity-Relationship (ER) Modelling
- Relational Normalization / Normal Forms
- Enhanced ER Modelling (EER)
- Database Security Threats and Countermeasures
- Transaction Management and Concurrency Control
- Locks, 2PL and Deadlocks
- Software Analysis and Testing

Level 3

- Software Project Management
- Distributed Systems
- E-Business

Site Search



- Sponsored Links
- Ads by Google
  - Entity Diagram
  - Erd Design
  - Relationship
  - Data Modelling

Related Products

## ● Notes > Database Systems > Entities and Entity-Relationship (ER) Modelling

ER Modelling provides a fully scalable solution to modelling relationships between groups of data elements. These groups of data elements can be described as "entities" or "entity types". These entities are items (real or otherwise) that are of relevance to the business.

An ER model will contain the following concepts:

- Entity types
- Relationship types
- Attributes

An entity type describes a type of item which is distinctly identifiable. The identification of entities is typically open to the interpretative skills of the systems analyst or designer. There are no quick and easy rules which can be used to identify entities.

An ER model consists of an ER diagram or set of ER diagrams, as well as a set of normalised relations (tables) which correspond to the ER diagrams. Additional information that can be provided along with the ER model is as follows:

- written description of entities / relationships
- any assumptions
- additional constraints on the model

### Common Data Model

Provide a Common Data Model for SOA-Based Applications.



### BI White Paper

Learn how all users can access all their business intelligence.

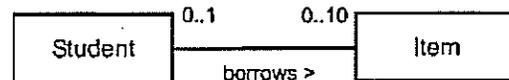
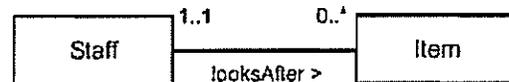
Ads by Google

### Entity-Relationship (ER) Diagrams

Entities are shown within a box. The entities "Student", "Book", "DVD" and "Staff" can therefore be represented as shown below:



Relationships between entities can be shown by joining the entities with a line:



As shown in the above diagram, the relationship can be named and given a direction (shown with a small arrow) to indicate which way the named relationship applies. This naming of relationships adds meaning and can reduce ambiguity.

Cardinality constraints show the number of instances of entities that can be involved in a relationship. For example, 1 member of staff can look after no books, 1 book, or many books. A book has to be assigned to some member of staff though. This is represented by the "1..1" cardinality constraint. One and only one member of staff must be assigned to each book.

### Relationship Degrees

The degree of a relationship is the number of entities that participate in that relationship. A simple relationship involving two entities is known as a binary relationship. Relationships

Less

10  
ore™  
sor  
y.

RS906



Business Information Systems

Prof Darrel Ince, ...

Best Price £0.12 or Buy New £30.54



Privacy Information

with three entities are known as ternary relationships. It is possible to have more entities involved in a relationship but this would lead to an increasing level of complexity.

Unary relationships involve a single entity which has a relationship with itself (i.e. the same entity type). Unary relationships are also known as recursive relationships.

See also Enhanced ER Modelling (EER)

**ETL Open Source Download**  
Fast, Powerful, Affordable, Easy Migration Tools. Free Download!



**Entity Relationship Model**  
Download Now! Windows, Linux, OSX Diagram Databases and ER Models

Ads by Google

Search for "Entities and Entity-Relationship (ER) Modelling" on: Google | Kelkoo | Amazon | eBay (UK) | eBay (US)

Search for "Entities and Entity-Relationship (ER) Modelling" on the rest of Computing Students: Entities and Entity-Relationship (ER) Modelling

Home | Contact | Shop | Notes | Questions | Programming | Links | Dictionary | Coursework | Tutors

Sponsored Links: Affiliate Program Articles | Computer Science Definitions | CS Degree Notes

Copyright © 2005-2009 ComputingStudents.com  
This site is to be used in accordance with the ComputingStudents.com User Agreement



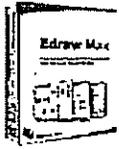
Birch Hill Technology Grp The Data Warehousing Experts. Struggling? We can help. [www.birchhillgroup.com](http://www.birchhillgroup.com)

Top Technology Degrees Earn Your Associates in Technology Online. Accredited Schools Only. [EducationDegreeSource.com](http://EducationDegreeSource.com)

Top 40 CRM Vendors Rated 2010 Top 40 CRM Software Rankings. Download Free CRM Research Report. [BusinessResearch.com](http://BusinessResearch.com)

Ads by Google

> Home > Knowledge Base



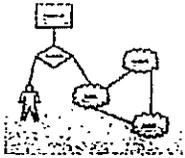
## Edraw Max

Include all functions and libraries of Edraw product serial



### Chen ERD Software

Draw entity relationship diagrams (ER diagrams) easily with Edraw!



Edraw is a very easy-to-use and intuitive database design tool and ER Diagram tool which can save you hundreds hours of work.

Edraw is not only an excellent tool for ER Diagram, but also the one that allows you to reverse engineer already existing database structures, create detailed HTML or PDF reports.

Free Download Chen ERD Software and View All Templates

#### Symbols for Chen ERD

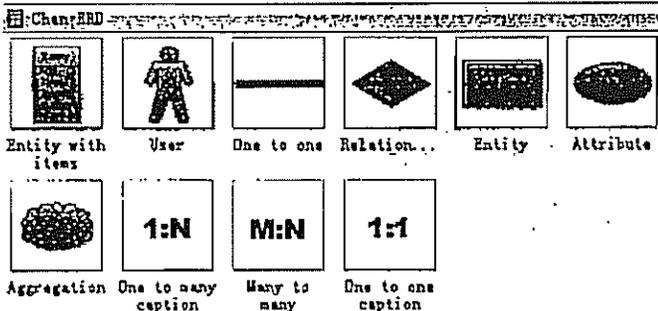
Thanks to an extensive set of library objects such as entities, links, items, attributes, users, types, captions, inheritance, references, boundaries, events, clouds etc Edraw is a perfect tool for database design and ERD diagramming.

You can also use Edraw to draw Chen ERD, Database model diagram, Express-G, Martin ERD, ORD Diagram and a lot more.

With Edraw you can visually design schemas for leading clients/servers and desktop databases. By graphically showing the relationships between Information tables stored in the database a database diagram helps to ensure that the database you are designing is accurate, complete and efficient. Edraw allows you to visually create Entity Relationship diagrams (ERD) for various database systems.

Moreover, data flow between tables can also be easily described by creating appropriate Data Flow Diagrams (DFD).

Cross-DBMS target compatibility allows you to design once and use for all.



#### Examples of Chen ERD Diagram

**EDRAW MAX**  
Latest Version: 5.4  
Free download  
The Best Choice for Diagramming

Don't lose your chance to save!



Time Limited Offer, Buy One Get One Free

**Order Now**

BOOKMARK |

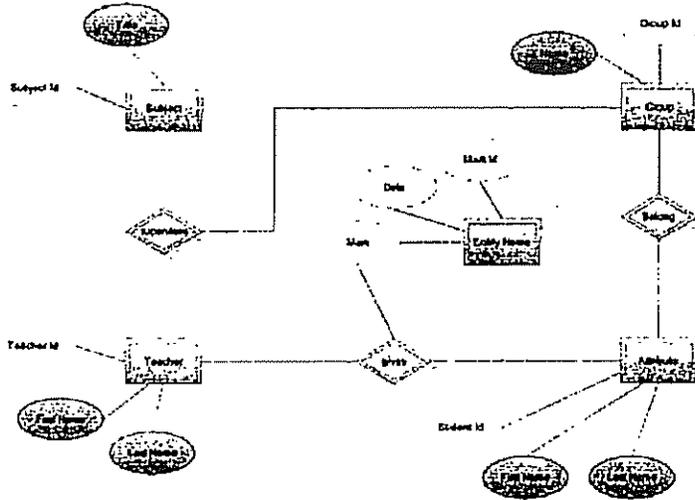
#### Easy to Create

- > Flowchart
- > Network Diagram
- > Organizational Chart
- > Business Forms
- > Mind Map
- > Business Diagram
- > Business Process
- > Clip Art
- > Database and ERD
- > Directional Map
- > Engineering Diagram
- > Fashion Design
- > Building Plan
- > Graphics and Charts
- > Software Diagram
- > UML Diagram
- > Web Diagram
- > Work Flow Diagram
- > Project Management Diagram

#### Good Reads

- > Create a Basic Flowchart
- > Create an Organizational Chart
- > Create a Network Diagram
- > Create Effective Diagrams for Network Documentation
- > Create a Brainstorming Diagram
- > Create a Floor Plan
- > Apply a Professional Look to Your Drawings with Themes
- > Add a Hyperlink Navigation Shape to Your Drawing

# Chen ERD



## Quick Links

- › Screenshots
- › Compare Products
- › What's New
- › Update Policy

## Peter Chen

Chen's original method is the basis for many writings on ERD's. While the traditional aspects of entities and relationships are represented as boxes and lines (respectively), there are a number of unique attributes to his present method:

- The cloud symbol represents aggregations of smaller ER diagrams, in the context of a logical design (as opposed to a physical design).
- The double-box represents a weak entity, which is dependent on its parent entity.
- A diamond symbol represents a type of relationship.
- Relationship adornments are added adjacent to the lines to indicate connectivity (I, M, N) and cardinality.
- The data-structure diagram (an alternate form of the ERD) uses arrows to show I:M relationships.
- Circles represent the attributes of an entity, although Visio labels this icon as value.
- A human figure represents a user icon.

 Chrome fast.

 Chrome fast.

[Flow Chart](#) | [Network Diagram](#) | [Organizational Chart](#) | [Privacy Policy](#) | [Link to Us](#) | [Contact](#) | [Site Map](#)

Copyright EdrawSoft 2004-2010; All Rights Reserved.



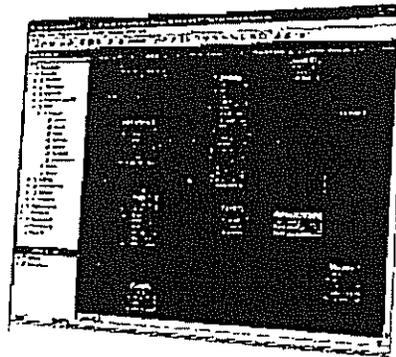
**Overview** [Learn](#) [Compare Editions](#) [What's New](#) [Databases](#) [Requirements](#)

## Robust and Easy to Use Database Design Tool

New version: V6 . Read what's new or upgrade.

DeZign for Databases is an intuitive database design tool for developers and DBA's that can help you model, create and maintain databases. The software uses entity relationship diagrams (ERDs) to graphically design databases and automatically generates the most popular SQL and desktop databases.

DeZign for Databases offers a sophisticated visual data modeling environment for database application development that makes your database development process much easier. The tool reduces faults in database development and improves your productivity. You can visualize database structures to understand your database, create new databases or reverse-engineer existing databases to modify, document, analyze, and optimize.



The tool is extremely easy to use. Whether you are a beginner or an expert database modeler, you will find your way in the tool very easily. DeZign for Databases provides all the features you expect in a professional database modeling and design tool.

[> Download trial](#) [> Purchase DeZign for Databases](#) [> Learn more](#) [> Take the screenshot tour](#)

"DeZign just keeps getting better. I've been really happy with the software - and that's a high compliment, because I am NOT easily pleased - I'm a senior database architect and have been in this arena for the last 15 years."

V. Newell, Senior Database Architect

## DeZign for Databases Key Features and Benefits

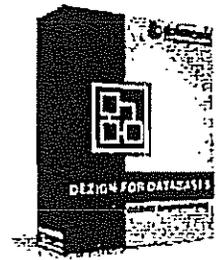
### Visual Database Modeling

DeZign for Databases offers a intuitive visual interface that allows you to accurately design your database. It uses entity relationship diagrams (ERDs) to graphically design databases and produces high quality database diagrams, enabling you to present your design at various levels of detail.

### Report Publishing Functions

Customizable data model reports with different levels of detail can be generated. Descriptions and other relevant information will be exported to HTML, MS Word or PDF.

### Better Database Designs



Latest version: 6.2.1  
Standard Edition: \$ 129  
Professional Edition: \$ 449  
Expert Edition: \$ 799

- [Buy Now](#)
- [Upgrade](#)
- [Download Trial](#)

### Products

- DeZign for Databases
- DB Data Diffective
- DB Schema Diffective
- DB Data Generator
- DB MultiRun

### DeZign for Databases supports the following databases:

- Oracle
- MS SQL Server
- MS Access
- MySQL
- PostgreSQL
- InterBase
- FireBird
- dBase
- Paradox
- DBISAM
- SQLite
- Pervasive
- IBM DB2
- Sybase
- Informix
- FoxPro

See DeZign for



## Database Development Lifecycle

**Generate databases:** Generate complete DDL scripts to create your database or generate your database directly.

**Import databases:** Derive a graphical data model from existing databases. Reverse engineer from the database directly or import from a SQL script.

**Database/model synchronization:** Our data modeling product offers bi-directional compare and synchronize functionality for all synchronization use cases: model-to-database, model-to-script, database-to-model, script-to-model, model-to-model.

### Team Work: Compare and Merge Models

When two or more people work on the same data model, DeZign for Databases V6 has a new compare and merge utility for projects. This utility allows developers to compare and merge two versions of the same data model.

> [Learn more...](#)

## Supported Databases

DeZign for Databases supports more than 15 databases including Oracle, MySQL, MS SQL Server, MS Access, DB2 and PostgreSQL. A complete list of supported databases can be found in the supported databases overview.

## Editions

DeZign for Databases' features vary by product edition. DeZign for Databases is available in three editions to accommodate your data modeling and database maintenance needs:

- DeZign for Databases Standard
- DeZign for Databases Professional
- DeZign for Databases Expert

Which DeZign for Databases edition is right for you or your organisation? Read it in the edition comparison chart.

You can validate a model for errors any time during the design process. During validation, DeZign for Databases checks to make sure the elements in your model are correct and complete. Reusable objects (domains and attribute packages) and name templates promote consistent database object definitions.

### Physical Models or Database-Independent Models

DeZign for Databases supports modeling for a specific target DBMS (physical) and it supports database-independent modeling with portable data types (logical).

### Easy-to-Use Modeling Environment

DeZign for Databases follows the industry standard modeling processes and has a well-designed user interface which decreases the user's learning curve.

## Data modeling: Entity relationship (E-R) vs. dimensional data models

**Which data modeling technique (entity relationship (E-R) data modeling or dimensional modeling) is better in which situation? Is there any problem associated with performance for either of them?**

Yes, whether to use data modeling is good for reporting and point queries while dimensional data modeling is good for ad-hoc query analysis. Many times, this translates to an entity relationship-based data warehouse and a dimensional data mart layer. Dimensional imposes some rules on the modeling, but results in a data model that has the access methods inherent by virtue of the relationships. Users are also better able to relate to the 'see measure by dimensional value(s)' paradigm than 'anything goes'.

Although, especially in shops that start small and grow into a robust architecture, the data warehouse itself may be dimensional. Dimensional data modeling is hard to come by directly from source, so in this approach dimensional is probably supported by some E-R, normalized tables. Also, there should be flexibility in the mart architecture such that marts are designed 'for purpose' - not all marts are for ad-hoc query analysis or generalized purposes and thus, not all marts should be dimensional.

I have reviewed hundreds of data warehouse data models. I have yet to find a textbook-perfect E-R (i.e., 3rd normal form) or dimensional model so don't fret the details. Pick a technique as a guideline and build your data models for purpose.

All Rights Reserved, Copyright 2000 - 2010, TechTarget | Read our Privacy Statement

# Web and XML Glossary

1 2 3 4 5 6 7 A B C D E F G  
H I J K L M N O P Q R S T U  
V W X Y Z



dret@delicious

## E

E.164 · E4X · EAI · EAN · EAP · EARL · EBCDIC · EBNF · EBX · ECC · ECDSA · ECMA · ECMAScript · ECML · E-Commerce · EDAP · EDD · EDGE · EDI · EDIFACT · EEPROM · EER · EFM · EFR · EGA · EGNOS · EGP · EIAJ · EIDE · EII · EJB · ELD · EMF · EML · EMMA · EMS · EMX · ENUM · EPOC · EPP · EPROM · EPS · ER · ERE · ERX · ERex · ESAX · ESIS · ESMTP · ESP · ETSI · EUC · EXI · EXIS · EXPRESS · EXSLT · Eiffel · Ethernet · Examplotron · ExifexXML · exFAT

Google

Web dret.net/glossary

## ER (Entity-Relationship Model)

### Type Associations

- Topic(s) from which this Topic is derived:
  - Modeling Language

### Associations

- ER is used as a base by
  - EER · ERX · HNER · XER

### Bibliographic References

- Peter Pin-Shan Chen: The Entity-Relationship Model — Toward a Unified View of Data [1]
- G. Di Battista, M. Lenzerini: Deductive Entity Relationship Modeling [0.9]
- Giuseppe Psaila: From XML DTDs to Entity-Relationship Schemas [0.7]

### Additional Information

- Topic Creation: 2003-07-12; HTML Creation: 2010-09-19, 07:00:18
- Comments? Corrections? Updates? Please send Email!

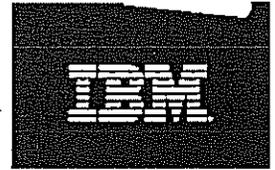
ABNF Ajax Android API architecture ASCII Atom AtomPub authentication AVS BibTeX BNF BOND1 brew browser ccd ccREL ccTLD CGI conference cookie CORBA CQL CSS CURIE DataRSS DITA DNS DocBook DOM DRM DTD EBNF eBook EC2 EEML EGM ETag EXI Exif facebook FastCGI fIQL Firefox flash FOAF GData GeoJSON georss GIS GML GODDAG GRDDL gTLD HTML HTML5 HTTP HTTPS I18N IANA iCalendar IMP Internet IOS iPad iPhone IRI ISBN J2ME Java JavaScript JAX-RS journal JSON JSOHP KML L10N LaTeX LBS license linkeddata LMNL LocWeb LOINC maps MD5 MediaRSS metadata MIME MQTT N3 networks NoSQL OASIS OAuth OData oRPC ontology oOXML OpenID OpenSearch OpenSocial ORE ORM OSGi Oslo OWL PDF PDF/A PhoneGap PubSub PubSubHubbub python QR race RAID RDBMS RDF RDFa RDFS RDQL regex RelaxNG REST RFC ROA RSS S3 SAX SCGI Schematron scovg SemWeb SEO ServiceScience SGML SHA SKOS SLAP SMS SOA SOAP SPARQL SPDY SPIN SQL SRU SSI SUP SVG TCP TexMECS ThruDB TLS TopicMaps TrIQ TrIX twitter UCD UDDI UI Unicode URI UTF-16 UTF-8 validator vCard void w3c WADL WAI WCAG webarchitecture WebDAV WOA

workshop WRAP WS-  
WSDL WURFL XBRL XCONCUR  
XDSHS XDM XForms XHTML  
XInclude XML XML-RPC xml:id  
XMPP XPath XPointer XProc  
XQuery XRDS XSD XSLT  
YQL YUI

1 2 3 4 5 6 7 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A technical discussion on modeling with UML  
06/11/03

**Rational** software



## Entity Relationship Modeling with UML

*Davor Gornik*

**Table of Contents**

---

<b>Entity Relationship Modeling .....</b>	<b>1</b>
<b>Core Elements of ER Modeling .....</b>	<b>1</b>
Entity Types .....	1
Attributes .....	2
Relationship Types .....	2
Attributes of Relationship Types .....	3
Simple Constraints in ER Modeling .....	4
Cardinality .....	4
Dependency .....	6
Specialization and Generalization .....	7
Specialization and Generalization .....	7
Categorization .....	8
<b>Notations for the ER Methodology .....</b>	<b>9</b>
Entity Types in UML .....	9
Attributes in UML .....	10
Relationship Types in UML .....	10
Attributes of Relationship Types in UML .....	11
Simple Constraints in UML .....	11
Cardinality .....	11
Dependency .....	12
Specialization and Generalization .....	12
Categorization .....	13
<b>Summary .....</b>	<b>14</b>

## ***Entity Relationship Modeling***

---

One of the most misinterpreted terms in the software industry is actually one we know very well: entity relationship (ER). That's because we often lack a common definition that is understood by all members of the development team. We assume that everyone on the team shares the same clear understanding of the methodology, syntax, and mechanics associated with ER and ER modeling.

ER modeling itself defines the methodology used in the analysis and design of information-based systems. Database designers often use this methodology to gather requirements and define the architecture of the database systems. The output of this methodology is a list of entity types, relationship types, and constraints.

Unfortunately, ER modeling does not define the graphic syntax for the representation of ER diagrams. Many times notations are used solely by the database team and limit the ER modeling to relational database design. We need a notation that allows broader understanding by members of the entire system development team.

The Unified Modeling Language (UML) is a widely accepted language used by analysts and software developers that is an excellent fit for the graphic representation of ER diagrams. By using UML, development teams gain significant benefits, including easier communication between team members, easy integration to repositories due to this language based on meta-models, use of a standardized input/output format (XMI), universal use for application and data modeling, unified representation from analysis to implementation to deployment, and completeness of specification.

This white paper defines the core concepts of ER modeling and explains how UML can be used by development teams to develop ER models.

## ***Core Elements of ER Modeling***

---

ER modeling is based on artifacts, which can be either a representation of physical artifacts, such as Product or Employee, or a representation of a transaction between artifacts, such as Order or Delivery. Each artifact contains information about itself. ER modeling also focuses on relationships between artifacts. These relationships can be either binary, connecting two artifacts, or ternary, among several artifacts.

The four essential elements of ER modeling are:

- Entity types
- Attributes
- Relationship types
- Attributes on relationships

### **Entity Types**

An entity type is a set of artifacts with the same structure and independent existence within the enterprise. Examples of an entity type would be Employees or Products.

A single occurrence of an artifact is an entity. While an entity type describes the structure, the entity itself identifies the single instance and all of the data of this instance. An example of an Employees entity would be the Employee Joe Ward.

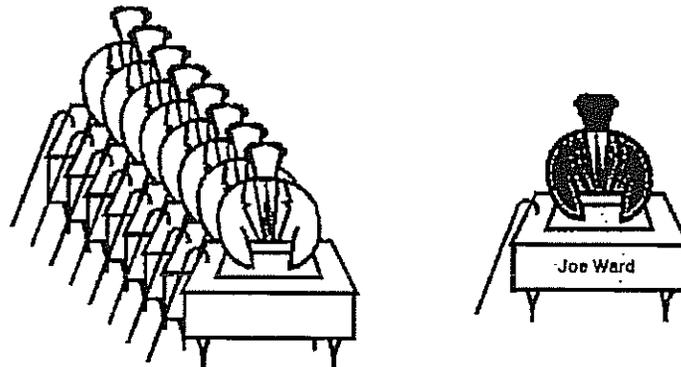


Figure 1 Entity Type Employees and Entity Employee Joe Ward

## Attributes

The structure of an entity type is defined with attributes. An attribute can be seen as a property of an entity type. Attributes of an Employee might be Name, Address, Social Security Number, Birth Date, Date Joined, and Position.

Entities differ from each other by the values of their attributes. Since it is possible to have entities with exactly the same values for attributes, we lose the ability to differentiate and address a specific entity. Therefore, we must always make sure that the values of attributes of a specific entity are unique to values of other entities. Each Employee has a unique combination of Name and Social Security Number attributes.

An example of associated values for attributes of an Employee is: Joe Ward, living at 34 Main Road, Redmond, WA, 98053, has the Social Security Number 555-32-2222, was born on September 7, 1971, and joined our company October 1, 2001, as a service engineer for consumer electronics.

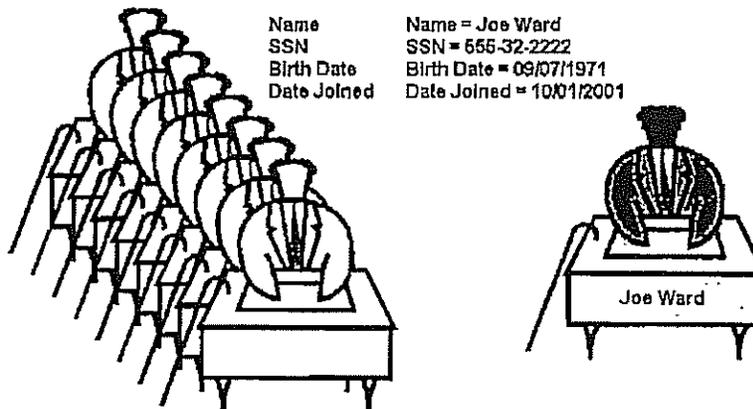


Figure 2 Attributes of the entity types Employees and the values for attributes for the entity Employee Joe Ward

## Relationship Types

While entity types describe independent artifacts, relationship types describe meaningful associations between entity types. To be precise, the relationship type describes that entities of entity types participating in the relationship can build a meaningful association. The actual occurrence of the association between entities is called a relationship.

It is important to understand that although we defined a relationship type, this does not mean that every pair of entities builds a relationship. A relationship type defines only that relationships can occur.

An example of a relationship type is the Employee owns Product. In this example the relationship type is Owns. The relationship itself is: The Employee Joe Ward owns a Product, the yellow Telephone, with the Serial Number 320TS03880.

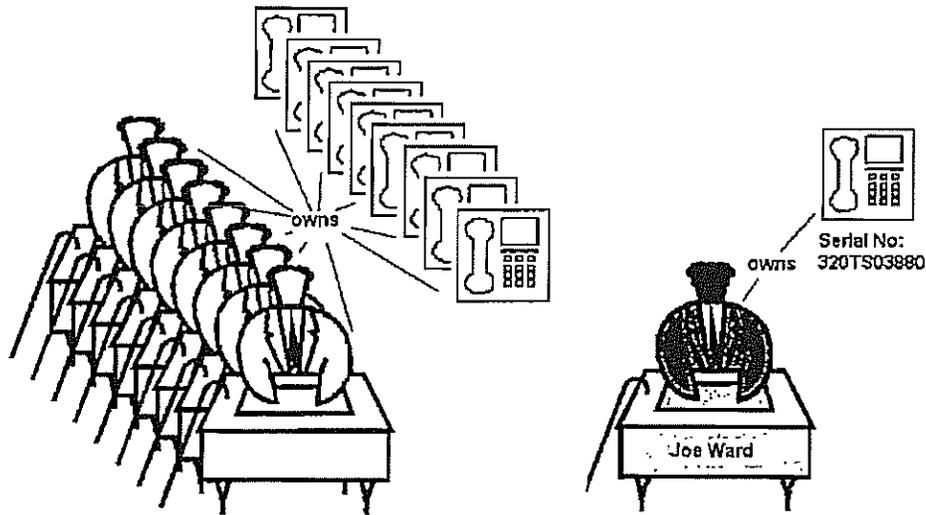


Figure 3 Relationship Type owns and Relationship owns between Employee Joe Ward and the Product with the serial number 320TS03880

There might be a second employee Martin Weber who does not own a Telephone.

### Attributes of Relationship Types

Relationship types can also contain attributes. For example the relationship type Services between the Employee and the Product could contain the attributes Date and Status, which identify the date of the service and the status of the product after the service is done.

When the relationship is realized in a concrete occurrence of the service, the values of the attributes of the relationship are set. The meaning of the relationship could be: Joe Ward services the black Toaster with the Serial Number 0462834DF4 on July 3, 2002, and establishes the status as good working condition.

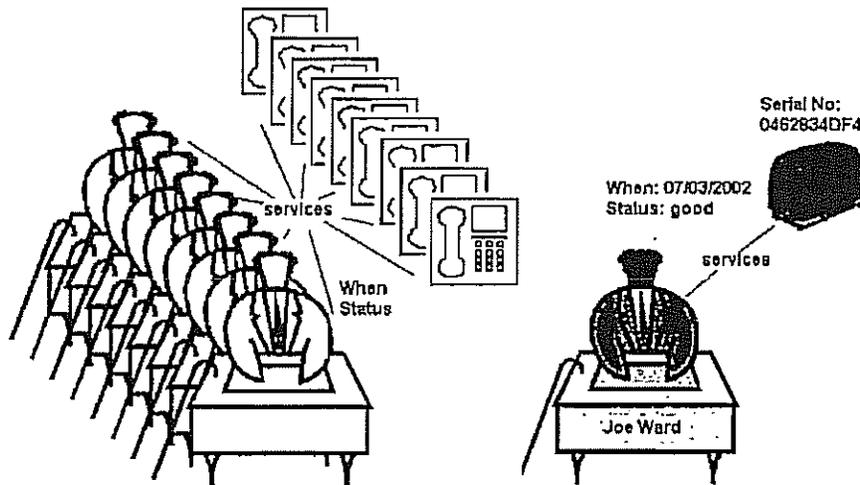


Figure 4 Attributes on Relationship Type services and Attributes on Relationship Joe Ward services Product with the Serial No 0462834DF4

## Simple Constraints in ER Modeling

Entities, relationships, and attributes within the ER model establish restrictions that define the structure of the enterprise. The structure is limited by rules called constraints. For example, it is not feasible that an Employee deals with more than 100 customers. Or, every employee must be associated with exactly one department.

### Cardinality

Each specified relationship type defines the possibility of establishing relationships between all of the participating entities. In most cases, this is not necessary. For example, not all Employees own all of the Products.

Relationships are bi-directional, connecting two entity types (Employees and Products) or the same entity type playing two different roles (Employees as a manager and Employees as a subordinate). Relationships can also be multi-directional, connecting more than two entity types. One example of a multi-directional relationship would be a phone call connecting an employee, a customer, and two phones. In either case, each entity type specifies the cardinality towards the relationship type.

The simplest cardinality is specified by the number of relationships allowed per entity. If only one Department participates in the relationship associated with an Employee, we write a 1 on the connector. This would mean that Joe Ward must be associated with one and only one Department.

Other possibilities for cardinality are either Not Specified or Specified By a Variable. The Not Specified cardinality is unlimited. The same is true for a cardinality Specified By a Variable (mostly M or N).

When the lower and upper limits of participating entities in a relationship are different, we specify a pair of values for the lower and higher limits enclosed in parenthesis and separated by a comma (M, N). An optional relationship would be recognized by (0, 1) or (0, N) dependent on the upper limit.

For example, a setup for Players on a Soccer team would be something like (1, 18). The entity Redmond Lions Soccer Team builds a relationship to the entity type Players, which consists of Joe Coplen, David Archer, John Good, Kevin Hale, Ivan Komashinsky, Steven Cooper, Andrew Bliven, Art Lounsbery, Chad Beery, Randall DuBois, Ron Baghai, Lance Delo, Tito Magobet, Curtis Hrischuk, and Ian Leslie.



There are several occasions when soccer players receive yellow or red cards for bad behavior or fouls. They are described in the Cards entity type. The Cards entity type builds a Received relationship type with Players with a cardinality of (0, N). This means that the Player David Archer can have a Received relationship to 3 Card entities, whereas Lance Delo does not have any.

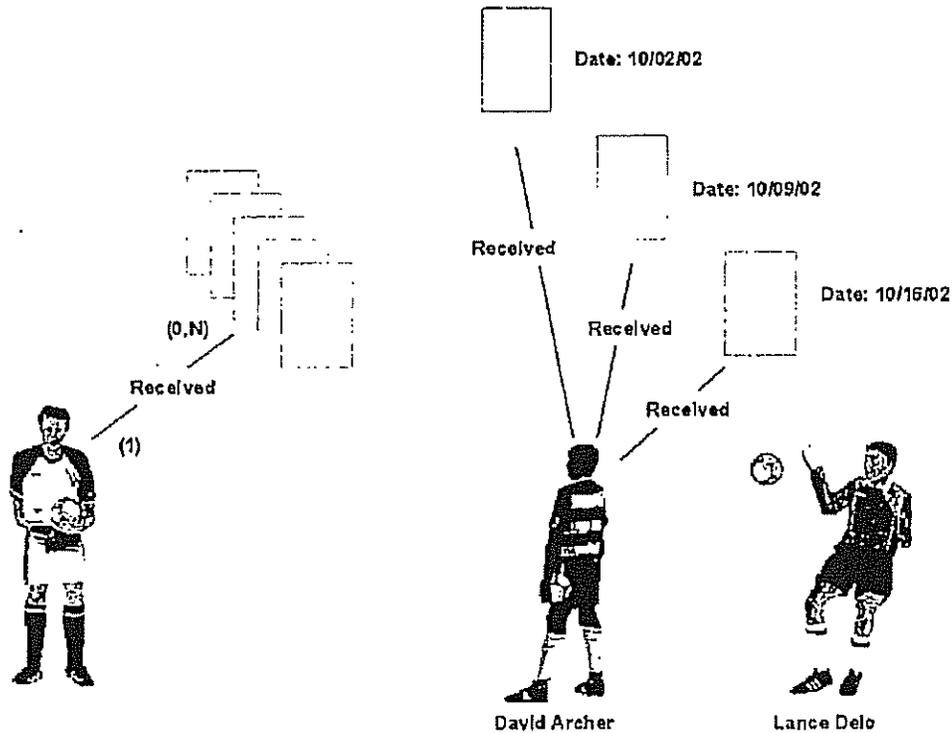


Figure 7 Entity type Players can Receive 0 or any number of entity type Cards: player David Archer received 3 cards, player Lance Delo did not receive any cards yet

### Dependency

Dependency refers to a situation in which a particular entity can not meaningfully exist without the existence of another entity as specified within a relationship. The entity type is dependent on another entity type when each entity of a dependent entity (subtype) depends on the existence of the corresponding parent entity in the super type.

A mandatory dependency relationship has to be specified by explicitly defining the lower limit for cardinality that is not equal to 0 or by a fixed cardinality that is not equal to 0.

An example of a dependency between two entities would be the entity type Marriage Certificate, which depends on the entity type Person. The relationship is Married with a fixed dependency on one Marriage Certificate and two Persons.

For example, the entity Marriage Certificate 352647003 has a fixed dependency to entities Joe Ward and Melinda Bell. This means that if either Melinda Bell or Joe Ward leaves the relationship, the Marriage Certificate 352647003 becomes invalid – at least from the data point of view.

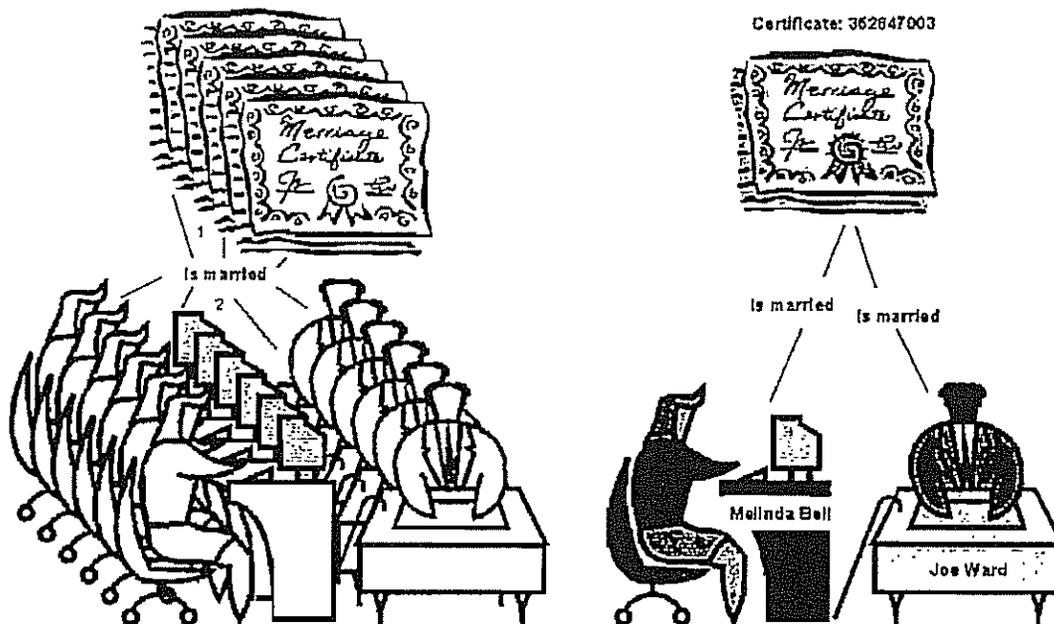


Figure 8 Dependent entity type Marriage Certificate on the entity type Persons and dependency of the entity Marriage Certificate 352647003 on Melinda Bell and Joe Ward

## Specialization and Generalization

The core ER model defines only basic relationships between entity types. While the basic entities and relationships can easily represent most of the simple data structures in business organizations, technical applications require more complex structures based on similarity and differences between entity types.

### *Specialization and Generalization*

The intent of specialization and generalization is reuse of the attributes and behaviors associated with entity types.

Specialization is used to define an entity type that represents a specific segment of a larger entity type. The specialized entity type inherits the structure and behavior, such as business rules, from the parent entity type. However, while the specialized entity type extends the parent structure or behavior, it is never less than the parent.

For example, Employees is a specialization of the entity type Persons, which requires all of the attributes and relationships of the entity type Persons. There may also be a second entity type called Customers that is a specialization of the entity type Persons. Both entity types have the same Persons attributes, which are seen as attributes of Employees or attributes of Customers. Therefore, when looking at the Customers, we see all of the attributes specified in the Persons entity type and specified in the Customers entity type.

Generalization is exactly the opposite workflow. The generalized entity type (or parent type) represents the common structure and behavior of all subtypes and contains all of the common attributes from child entity types. The child entity types have a complete view of the parent attributes and their own attributes.

The process of generalization finds the common structure and extracts it in the parent entity type. The parent entity type is commonly found during refactoring when comparing entity types and simplifying the model.

While specialization can be directly implemented only with object-oriented or object-relational databases, generalization can be directly implemented with any relational database using foreign keys.

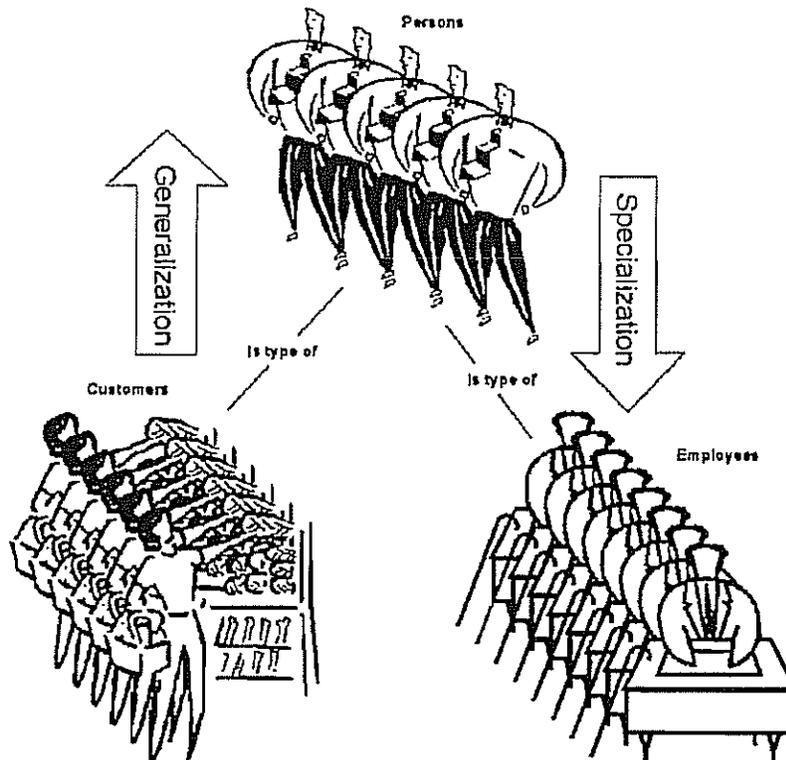


Figure 9 Generalization of Customers entity type and Employees entity type to Persons entity type; Specialization works in the opposite direction

## Categorization

While specialization is done on entities, categorization defines constraints on relationship types. In most cases, categorization is exclusive, meaning that one entity participates in either relationship A or relationship B, depending on the entities' status. The status can be either a value of an attribute, a presence of another relationship, or some external status.

Categorization does not change the attributes of an entity. It requires the data access and manipulation to consider the constraints specified in the categorization.

A good example of categorization is Vehicles. Depending on the kind of vehicle, we need to build different relationships. For trucks, we need the cargo information, while for busses, we need the names of passengers. This information will be used in different relationships to deliver meaningful context to the relationship.

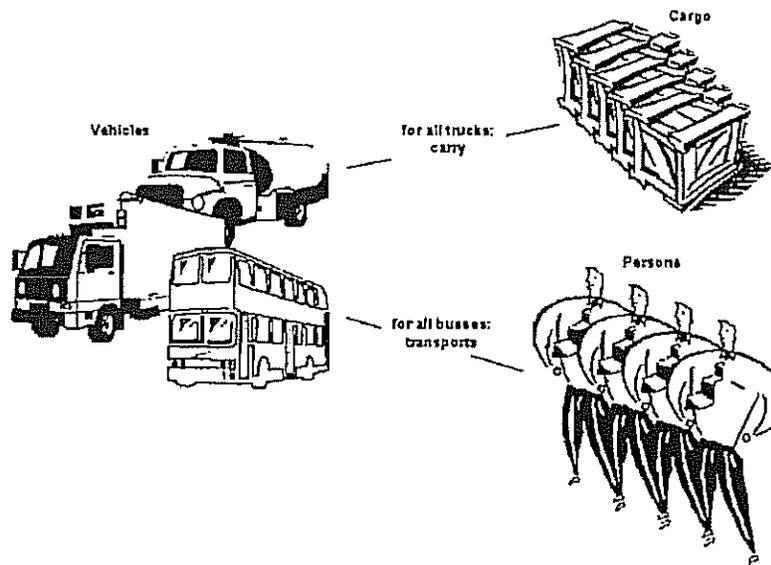


Figure 10 Categorization in trucks and busses with relationship types to Cargo entity type and Persons entity type depending on the category

## Notations for the ER Methodology

Today, several notations are used within ER modeling, including Chen's ER, Barker ER Information Engineering (IE), and IDEFIX, most of which also cover the relational notation.

Object role modeling (ORM) notation is also a player in the ER methodology. It is able to express very precise and complete business rules and can illustrate constraints graphically. Unfortunately, this level of detail requires huge diagrams with plenty of details. The question related to ORM is whether we need the precision of the notation at the level of the ER model. The ORM notation is also very complex and differentiates completely from other notations, making it less understandable to project team members not using ORM.

UML is a notation language initially created for software design that has expanded into business and database design. It includes elements and diagrams necessary to specify everything from analysis to implementation to deployment. By employing several types of diagrams and tens of different elements, UML is able to express different levels of abstraction for a system. This is a very unique capability. However, we do not need to know all of UML or every possible view and representation to successfully use UML for ER modeling.

To better understand the role that UML plays in the ER methodology, we will describe how UML handles each of the core elements of ER modeling that have been outlined earlier in this white paper.

### Entity Types in UML

As was mentioned earlier, an entity type identifies a series of artifacts of the same structure. The entity type is a blueprint that can produce any number of artifacts that differ from each other only by their identity and status.

The corresponding element of the UML is the class. The class by definition has the ability to hide the content, while the entity has accessible interfaces. This seems to be contradictory, but it is not. UML allows the class to publicize the structure using public attributes.

Classes are drawn as rectangles with up to three compartments:

Compartment one includes the stereotype and the name of the class. Stereotype refers to the further classification of elements in UML to enforce common characteristics. For example all of the legacy classes could carry the stereotype `legacy` to immediately classify them as not modifiable. While the class itself is a representation of a type, we classify the type with the stereotype `<<Entity>>` (`<<...>>` is the syntax used to specify the stereotype).

Compartment two contains attributes with types and visibility. It can also contain other details about attributes such as initialization value and stereotype. The second compartment can be omitted when displayed in overview diagrams.

Compartment three is reserved for the behavior of the class. Since the entity type does not need the behavior, we will omit this compartment.

The class can be displayed with one, two, or three compartments on diagrams depending on the level of abstraction.

An entity is an instance of the entity type. In UML, object is an instance of a class. This means that the entity itself corresponds to the object.

The representation of the object is derived from the representation of the class. The most visible difference is that the name of the object is underlined and that there are only one or two compartments.

The first compartment contains the optional stereotype, the name of the object, and the name of the derived class, divided by a colon. At least one of the names must be specified. The second compartment contains the relevant attributes with their values.

A great way to represent the object is to use just one compartment and specify the identifier<sup>1</sup> as the name of the object.



Figure 11 Entity type Employees and entity 553-32-2222 displayed as class and object in UML

### Attributes in UML

The status and information about the entity is stored as attributes of an object. The attributes of entity types have to be visible – or public – to other entity types. The attributes are specified with the visibility, name, and type in the class specification. The type of the attributes is the analysis type. It may change during the design phase.

The attributes are specified in the second compartment of the class. They contain the visibility specification that is always “+” (public) for entities. The name is divided from the type by a colon.

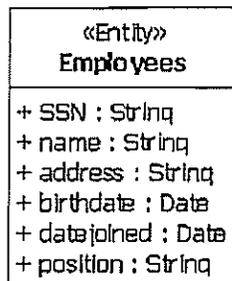


Figure 12 Entity Type Employees with public attributes SSN, name, address, birth date, date joined, and position

### Relationship Types in UML

All relationships between classes in UML are specified for the type and not for the instance. The numbers at both ends of the relationship specify the cardinality: the number of possible instances participating in the relationship.

The name of a relationship is specified directly on the relationship line and is used to identify the relationship. It can help a reader understand why the relationship exists. If a relationship is not named, role names may be used to help a reader

<sup>1</sup> Identifier is the chosen attribute that uniquely identifies an object from other objects.

understand a relationship. The figure below can be read as Products are owned by Employees or Employees are owner of Products. The singular and plural wording in the description of the relationship is defined by cardinality.

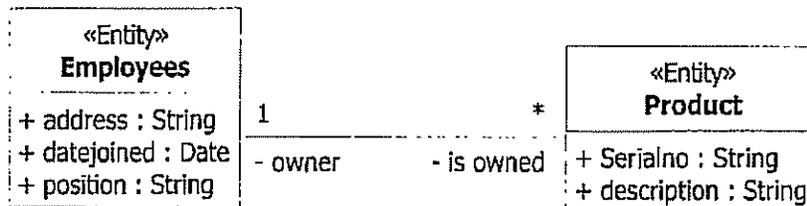


Figure 13 Relationship between the entity types Employee and Product

The data types used in the entity types are not standardized. Users are allowed to use any data type needed. It is a good practice to create a glossary with all data types to allow standardization and understanding across several designers and projects within a company.

### Attributes of Relationship Types in UML

A relationship can have attributes that are shown in UML association classes. The association class is displayed in a rectangle and contains the list of public attributes for the relationship. The association class is attached to the relationship with a dotted line. The class does not need a stereotype to explain the use and classify the class because the attachment already defines it.

The attributes in the association class are specified with the visibility, name, and type.

Although it seems like the association class, the attachment, and the relationship are independent elements, they actually represent the same element. The names of the relationship and the association class must be related.

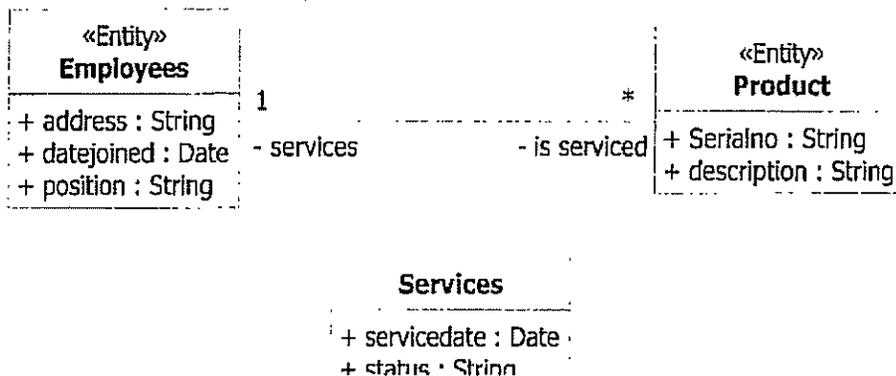


Figure 14 Attributes of a relationship type Services are specified in the association class

### Simple Constraints in UML

#### Cardinality

UML defines a very consistent way to specify cardinality. It is always specified by numbers at each end of relationships. Possible definitions include a single number for a specified cardinality of certain number of instances, which could be also unlimited, and a pair of numbers divided by “..” that specify a range for cardinality. The symbol used for unlimited cardinality is “\*” and can be used either alone, meaning an optional unlimited relationship, or in combination with another low value to specify the mandatory relationship (like “1..\*”). The values for lower and upper limits of the cardinality can be any positive number or “\*”, where the first number must be smaller or equal to the second one.

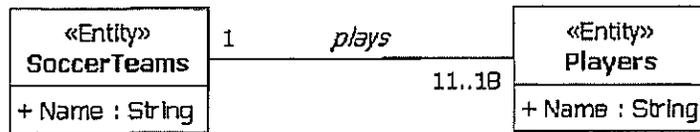


Figure 15 The SoccerTeam entity type defines a relationship plays to 11 to 18 players of the entity type Player

### Dependency

UML distinguishes two forms of dependencies between entity types. An aggregation is a dependency between two entity types that is required for the existence of the dependent entity type. The syntax for aggregation in UML is a hollow diamond on the side of the aggregate. The same side has a mandatory cardinality of 1, which can be omitted.

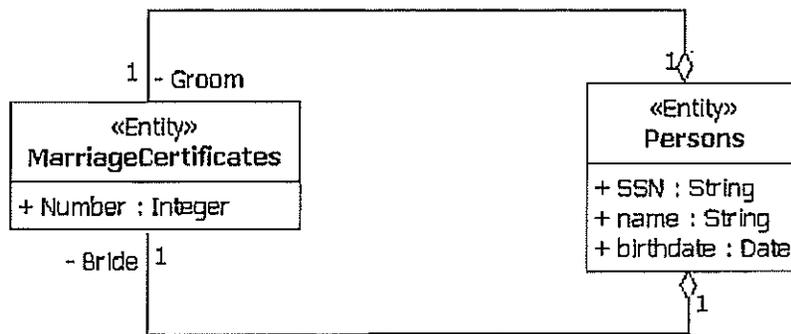


Figure 16 Each MarriageCertificates entity type depends on two Persons in the role of a Bride and a Groom

Aggregation is used when the aggregate is not unique for all of the dependencies and not all of the dependent instances must relate to the same entity. In case the aggregate is a single entity for all of the dependencies, UML specifies a strong dependency called composition. Composition is represented in the graphics as a filled diamond on the aggregate side. This relationship is used when the aggregate contains the subordinate entity type.

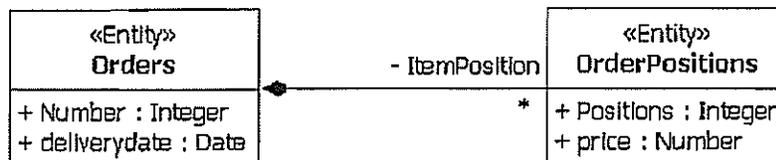


Figure 17 The entity type OrderPositions is completely defined by the entity type Orders as specified by the composition

Cardinality can be combined with aggregation and composition to define constraints on these relationships.

### Specialization and Generalization

An essential part of analysis is dedicated to the similarities and differences of entity types. Specialization reduces the risks of under specification and reduces the lack of requirements by inheriting requirements from the parent. Generalization simplifies the model and the implementation of entity types in the system.

Generalization is defined in UML by a hollow arrow on the parent side of the relationship. Generalization is not a relationship between two entity types. It is a derivation of the general entity type to the specialized entity type. Cardinality is not allowed on generalization relationships.

All of the attributes and relationships of the parent entity type are inherited by the specialized entity type. Both the attributes and relationships to other entity types cannot be removed from the specialized entity type.

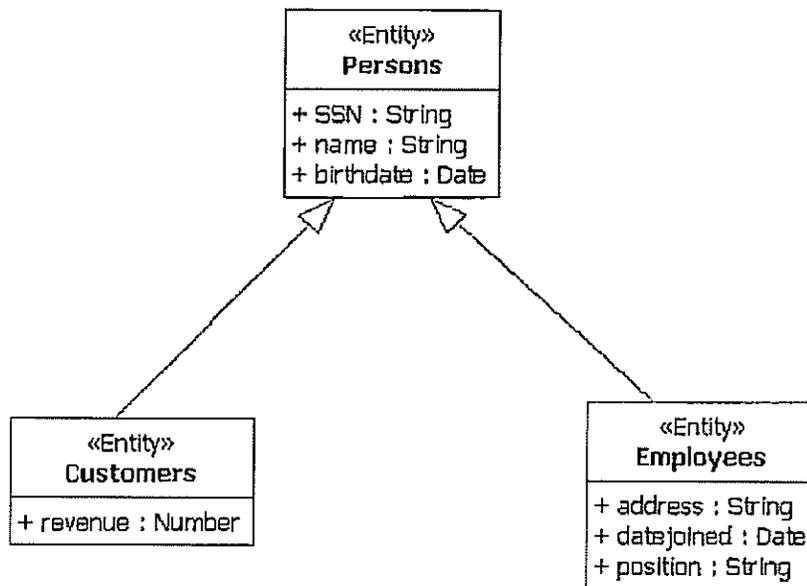


Figure 18 Generalization defines the entity types Employees and Customers as type of the entity type Persons Inheriting Persons attributes

## Categorization

Categories of the same entity type specify how the entities relate to each other depending on their characteristics, like all employees on leave of absence for example. The syntax of the categorization uses the constraints specified in OCL (object constraint language) connected to the relationship.

The constraints are meant to specify dynamic behavior. When constraints evaluate as valid, the relationships are valid.

Typically the constraints are mutually exclusive, which can be modeled using a constraint {xor} between exclusive relationships.

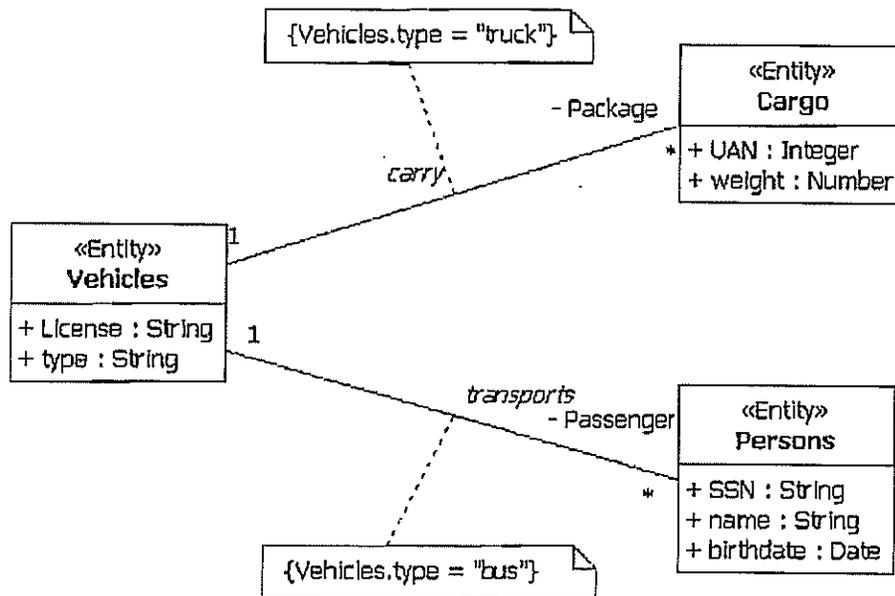


Figure 19 Categorization of entity types defines the criteria for relationship types – Cargo is important for Vehicles of the type “truck”; Persons can be transported for the Vehicles of the type “bus”

## Summary

Against the common opinion, ER methodology is not limited to development of relational databases. I have to agree that in most cases the output of the design process will be realized in a relational database; however this is not a condition for it. ER methodology is focused on artifact-based design. Its output is artifacts (entity types) with relationships between them and constraints clarifying entity types and relationships. This output can be used to create a relational model that will have additional technology-related constraints.

The ER methodology is used for analysis and design of artifact-driven systems. It is used for conceptual and logical modeling, but is not intended for physical design. The ER methodology describes only the static view of the artifacts, not the dynamics of the system.

To create a smooth development process, it is important that all members of the development team “speak a common language.” That’s because misinterpretation of information can cause delays, create unexpected errors, and reduce the overall efficiencies of team members. UML helps eliminate these concerns by providing a standardized language that is easily understood by all members of the system development team.



### **IBM software integrated solutions**

IBM Rational supports a wealth of other offerings from IBM software. IBM software solutions can give you the power to achieve your priority business and IT goals.

- *DB2<sup>®</sup> software helps you leverage information with solutions for data enablement, data management, and data distribution.*
- *Lotus<sup>®</sup> software helps your staff be productive with solutions for authoring, managing, communicating, and sharing knowledge.*
- *Tivoli<sup>®</sup> software helps you manage the technology that runs your e-business infrastructure.*
- *WebSphere<sup>®</sup> software helps you extend your existing business-critical processes to the Web.*
- *Rational<sup>®</sup> software helps you improve your software development capability with tools, services, and best practices.*

### **Rational software from IBM**

Rational software from IBM helps organizations create business value by improving their software development capability. The Rational software development platform integrates software engineering best practices, tools, and services. With it, organizations thrive in an on demand world by being more responsive, resilient, and focused. Rational's standards-based, cross-platform solution helps software development teams create and extend business applications, embedded systems and software products. Ninety-eight of the Fortune 100 rely on Rational tools to build better software, faster. Additional information is available at [www.rational.com](http://www.rational.com) and [www.therationaleledge.com](http://www.therationaleledge.com), the monthly e-zine for the Rational community.

IBM is a wholly owned subsidiary of the IBM Corporation. (c) Copyright Rational Software Corporation, 2003. All rights reserved.

IBM Corporation  
Software Group  
Route 100  
Somers, NY 10589  
U.S.A.

Printed in the United States of America  
05-03 All Rights Reserved. Made in the U.S.A.

IBM and the IBM logo are trademarks of International Business Machines Corporation in the United States, other countries, or both.

Rational, and the Rational logo are trademarks or registered trademarks of Rational Software Corporation in the United States, other countries or both.

Other company, product or service names may be trademarks or service marks of others.

The IBM home page on the Internet can be found at [ibm.com](http://ibm.com)

TP202

**Schwartz, Andrew**

---

**From:** Strafford Director of CLE [custserv@spb-g.com]  
**Sent:** Tuesday, September 21, 2010 11:43 AM  
**To:** Schwartz, Andrew  
**Subject:** Bankruptcy & Attorney-Client Privilege CLE: Thursday, September 30th, 1:00 PM EDT

[Upcoming live interactive CLE web seminar — cutting-edge information without leaving the office! View on a web page](#)

---

## **Strafford CLE Webinars**

---

# **Bankruptcy: Attorney-Client Privilege and Work Product**

Protecting and Maintaining Confidentiality and the Work Product Defense

Thursday, September 30, 1:00pm-2:30pm EDT

**REGISTER NOW >**

*Program outline and faculty bios now available for your review*

This CLE live web seminar will provide guidance for bankruptcy counsel on the implications of the attorney-client privilege for stakeholders in the bankruptcy process—debtors, creditors, committees and professionals. The panel will discuss legal strategies for preserving the privilege and leveraging the work product defense.

Following the speaker presentations, you'll have an opportunity to get answers to your specific questions during the interactive Q&A.

### **MORE INFORMATION >**

Including the program outline and faculty bios

Promo code: BLLBZC

Call 1-800-926-7926 ext. 10

### **FACULTY**

Gregory W.  
Werkheiser  
*Morris Nichols Arsht  
& Tunnell*

Ronald R. Sussman  
*Cooley*

Alec P. Ostrow  
*Becker Glynn  
Melamed & Muffly*



---

Confidentiality Statement: The information contained in this email and any attachments is legally privileged and confidential. However, if the information included would be of value or pertinent to other individuals in your organization, please feel free to pass it along. If you have received this email in error, please notify the sender. Thank you.

This promotional message is sent to you by Strafford Publications. [Manage your email preferences](#)

Strafford Publications, Inc.  
360 North Albany Road, NE PO Box 13729  
Atlanta, GA 30321-0729 USA  
Phone: 1-800-926-7926 x 10 or 404-531-1141 x 10  
© 2010 by Strafford Publications, Inc. All rights reserved. Copying is prohibited.

# Entity-Relationship Modeling: Historical Events, Future Trends, and Lessons Learned

Peter P. Chen

Computer Science Department  
Louisiana State University  
Baton Rouge, LA 70803, USA  
E-mail: [pchen@lsu.edu](mailto:pchen@lsu.edu)

**Abstract.** This paper describes the historical developments of the ER model from the 70's to recent years. It starts with a discussion of the motivations and the environmental factors in the early days. Then, the paper points out the role of the ER model in the Computer-Aided Software Engineering (CASE) movement in the late 80's and early 90's. It also describes the possibility of the role of author's Chinese culture heritage in the development of the ER model. In that context, the relationships between natural languages (including Ancient Egyptian hieroglyphs) and ER concepts are explored. Finally, the lessons learned and future directions are presented.

## 1 Introduction

Entity-Relationship (ER) modeling is an important step in information system design and software engineering. In this paper, we will describe not only the history of the development of the ER approach but also the reactions and new developments since then. In this perspective, this paper may be a little bit different from some other papers in this volume because we are not just talking about historical events that happened twenty or thirty years ago, we will also talk about the consequences and relevant developments in the past twenty-five years. At the end, we will talk about lessons learned during this time period. In particular, we intend to show that it is possible that one concept such as the ER concept can be applied to many different things across a long time horizon (for more than twenty-five years) in this fast-changing Information Technology area.

This paper is divided into 8 sections. Section 1 is the Introduction. In Section 2, the historical background and events happened around twenty-five years ago will be explained. For example, what happened at that time, what the competing forces were, and what triggered researchers like the author to work on this topic will be explained. Section 3 describes the initial reactions in the first five years from 1976 to 1981. For example, what the academic world and the industry viewed the ER model initially? Section 4 states the developments in the next twenty years from 1981 to 2001. In particular, the role of the ER model in the Computer-Aided Software Engineering (CASE) will be discussed. Section 5 describes a possible reason for the author to come up with the ER modeling idea, that is, the author's Chinese culture heritage. The author did not think about this particular reason until about fifteen years ago. Section 6 presents our view of the future of ER modeling. Section 7 states the lessons learned. For those of you who have similar experience in the past twenty-five years, you probably have recognized similar principles and lessons in this section. For those who just started their professional careers recently, we hope the lessons learned by the author will be helpful to those readers. Section 8 is the conclusion.

## 2 Historical Background

In this section, we will look at the competing forces, the needs of the computer industry at that time, how the ER model was developed, and the main differences between the ER model and the relational model.

### 2.1 Competing Forces

First, Let us look at the competing forces in the computer software area at that time. What are the competing forces then? What triggered people like the author to work on this area (data models) and this particular topic (ER modeling)? In the following, we will discuss the competing forces in the industry and in the academic world in the early 70's,

**Competing Forces in the industry.** There were several competing data models that had been implemented as commercial products in the early 70's: the file system model, the hierarchical model (such as IBM's IMS database system), and the Network model (such as Honeywell's IDS database system). The Network model, also known as the CODASYL model, was developed by Charles Bachman, who received the ACM Turing Award in 1973. Most organizations at that time used file systems, and not too many used database systems. Some people were working on developing better data or index structures for storing and retrieving data such as the B+-tree by Bayer and McCreight [1].

**Competing Forces in the Academic World.** In 1970, the relational model was proposed, and it generated considerable interest in the academic community. It is correct to say that in the early 70's, most people in the academic world worked on relational model instead of other models. One of the main reasons is that many professors had a difficult time to understand the long and dry manuals of commercial database management systems, and Codd's relational model paper [2] was written in a much more concise and scientific style. For his contributions in the development of the relational model, Codd received ACM Turing Award in 1981.

*Most People were working on DBMS Prototypes.* Many people at that time in the academic world or in the industry worked on the implementation of database management system prototypes. Most of them were based on the relational model.

*Most Academic People were investigating the definitions and algorithms for the Normal Forms of Relations.* A lot of academic people worked on normalization of relations because only mathematical skills were needed to work on this subject. They could work on the improvement of existing algorithms for well-defined normal forms. Or, they could work on new normal forms. The speed of research moved very fast in the development of normal forms and can be illustrated by the following scenario. Let us say that several people were ready to publish their results on normal forms. Assuming that one person published a paper on 4<sup>th</sup> normal form and another person who had written a paper on 4<sup>th</sup> normal form but had not published it yet, the 2<sup>nd</sup> person would have changed the title of the paper from 4<sup>th</sup> normal form to 5<sup>th</sup> normal form. Then, the rest would work on the 6<sup>th</sup> normal form. This became an endless game till one day somebody wrote a paper claiming that he had an infinity-th normal form and arguing that it did not make any sense to continue this game. Most practitioners also said loudly that any relational normal form higher than 3<sup>rd</sup> or 4<sup>th</sup> won't have practical significance. As a result, the game of pursuing the next normal form finally ran out of steams.

### 2.2 Needs of the System Software in the Early 70's

**The Needs of the Hardware/Software Vendors.** In terms of software vendors at that time, there were urgent needs for (1) integration of various file and database formats and (2) incorporating more "data semantics" into the data models.

**The Needs of the User Organizations.** For user organizations such as General Motors and Citibank, there were urgent needs for (1) a unified methodology for file and database design for various file and database system available in the commercial market and (2) incorporation of more data semantics including business rules into the requirements and design specifications.

### 2.3 How the ERM was Developed

Here, we will give some personal history of the development of the ER model: where the author was and what the author did in the early 70's, particularly on how the author developed the ER model.

**Harvard (Sept. '69 to June '73).** After the author got a B.S. in Electrical Engineering from National Taiwan University in 1968, the author received a fellowship to study Computer Science (at that time, it was a part of Applied Mathematics) at Harvard graduate school. The author received the Ph.D. degree in 1973. The thesis was very mathematically oriented – focusing on the file allocation problems in a storage hierarchy using the queuing theory and mathematical programming techniques. The knowledge the author learned in EE, CS and applied math was crucial in the development of the ER model in subsequent years.

**Honeywell and Digital (June '73 to August '74).** The author joined Honeywell Information Systems in Waltham, MA in June '73. He participated in the “next-generation computer system” project to develop a computer system based on distributed system architecture. There were about ten people in the team, and most of them were at least twenty years senior than the author. The team consisted of several well-known computer experts including Charles Bachman. One of the requirements of such a “distributed system” was to make the files and databases in different nodes of the network compatible with each other. The ER model was motivated by this requirement. Even though the author started to crystallize the concepts in his mind when he worked for Honeywell, he did not write or speak to anyone about this concept then. Around June of 1974, Honeywell abandoned the “next-generation computer system” project, and all the project team members went different ways. The author then spent three months at Digital Equipment Corporation in Maynard, MA to develop a computer performance model for the PDP-10 system.

**MIT Sloan School of Management (1974 – 1978).** In September 1974, the author joined MIT Sloan School of Management as an Assistant Professor. This was the place that he put the ER ideas down into an article. Being a professor in a business/management school provided the author many opportunities to interact with the user organizations. In particular, he was particularly impressed by a common need of many organization to have a unified methodology for file structure and database design. This observation certainly influenced the development of the ER model. As a result, the first ER paper was first presented at 1st International Conference on Very Large Databases in 1975 and subsequently published in the first issue of ACM Transactions on Database Systems [3] in March of 1976.

#### 2.4 Fulfilling the Needs

How did the ER model fulfill the needs of the vendor and user organizations at that time? We will first start with the graphical representation and theoretical foundations of the ER model. Then, we will explain the significant differences between the ER model and the relational model.

**The Concepts of Entity, Relationship, Types, and Roles.** In Fig. 1, there are two entities; both of them are of the “Person” type. There is a relationship called, “is-married-to,” between these two persons. In this relationship, each of these two Person entities has a role. One person plays the role of “husband,” and another person plays the role of “wife.”

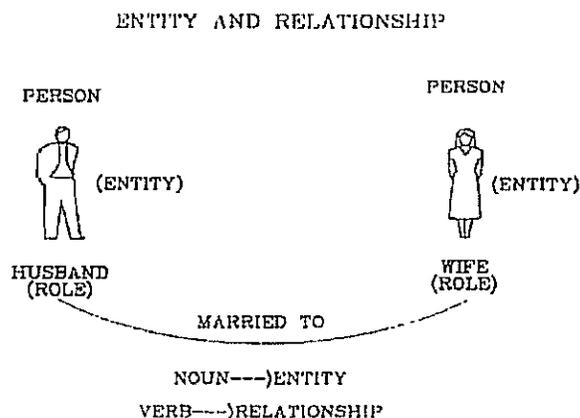


Fig. 1. The Concept of Entity and Relationship

**The Entity-Relationship (ER) Diagram.** One of the key techniques in ER modeling is to document the entity and relationship types in a graphical form called, Entity-Relationship (ER) diagram. Figure 2 is a typical ER diagram. The entity types such as EMP and PROJ are depicted as rectangular boxes, and the relationship types such as WORK-FOR are depicted as a diamond-shaped box. The value sets (domains) such as EMP#, NAME, and PHONE are depicted as circles, while attributes are the "mappings" from entity and relationships types to the value sets. The cardinality information of relationship is also expressed. For example, the "1" or "N" on the lines between the entity types and relationship types indicated the upper limit of the entities of that entity type participating in that relationships.

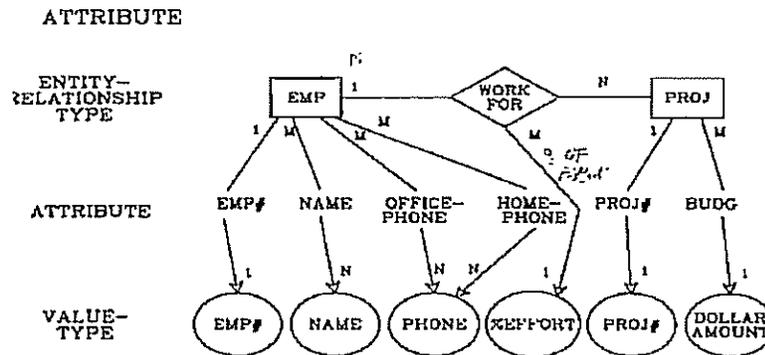


Fig. 2. An Entity-Relationship (ER) Diagram

**ER Model is based on Strong Mathematical Foundations.** The ER model is based on (1) Set Theory, (2) Mathematical Relations, (3) Modern Algebra, (4) Logic, and (5) Lattice Theory. A formal definition of the entity and relationship concepts can be found in Fig. 3.

### SET THEORY (DEFINITIONS)

ENTITY	$e$
ENTITY SET	$E; e \in E$
VALUE	$v$
VALUE SET	$V; v \in V$
RELATIONSHIP	$r$
RELATIONSHIP SET	$R; r \in R$

A RELATIONSHIP SET IS DEFINED AS A "MATHEMATICAL RELATION" ON ENTITY SETS

$$R = \{r_1, r_2, \dots, r_n\}$$

$$r_i = [e_1, e_2, \dots, e_n] | e_1 \in E_1, \dots, e_n \in E_n$$

Fig. 3. Formal Definitions of Entity and Relationship Concepts

**Significant Differences between the ER model and the Relational Model.** There are several differences between the ER model and the Relational Model:

*ER Model uses the Mathematical Relation Construct to Express the Relationships between Entities.* The relational model and the ER model both use the mathematical structure called Cartesian product. In some way, both models look the same – both use the mathematical structure that utilizes the Cartesian product of something. As can be seen in Figure 3, a relationship in the ER model is defined as an ordered tuple of "entities." In the relational model, a Cartesian product of data "domains" is a "relation," while in the ER model a Cartesian product of "entities" is a "relationships." In other words, in the relational model the

mathematical relation construct is used to express the "structure of data values," while in the ER model the same construct is used to express the "structure of entities."

*ER Model Contains More Semantic Information than the Relational Model.* By the original definition of relation by Codd, any table is a relation. There is very little in the semantics of what a relation is or should be. The ER model adds the semantics of data to a data structure. Several years later, Codd developed a data model called RM/T, which incorporated some of the concepts of the ER model.

*ER Model has Explicit Linkage between Entities.* As can be seen in Figures 2 and 4, the linkage between entities is explicit in the ER model while in the relational model is implicit. In addition, the cardinality information is explicit in the ER model, and some of the cardinality information is not captured in the relational model.

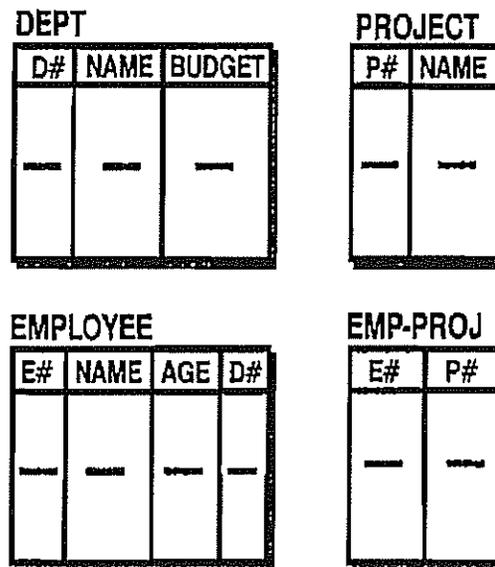


Fig. 4. Relational Model of Data

### 3. Initial Reactions & Reactions in the First Five Years (1976 – 1981)

#### 3.1 First Paper Published & Codd's Reactions

As stated before, the first ER model paper was published in 1976. Codd wrote a long letter to the editor of ACM Transaction on Database Systems criticizing the author's paper. The author was not privileged to see the letter. The editor of the Journal told the author that the letter was very long and single-spacing. In any case, Dr. Codd was not pleased with the ER model paper. Ironically, several years later, Codd proposed a new version of the relational data model called RM/T, which incorporated some concepts of the ER model. Perhaps, the first paper on the ER model was not as bad as Codd initially thought. Furthermore, in the 90's, the Codd and Date consulting group invited the author to serve as a keynote speaker (together with Codd) several times in their database symposia in London. This indicates that the acceptance of ER model was so wide spread so that initial unbelievers either became convinced or found it difficult to ignore.

#### 3.2 Other Initial Reactions and Advices

During that time, there was a "religious war" between different camps of data models. In particular, there was a big debate between the supporters of the Relational model and that of the Network model. Suddenly, a young assistant professor wrote a paper talking about a "unified data model." In some sense, the author

was a “new kid on the block” being thrown into the middle of a battle between two giants. The advice the author got at that time was: “why don’t you do the research on the n-th normal form like most other researchers do? It would be much easier to get your normal form papers published.” That was an example of the type of advices the author got at that time. Even though those advices were based on good intentions and wisdom, the author did not follow that type of advices because he believed that he could make a more significant contribution to the field by continuing working on this topic (for example, [4-13]). It was a tough choice for a person just starting the career. You can imagine how much problems or attacks the author had received in the first few years after publishing the first ER paper. It was a very dangerous but a very rewarding decision the author made that not only had a significant impact on the author’s career but also the daily practices of many information-modeling professionals.

### **3.3 IDEF, ICAM, and Other Believers**

There were a small but growing number of believers of the ER or similar data models. For example, Mike Hammer, who was an Assistant Professor at the EECS department of MIT, developed the Semantic Data Model with his student, Dennis McCleod. Later on, Hammer applied the idea in reverse engineering in the IT field to organization restructuring and became a management guru. Outside of the academic world, the industry and government agencies began to see the potential benefits of ER modeling. In the late 70’s, the author served as a consultant in a team that developed the data modeling methodology for the ICAM (Integrated Computer-Aided Manufacturing) project sponsored by the U.S. Air Force. One of the objectives was to develop at least two modeling methodologies for modeling the aircraft manufacturing processes and data: one methodology for process modeling and one for data modeling. The data modeling methodology was called IDEF1 methodology and has been used widely in US military projects.

### **3.4 Starting a Series of ER Conferences**

The first ER conference was held in UCLA in 1979. We were expecting 50 people, but 250 to 300 people showed up. That was a big surprise. Initially, the ER conference was a bi-annual event, but now it is an annual event being held in different parts of the world [14]. In November of this year (Year 2001), it will be held in Japan [15], and next year (Year 2002) it will be held in Finland. This series of conferences has become a major annual forum for exchanging ideas between researchers and practitioners in conceptual modeling.

## **4 The Next Twenty Years (’81 –’01)**

### **4.1 ER Model Adopted as a Standard for Repository Systems and ANSI IRDS.**

In the 80’s, many vendors and user organizations recognized the need for a repository system to keep track of information resources in an organization and to serve as the focal point for planning, tracking, and monitoring the changes of hardware and software in various information systems in an organization. It turned out that the ER model was a good data model for repository systems. Around 1987, ANSI adopted the ER model as the data model for Information Resource Directory Systems (IRDS) standards. Several repository systems were implemented based on the ER model including IBM’s Repository Manager for DB2 and DEC’s CDD+ system.

### **4.2 ER Model as a Driving Force for Computer-Aided Software Engineering (CASE) tools and Industry**

Software development has been a nightmare for many years since the 50’s. In the late 80’s, IBM and others recognized the needs for methodologies and tools for Computer-Aided Software Engineering (CASE). IBM proposed a software development framework and repository system called, AD Cycle and the Repository Manager that used the ER model as the data model. The author was one of the leaders who actively preached the technical approach and practical applications of CASE. In 1987, Digital Consulting Inc. (DCI) in Andover, Mass., founded by Dr. George Schussel, organized the 1<sup>st</sup> Symposium on CASE in Atlanta and invited the author to be one of the two keynote speakers. To everybody’s surprise, the symposium was a huge commercial success, and DCI grew from a small company to a major force in the symposium and trade show business.

### 4.3 Object-Oriented (OO) Analysis Techniques are Partially Based on the ER Concepts

It is commonly acknowledged that one major component of the object-oriented (OO) analysis techniques are based on the ER concepts. However, the "relationship" concept in the OO analysis techniques are still hierarchy-oriented and not yet equal to the general relationship concept advocated in the ER model. It is noticeable in the past few years that the OO analysis techniques are moving toward the direction of adopting a more general relationship concept.

### 4.4 Data Mining is a Way to Discover Hidden Relationships

Many of you have heard about data mining. If you think deeply about what the data mining actually does, you will see the linkage between data mining and the ER model. What is data mining? What does the data mining really is doing? In our view, it is a discovery of "hidden relationships" between data entities. The relationships exist already, and we need to discover them and then take advantage of them. This is different from conventional database design in which the database designers identify the relationships. In data mining, algorithms instead of humans are used to discover the hidden relationships.

## 5 In Retrospect: Another Important Factor – Chinese Culture Heritage

### 5.1 Chinese Culture Heritage

Many people asked the author how he got the idea of the Entity-Relationship model. After he kept on getting that kind of questions, the author thought it might be related to something that many people in Western culture may not have. After some soul searching, the author thought it could be related to his Chinese culture heritage. There are some concepts in Chinese character development and evolution that are closely related to modeling of the things in the real world.

Here is an example. Figure 5 shows the Chinese characters of "sun", "moon, and "person". As you can see, these characters are a close resemblance of the real world entities. Initially, many of the lines in the characters are made of curves. Because it was easier to cut straight lines on oracle bones, the curves became straight lines. Therefore, the current forms of the Chinese characters are of different shapes.

<u>Original Form</u>	<u>Current Form</u>	<u>Meaning</u>
		Sun
		Moon
		Person

Fig. 5. Chinese Characters that Represent the Real-World Entities

Chinese characters also have several principles for "composition." For example, Figure 6 shows how two characters, SUN and MOON, are composed into a new character. How do we know the meaning of the new character? Let us first think: what does sun and moon have in common? If your answer is: both reflect lights, it is not difficult to guess the meaning of the new character is "brightness." There are other principles of composing Chinese characters [10].

 (sun) +  (moon) =  (Bright/ Brightness by light)

Fig. 6. Composition of Two Chinese Characters into a New Chinese Character

What does the Chinese character construction principles have to do with ER modeling? The answer is: both Chinese characters and the ER model are trying to model the world – trying to use graphics to represent the entities in the real world. Therefore, there should be some similarities in their constructs.

## 5.2 Ancient Egyptian Hieroglyphs

Besides Chinese characters, there are other languages have graphic characters. Ancient Egyptian language is one of them. It turns out that there are several characters in ancient Egyptian characters are virtually the same as the Chinese characters. One is “sun”, another is “mouth, and the third one is “water.” It is amazing that both the Egyptian people and the Chinese people developed very similar characters even though they were thousands of miles away and had virtually no communication at that time. Ancient Egyptian Hieroglyphs also have the concept of composition. Interested readers should refer to [11].

Hieroglyph	Meaning	Hieroglyph	Meaning
(a) 	lower arm	(f) 	man
(b) 	mouth	(g) 	woman
(c) 	viper	(h) 	sun
(d) 	owl	(i) 	house
(e) 	sieve	(j) 	water

Fig. 7. Ancient Egyptian Hieroglyphs

## 6 The Future

### 6.1 XML and ER Model.

In the past few years, the author has been involved in the developing the “standards” for XML. He has participated in two XML Working Groups of the World Wide Web Consortium (W3C) as an invited expert. During this involvement, some similarities between XML and the ER model were discovered including the following:

**RDF and the ER Model.** There are several components in the XML family. One of them is RDF, which stands for Resource definition Framework. This is a technology that Tim Berners-Lee, the Director of W3C, pushes very hard as a tool for describing the meta-data in the web. There are some similarities and differences between RDF and the ER model, and Mr. Berners-Lee has written several articles discussing this issue. In a joint meeting of the RDF and Schema Working Groups over one year ago, they issued the Cambridge Communiqué [16] that states: “...RDF can be viewed as a member of the Entity-Relationship model family...”

**XLink and the ER model.** Most of us are familiar with the hyperlink in HTML. The XLink Working Group of W3C has been trying to do is to develop a new kind of hyperlink for XML. In HTML, the hyperlink is basically a “physical pointer” because it specifies the exact URL of the target. In XLink, the new link is one step closer to a “logical pointer.” In the evolution of operating systems, we have been moving from physical pointers to logical pointers. The XLink Working Group proposed a new structure called, “extended link.” For example, Fig. 8 is an extended link for five remote resources. The extended link concept in XML is very similar to the n-ary relationship concept in the ER model. Figure 8 can be viewed as a relationship type defined on 5 entity types.

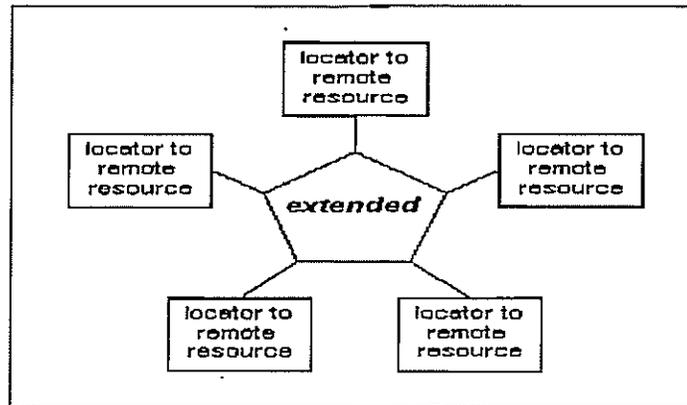


Fig. 8. "Extended Link" in XML is Similar to the N-ary Relationship Concept in the ER Model

## 6.2. Theory of the Web

One thing that is still missing today is the theory of the web. The ER model could be one of the foundations for the theory of the Web. The author plans to work on that topic and would encourage the readers to work on the subject, too.

## 7 Lesson Learned

### 7.1 Reflections on Career choices

In the past twenty-five years, the author made some tough career choices as some of the other authors in this volume did. It is the hope of the author that our experience will be useful to some other people who just started their professional careers and are making their career choices. Here are some reflections based on the author's own experience:

**Right idea, right place, right time, and belief in yourself.** In order to have your idea be accepted by other people, you need not only to have the right idea but also to present them at the right place and right time. You also need "persistence." In other words, you need to believe in yourself. This is probably the most difficult part because you have to endure some unnecessary pressures and criticisms when you are persistent on your idea and try to push it forward. Hopefully, some days in the future, you will be proved to be right. At that time, you will be happy that you have persisted.

**Getting Fresh Ideas from Unconventional Places.** After working on a particular area for a while, you may run out of "big" ideas. You may still have some "good" ideas to get you going, but those ideas are not "earth-breaking." At that time, you need to look for ideas in different subject areas and to talk to new people. For example, most of us are immersed in Western culture, and learning another culture may trigger new ways of thinking. Similarly, you may look into some fields outside of information technology such as Physics, Chemistry, Biology, or Architecture to find fresh ideas. By looking at the theories, techniques, and approaches used in other fields, you may get very innovative ideas to make a breakthrough in the IT field.

### 7.2 Implications of the Similarity and differences between the Chinese Characters and Ancient Egyptian Hieroglyphs on Software Engineering and Systems Development Methodologies

As we pointed out earlier, there are several Chinese characters that are almost the same as their counterparts in ancient Egyptian hieroglyphs. What does this mean? One possible answer is that human beings think alike even though there was virtually no communication between ancient Chinese people and ancient Egyptian people. It is very likely that the way to conceptualize basic things in the real world is

common to most of the races and cultures. As was discussed earlier, the construction and developments of other characters are different in Chinese and in Ancient Egyptian Hieroglyphs. It is valid to say that the language developments were dependent on the local environment and culture. What is the implication of the similarities and differences in character developments on the development of software engineering and information system development methodologies? The answer could be: some basic concepts and guidelines of software engineering and system development methodologies can be uniformly applied to all people in the world while some other parts of the methodologies may need to be adapted to local cultures and customs.

## **8. Conclusions**

The author was very fortunate to have the opportunity to meet the right people and to be given the opportunity to develop the Entity-Relationship (ER) model at the time and environment such a model was needed. The author is very grateful to many other researchers who have continued to advance the theory of the ER approach and to many software professionals who have practiced ER modeling in their daily jobs in the past twenty-five years. We believe that the concepts of entity and relationship are very fundamental concepts in software engineering and information system development. In the future, we will see new applications of these concepts in the Web and other new frontiers of the software world.

## References

1. Bayer, R. and McCreight, E., "Organization and Maintenance of Large Ordered Indexes," Acta Informatica, Vol. 1, Fasc. 3, 1972, pp. 173-189.
2. Codd, E. F. "The Relational Model of Data for Large Shared Data Banks," Comm. of the ACM, Vol. 13 (6), 1970, pp. 377-387.
3. Chen, P.P., "The Entity-Relationship Model: Toward a Unified View of Data," ACM Trans. on Database Systems, Vol.1, No.1, March 1976, pp. 1-36.
4. Chen, P. P., "An Algebra for a Directional Binary Entity-Relationship Model," IEEE First International Conference on Data Engineering, Los Angeles, April 1984, pp. 37-40.
5. Chen, P. P., "Database Design Using Entities and Relationships," in: S. B. Yao (ed.), Principles of Data Base Design, Prentice-Hall, NJ, 1985, pp. 174-210.
6. Chen, P. P., "The Time-Dimension in the Entity-Relationship Model," in: Information Processing '86, H. -J. Kugler (ed.), North-Holland, Amsterdam, 1986, pp. 387-390.
7. Chen, P. P. and Zvieli, A., "Entity-Relationship Modeling of Fuzzy Data," Proceedings of 2nd International Conference on Data Engineering, Los Angeles, February 1986, pp. 320-327.
8. Chen, P. P. and Li, M., "The Lattice Concept in Entity Set," in: Entity-Relationship Approach, S. Spaccapietra (ed.), North-Holland, Amsterdam, 1987, pp. 311-326.
9. Chandrasekaran, N., Iyengar, S.S., and Chen, P. P., "The Denotational Semantics of the Entity-Relationship Model," International Journal of Computer Mathematics, 1988, pp. 1-15.
10. Chen, P. P., "English, Chinese and ER Diagrams," Data & Knowledge Engineering, Vol. 23, No. 1, June 1997, pp. 5-16.
11. Chen, P. P., "From Ancient Egyptian Language to Future Conceptual Modeling," in: Conceptual Modeling: Current Issues and Future Directions, Chen, P.P., et al. (eds), Springer-Verlag, Berlin, Lecturing Notes in Computer Sciences, No. 1565, 1998, pp. 57-66.
12. Yang, A. and Chen, P. P., "Efficient Data Retrieval and Manipulation using Boolean Entity Lattice," Data & Knowledge Engineering, Vol. 20, 1996, pp.211-226.
13. <http://www.csc.lsu.edu/~chen/>
14. <http://www.er2000.byu.edu/>
15. <http://www.arislab.dnj.ynu.ac.jp/ER2001>
16. <http://www.w3.org/TR/schema-arch>

## **A Comparative Analysis of Entity-Relationship Diagrams<sup>1</sup>**

**Il-Yeol Song**  
Drexel University

**Mary Evans**  
USConnect

**E.K. Park**  
U.S. Naval Academy

The purpose of this article is to collect widely used entity-relationship diagram (ERD) notations and so their features can be easily compared, understood, and converted from one notation to another. We collected ten different ERD notations from text books and CASE tools. Each notation is depicted using a common problem and includes a discussion of each characteristic and notation. According to our investigation, we have found that ERD features and notations are different in seven features: whether they allow n-ary relationships, whether they allow attributes in a relationship, how they represent cardinality and participation constraints, the place where they specify constraints, whether they depict overlapping and disjoint subclass entity-types, whether they show total/partial specialization, and whether they model the foreign key at the ERD level. We conclude that many of the ER diagrams we studied are different in how they depict the criteria listed above. In order to convert one diagram to another, some notations must be extended and carefully converted from one notation into another. We also discuss the limitations of existing CASE tools in terms of modeling capabilities and supporting diagrams.

**Keywords:** Entity-Relationship Diagrams, ERD, design, modeling, CASE

---

<sup>1</sup>Correspondence should be addressed to Il-Yeol Song, College of Information Science and Technology, Drexel University, 32nd and Chestnut Streets, Philadelphia, PA 19104. Email: songiy@post.drexel.edu

## 1 INTRODUCTION

The purpose of this article is to collect widely used entity-relationship diagram (ERD) notations and so their features can be easily compared, understood, and converted from one notation to another. The types of ERDs we examine in this article are those used in database textbooks or CASE Tools used for the design of relational databases. We extract the most significant features of each method and notation, rather than exhaustively compare all the features of those methods.

The Entity-Relationship diagram has been widely used in structured analysis and conceptual modeling. The ER approach is easy to understand, powerful to model real-world problems and readily translated into a database schema. The ERD views that the real world consists of a collection of business entities, the relationships between them and the attributes used to describe them. Other ER modeling semantics used by most methodologies include cardinality, participation and generalization. The typical semantic constructs of the ER model and its variations we consider in this article include the following features:

- An *entity type* represents a distinguishable object type. In real-world modeling, an entity type is an important business object that contains more than one property. We will simply call an entity, instead of an entity type, as in many practice. A *weak entity* is a special type of entity whose existence is dependent upon another entity called the owner entity. This dependency is called existence dependency. Thus a weak entity does not have its own identifier. Hence, the identifier of a weak entity is a combination of the identifier of the owner entity and the partial key of the weak entity.
- A *relationship type* represents an association between or among several entities. In real-world modeling, a relationship represents an association that needs to be remembered by the database system. We will simply call relationship, instead of relationship type. A relationship type can be unary, binary, or n-ary, depending on whether the number of entities involved in the relationship is 1, 2 or more than 2.
- An *attribute* is a property that is used to describe an entity or a relationship. Note

that some methods do not allow an attribute in a relationship. An attribute which is a primary key of another relation is called a *foreign key*.

- A *cardinality* constraint specifies the number of relationship instances in which an entity can participate. They are in the form of 1:1, 1:N, M:N, in binary relationships, and 1:1:1, 1:1:N, 1:N:M, and M:N:P in ternary relationships. This constraint corresponds to *maximum* cardinality in some notations.
- A *participation* constraint specifies whether an entity instance can exist without participating in a relationship with another entity. This constraint corresponds to *minimum* constraints in some notations. *Total (or mandatory)* and *partial (or optional)* participation are the two types of participation. Total participation exists when an entity instance cannot exist without participating in a relationship with another entity instance. Partial participation exists when the entity instance can exist without participating in a relationship with another entity instance. Some methods combine cardinality and participation constraints and represent them using minimum and maximum constraints in the form of (min, max) notation.
- *Generalization/specialization* specifies superclass and subclass relationship between entity types. In a generalization/specialization hierarchy, there are two constraints - disjoint and complete [1]. The disjoint constraint specifies whether an entity can appear in more than one subclass entity (overlapping) or not (disjoint). The specialization is said to allow overlapping if one entity instance in the super class can appear in multiple subclass entities. Otherwise, the subclasses are disjoint. The second constraint is the completeness constraint. It specifies whether a super class entity instance can exist without belonging to at least one subclass entity (partial specialization) or not (total specialization).

We note that we discuss only the above constructs which are widely discussed in literature and CASE tools. We do not discuss more specialized constructs such as category or aggregation, which are discussed in only Elmasri and Navathe's book [1]. We also exclude ERD notations that has name of object-oriented. For the comparison of ERD and object-oriented notations, see Kushner, Song, and Whang [2], and for the comparison of various notations for object-oriented analysis, see Lind, Song, and Park [3]. We also exclude the variation of ERDs which are

modified to include object-oriented features, such as, complex entity relationship model [4] or ERC<sup>+</sup> model [5].

A variety of ERD notations has been developed to represent above concepts. Some of them allow n-ary relationships while others do not. Some notations allow attributes to be modeled in relationships. Some of them represent cardinality and participation constraints separately, while others use min/max notations by combining cardinality and participation constraints. Some of them specify the cardinality constraints across the relationship while others near the entity. Authors of database text books and CASE Tools use different ERD notations. These cause greater confusion and difficulty to novice database designers and users, and make the ER diagram less-transferable among authors, textbooks and CASE Tools. Hence, in this article we collected ten widely used ERD notations from various textbooks and CASE Tools. Based on our investigation, we compare/contrast them by the following seven points:

- 1) The way they allow n-ary relationships or not (see Section 2.1)
- 2) The way they represent cardinality and participation constraints or min/max notations (see Section 2.2)
- 3) The *place* they specify the constraints (see Section,2.3)
- 4) An attribute shown attached to a relationship,
- 5) Foreign keys modeled at the ERD level,
- 6) Overlapping and disjoint subclass entity types depicted, and
- 7) Complete and partial specialization

The ten selected methods are Chen [6], DDEW [7] or Teorey [8], Elmasri & Navathe [1], Korth & Silberschatz [9], McFadden & Hoffer [10], Batini, Ceri, & Navathe [11], Oracle CASE\*Methods [12], Information Engineering [13], IDEF1X used in ERWin [14], and Bachman [15, 16].

An example situation is described in order to discuss and illustrate each ERD technique.

The rest of this article is organized as follows: Section 2 introduces the definitions which need illustration, and Section 3 illustrates the ten ERD models that are depicted for the sample database problem. Section 4 summarizes and evaluates the differences of those ERD notations. Section 5 concludes our paper.

## 2 TERMINOLOGY

In this section, we illustrate the following three different points of ER diagrams:

- how they depict n-ary relationships in binary models;
- *where* they represent cardinality and participation constraints;
- how they represent cardinality and participation constraints.

### 2.1 Binary Models vs. N-ary Models

Some ERD methods are called Binary models, in that they allow only binary relationships and do not allow ternary or higher relationships. In binary models, every object that would have an attribute is considered an entity. Thus binary models do not allow an attribute in a relationship, and hence do not use a symbol, such as a diamond, to represent a relationship (see Figure 3(d)). In those binary models, one way to handle a ternary relationship is to convert it into an entity type. In binary models, a many-to-many relationship with at least one non-key attribute is also converted into an entity type.

A binary relationship exists when one instance of an entity can be associated with one instance of another associated entity. A ternary relationship exists when one instance of an entity can be associated with a pair of instances of the other two associated entities. These three entity instances must be associated at the same time in the ternary relationship. For example, the relationship BORROW among STUDENT, MAGAZINE, and BOOK in a library context cannot be modeled as a ternary relationship because a student does not have to borrow both a magazine and a book. We note that the interpretation of a ternary relationship is based on Teorey, Fry, & Yang [17]. In Figure 1(a), a pair of a PROJECT and a PART can be associated with P SUPPLIERS, a pair of a PROJECT and a SUPPLIER can be associated with N PARTS, and a pair of a PART and a SUPPLIER can be associated with M PROJECTS.

Figure 1 shows two ternary relationships and a set of binary relationships that simulate the ternary relationships. That is, a single ternary relationship is replaced by three one-to-many relationships. In Figure 1(a) SUPPLY is modeled as a ternary relationship and thus the identifier of the SUPPLY relationship is the combination of the identifiers of three participating entity types. In Figure 1(b) SUPPLY relationship is converted into an entity, and thus naturally SUPPLY entity

can have its own single-attribute identifier. The new entity is called the *intersection entity* or the *associative entity* or *Gerund* [18, 10]. Note that the new gerund always has many side cardinality, regardless of the cardinality of the original ternary relationship, as shown in Figure 1(b) and 1(d).

However, the semantics of a ternary relationship is not always the same as three binary relationships and the gerund [10]. For example, suppose we have many-to-many-to-one relationship as shown in Figure 1(c). That is, for a given pair of a PROJECT and a PART, there is only one SUPPLIER. In binary models, Figure 1(c) is represented as in Figure 1(d). Note that Figure 1(d) is identical to Figure 1(b). In Figure 1 (d), comparing with Figure 1(c), we lose the semantics that a PART used by a PROJECT has only one supplier. There are other differences between binary and ternary relationships [19, 20]. Jones and Song show that not every binary representations of ternary relationships are functional-dependency preserving [20]. This implies that n-ary models are semantically more powerful than binary models. Methods that allow n-ary relationships and those that allow only binary relationships are summarized in Table 1 in Section 4.

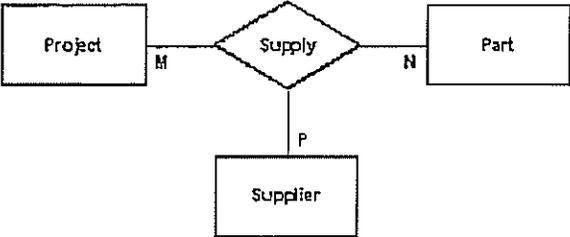


FIGURE 1 (a)  
m:n:p Ternary Relationship

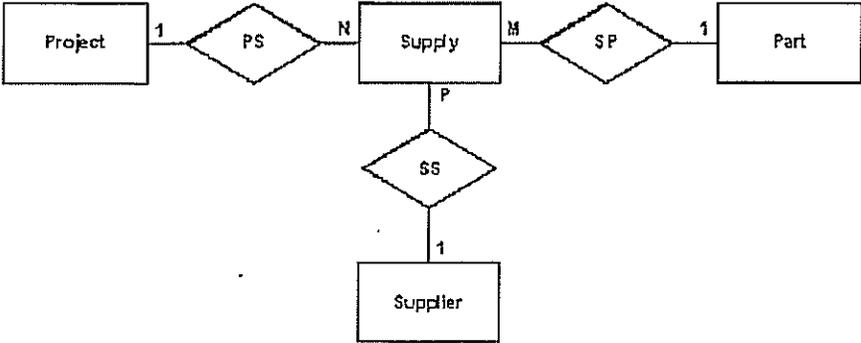


FIGURE 1 (b)  
Representing Fig 1 (a) in binary models

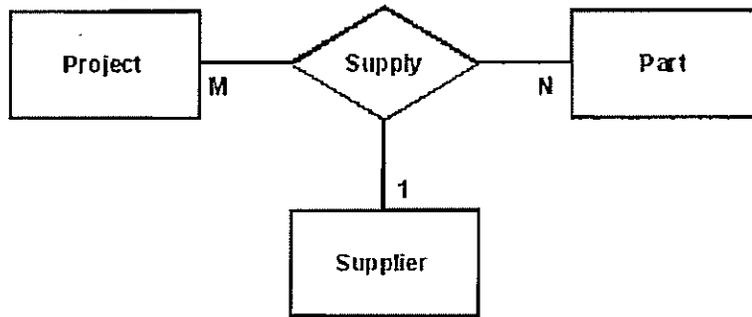


FIGURE 1 (c)  
m:n:1 Ternary Relationship

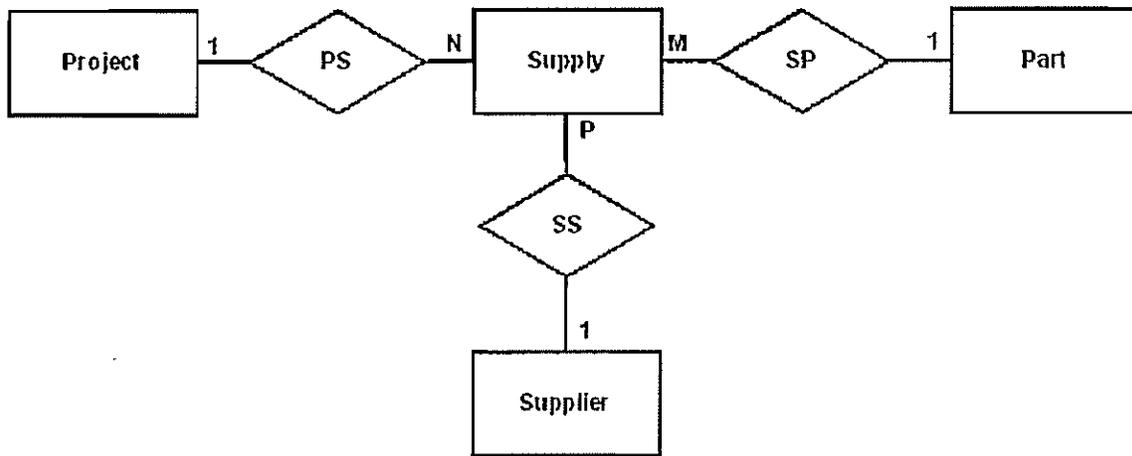


FIGURE 1 (d)  
Representing Fig 1 (c) in binary models

## 2.2 Look Across & Look Here Notations

To our knowledge, the terminology *Look Across* and *Look Here* was first used by Ferg [21] to refer to the place where the cardinality (maximum) or participation (minimum) constraints are specified in ER diagrams. The cardinality and participation constraints can be specified by looking across the relationship from the other direction or looking here first. The cardinality constraints in example (a) of Figure 2 shows Look Across notation and (b) shows Look Here

notation. In Look Across notation, the fact that one employee works for only one department is represented by placing 1 across the relationship WORKS FOR from EMPLOYEE entity. In Look Here notation, the fact, that one department can have many employees is specified by placing N across the relationship WORKS FOR from DEPARTMENT entity. Figure 1 uses Look Across notation. Section 4 summarizes the methods that use Look Across and Look Here conventions.

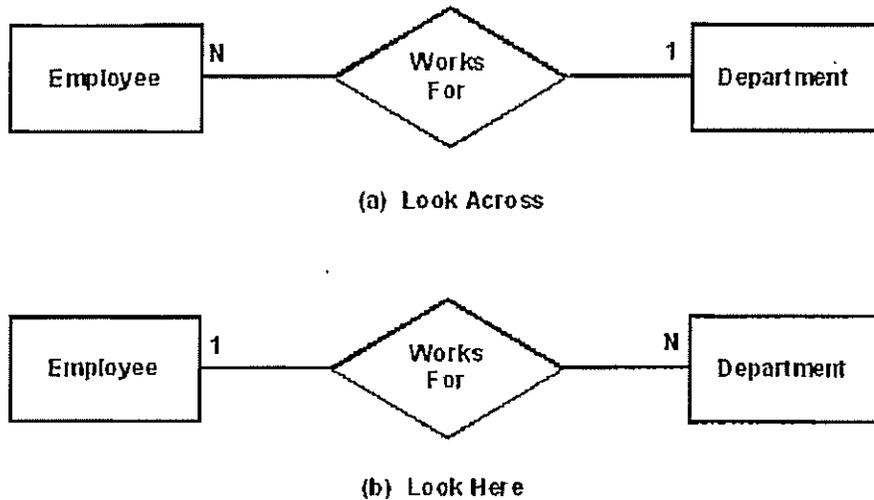


FIGURE 2  
Look Across and Look Here Notation

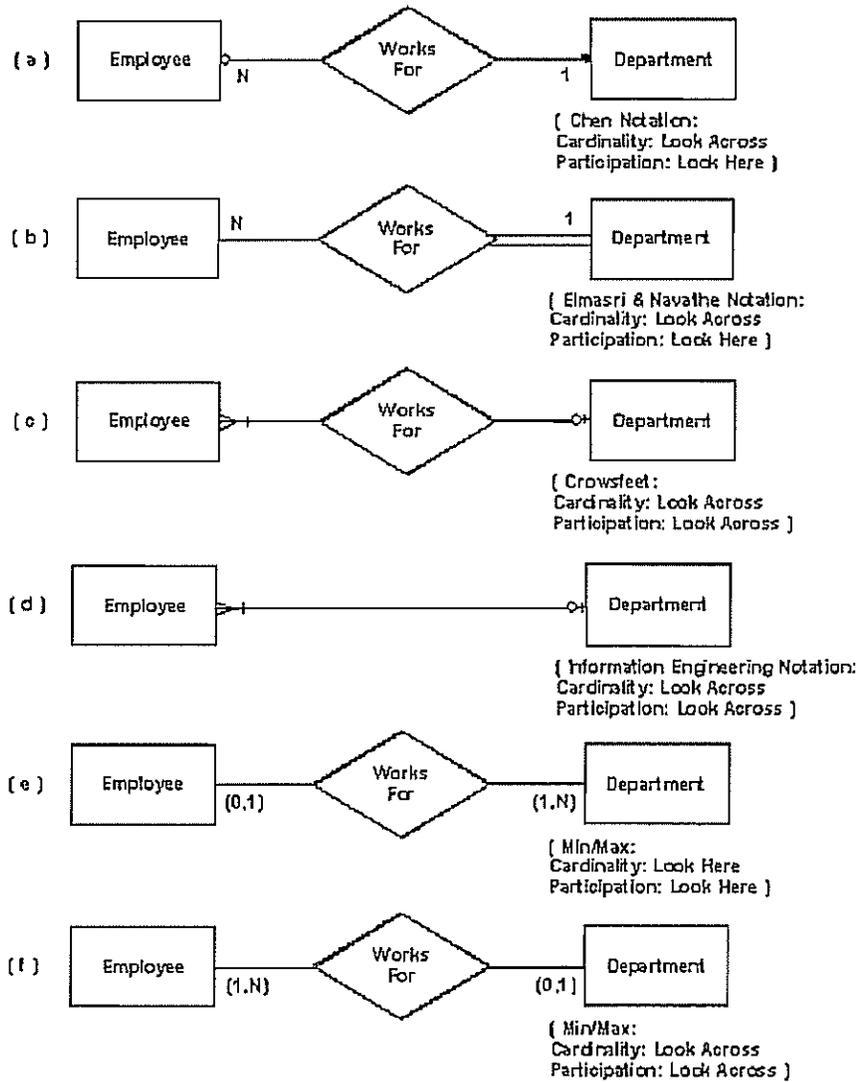
One binary relationship representing the following two sentences.  
"One employee works for only one department. One department can have many employees."

### 2.3 Cardinality & Participation Constraints

The cardinality constraint represents the *maximum* number of entity instances that may or must occur in order to participate in the relationship. The participation constraint represents the *minimum* number of entity instances that must occur in order to participate in the relationship. Thus, the participation constraint represents the total (mandatory) or partial (optional) existence of an entity instance as it relates to its relationship to another entity.

Figure 3 shows several popular ERD notations representing the cardinality constraint (one employee can work for one department and one department can have many employees) and the participation constraint (one employee can exist without working for a department (partial), but department cannot exist without having an employee (total)).

In Figure 3 (a) and (b), the cardinality constraints used Look Across notation, while the participation constraints used the Look Here notation.



**FIGURE 3**  
Various Notations for Cardinality and Participation Constraints showing "one employee can work for zero or one department and a department can have one or more employees."

In Figure 3(a), total participation is represented by a closed circle, while partial participation uses an open circle. In Figure 3(b), total participation is represented by a double line, while partial participation is represented by a single line. In Figure 3(c), the crowfoot

notation with the diamond [18] is used. In Figure 3(d), crowfoot notation without the diamond is used [13]. In Figure 3(e), (min, max) notation is used in the Look Here convention, and finally, in Figure 3(f), (min, max) notation is used in the Look Across convention [18].

In Figure 3(a) and (b), the cardinality and participation constraints are separated, while in 3(c), (d), (e), and (f) they are combined in (min, max) notation. See Ferg [21] for a more detailed discussion of various cardinality and participation constraints.

### 3 VARIOUS ERD NOTATIONS

In this section, we show the various notations of ERD used in different CASE Tools and text books. The problem description of the sample database is as follows:

The RESEARCH INSTITUTE database keeps track of its employees, departments and projects. The research institute is arranged by departments. Each department has a name and number. A department controls a number of projects. Each project has a name, number and project type. Each project is using zero or more parts supplied by any number of suppliers. One supplier can supply many parts to many projects, but must supply at least one part to a project. The research projects are subdivided into internal and external funded projects. Funded projects are subdivided by foundation and corporation. Each foundation and corporation associated with the institute is tracked by account. Each account stores a name, number, contract and account type. The employee's name, social security number and employee type are stored. An employee may be assigned to a department and may work on several projects, controlled by more than one department. The dependent's name and sex are stored for each employee. Most of the employees are subdivided into three major employee types- research, technical and secretary.

From the example described above, the following entity and relationship types are specified:

- Entity types are EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENT, SUPPLIER, PART and ACCOUNT.
- Relationship types are WORKS FOR, WORKS ON, DEPENDENT OF, CONTROLS, ORDERS and SPONSORS.

For the techniques of identifying entity types and relationship types, see Song and Froehlich [22]. The handling of attributes, generalization, participation and cardinality vary the most with the styles of each information modeling technique. The various modeling style techniques are described in the following sections.

### 3.1 Chen Notation

The entity relationship diagram was introduced by Chen in 1976 [6]. Figure 4 shows the example ERD using Chen's original notation with explicit notation for participation constraints. In this ERD, entities are represented by a box and relationship types are symbolized by a diamond. A double rectangle and a double diamond represent a weak entity type and a weak relationship, respectively. Attributes are represented by oval symbols. "Many" cardinality is indicated with the "N" near the entity's box, while a "1" indicates "one". Closed circles represent total participation and open circles represent partial participation.

The original Chen's notation [6] had notations only for entities, relationships, attributes and cardinality, but did not use generalization or participation constraint. Later Scheuermann, Schiffner, and Weber added generalization, aggregation, and participation constraints [23]. (In participation constraint, they use a closed circle for total participation, but do not use any notation for partial participation. We use an open circle to explicitly represent the partial participation.) The cardinality is represented by Look Across notation. The participation constraints uses Look Here notation. The ER Designer developed by Chen & Associates [18] supports this notation as well as (min, max) notation, as shown in Figure 3(f), and crowfoot notation, as shown in Figure 3(c). Entity identifiers are represented by double ellipses in Chen's notation [6].

### 3.2 Teorey Notation

Figure 5 shows the example ERD used in Teorey's notation [17, 8]. Even though his notation was first used in DDEW project [7], we call Teorey's notation since he popularized the notation through his articles. The entity is depicted by a box and assigned an unique name. Relationship type is depicted by a diamond with the name listed beside it. Cardinality is shown by shading the relationship diamond. The many side of the diamond is shaded while the one-side is not shaded.

Generalization connects the is-a relationships with hollow arrows. A weak entity is depicted with a box surrounded with double bars. The ternary relationship is depicted with three entities connected with a relationship diamond. ERDs in this notation do not always show attributes. Total participation is shown with a black dot or is not drawn and is the default syntax. Partial participation is shown with a hollow dot. This notation uses both cardinality and participation constraints using Look Across notation. Neither disjoint nor completeness constraints are supported. We could not find any example ER diagrams illustrating the concept of entity identifiers from Teorey's book [8]. We note that Teorey's new book [24] uses Chen's notation.

We observe that, when participation constraint is represented by Look Across notation, the participation constraint for ternary relationship cannot be properly represented. In Figure 4, PROJECT entity has a partial participation with ORDER relationship. In Figure 5, this partial participation cannot be properly represented in ORDER ternary relationship since there are two entities across PROJECT entity. This implies that in n-ary models, participation constraints must use Look Here convention. If we want to use Look Across convention for participation constraint, we must use the binary models.

### 3.3 Elmasri & Navathe Notation

Figure 6 shows the example ERD using Elmasri & Navathe's notation [1]. The entity is depicted by a box and it is assigned an unique name. Cardinality constraints are shown as 1 (one), N (many), or M (second relationship in many-to-many). A weak entity is depicted with a box surrounded with double bars. The ternary relationship is depicted with three entities. Attributes are shown with a labeled ellipses circle with a line drawn in the entity it belongs. The entity identifier is distinguished with a line drawn under the attribute's name. The notation distinguishes a single-valued attribute from a multivalued attribute, a composite attribute and a derived attribute. Multivalued attributes are shown with a double egg-like circle. A derived attribute is shown as a dotted ellipse circle. Total participation is shown with a double relationship line between an entity and its associated relationship, while partial participation is shown with a single line. Cardinality constraints use Look Across notation and participation constraints use Look Here notation.

In the generalization/specialization, both disjoint constraints and completeness constraint are fully represented. Disjoint subclasses are represented with a "d" symbol in a circle.

Overlapping subclasses are depicted with a connection between the superclass to the subclasses with a letter "o" symbol in a circle. An arc connects the circle to any type of subclass described above. Total specialization is represented by a double line and partial specialization is represented by a single line from the super class to the circle with either "d" or "o". Optionally, a discriminating attribute that classify subclasses can be shown. We note that Elmasri and Navathe discuss the notion of categorization which was first proposed by Elmasri, Weddreyer, and Hevner [25]. Categorization is a subclass built from two different superclasses. It is shown with a connection between the superclasses to the subclass with a U symbol in a circle. We do not discuss the Category in this article, since it is not supported by any other ERD methods discussed in this article. Among the ERD notations compared in this article, Elmasri and Navathe's notation is the most semantically rich in terms of modeling components and constraints.

### **3.4 Korth & Silberschatz Notation**

Figure 7 shows the example ERD using Korth & Silberschatz's notation [9]. Entity types are represented as rectangles. Attributes are symbolized as ellipses. Relationship types are represented as diamonds. Entities are linked with attributes with lines. Entity and relationship types are linked together with lines. Cardinality is distinguished between the entity and relationship either by a directed line (arrow) for one-side or an undirected line to represent many-side. Cardinality constraint uses Look Across notation. Generalization/specialization is shown with a triangle labeled with ISA. This joins the higher-level entity to the lower-level entity. While Generalization (which does not allow overlapping among subentity types) uses thick lines between the ISA triangle and each entity, specialization (which allow overlapping among subentity types) uses the regular thin lines. Participation is not depicted in Korth & Silberschatz's notation. Hence, the completeness constraint in a generalization hierarchy is not supported. There is no notation used for entity identifiers.

### **3.5 McFadden & Hoffer Notation**

Figure 8 shows the example ERD using McFadden & Hoffer's notation [10]. Entity types are represented as rectangles. Attributes are symbolized as ellipses. Relationship types are represented as diamonds. Entities are linked with attributes with lines. Entity and relationship types are linked together with lines. A cardinality constraint is represented by a bar for one-side and crowfoot for many cardinality. Ternary relationships are allowed in this method. A

participation constraints represented by "1" for total and "0" for partial. Both cardinality and participation constraints use Look Across notation using (min, max) form. Note that in this method we used a gerund to represent the ternary relationship ORDER. The reason is that a participation constraint in LOOK ACROSS convention cannot be represented in a ternary relationship. By converting the ternary relationship into a gerund, we can represent the participation constraint of the PROJECT entity.

Generalization hierarchy is shown with a round box labeled with ISA. This joins the superclass entity to the lower-level entity. Disjoint constraint is represented by an arc connecting lines to subclass entities. Partial specialization in this notation can be represented by adding an empty rectangle implying an undesignated subclass entity. Interestingly, McFadden & Hoffer [10] do not discuss the weak entity or dependent entity concept.

### 3.6 BATINI, CERI, and NAVATHE Notation

Figure 9 illustrates the example ERD in Batini, Ceri, and Navathe's notation style [11]. Entity types are represented as rectangles. Attributes are symbolized as small circles with a line connected to its entity and its attribute name labeled beside it. Primary key attributes are shown with the black circle while others are shown with an open circle. For entities with a composite primary keys, a line and black circle are drawn across those attributes that make up the primary key. Relationship types are represented as diamonds. Entities are linked with attributes with lines. Entity and relationship types are linked together with lines. Cardinality is indicated by the characters "0" (zero), "1" (one), and "N" (many). Participation and cardinality constraints are combined into the (min, max) form, such as (0,N) or (1,1), respectively. Look Here notation is used for both cardinality and participation constraints. Generalization and specialization are shown with a directed arrow that joins the lower-level entity to the higher-level entity. They do not distinguish between disjoint and overlapping among subentities in the hierarchy. Ternary relationships are allowed in this method. The interpretation of ternary relationships using Look Here notation needs elaboration. In Figure 9, a project can have minimum zero and maximum many orders; a supplier (part) have minimum one and maximum many orders. The ternary semantics of Batini, Ceri, and Navathe implies the cardinality when a ternary relationship is converted into a gerund. Note that this interpretation is different from what Chen, Teorey, and Elmasri & Navathe used. The latter used "for a pair of supplier and a part, there are zero or many

projects." We note that the weak entity notation is not directly represented as in other methods but they are implied by a composite primary key connected through two entity types.

### 3.7 Oracle's CASE\*METHOD Notation

Figure 10 shows the example ERD using the Oracle's CASE\*METHOD notation [12]. This method belongs to a binary model which does not allow a n-ary relationship and an attribute to be shown in a relationship. Hence, a relationship is just represented by a line. Entity type is represented as a box with the entity name capitalized and its attributes are listed below in lower case. Relationship type is shown as a line between associated entities. Cardinality constraints use Look Across and participation constraints use Look Here notation. Many cardinality is indicated with crowfoot at the end of the line. A single line represents one cardinality. The participation constraint is called optionally, and the term mandatory/optional is used instead of total/partial. Total participation is shown as a solid line while a dotted line indicates partial participation. Naming each end of the relationship reflects the participation and identifies the association between the entities. Optional attributes, whose value may be the null value, are illustrated by a small 'o' in front of the attribute name. Mandatory attributes, whose value is always required, are indicated by a small '\*' in front of the name. A unique identifier is the primary key which identifies each unique instance in the entity. The primary keys are represented with a '#' preceding the attribute that contributes to the identifier.

Subclass entity subtypes are shown as an inner box within the superclass entity type. Disjoint constraints and completeness constraints in a generalization hierarchy are not explicitly discussed in [12]. Oracle CASE\*METHOD supports mutually exclusive relationships between one entity and two relationships, and are shown with an arc with the black dot across the mutually exclusive relationship ends. In this situation, an entity instance can be associated with only one of the two mutually exclusive relationship. The mutually exclusive relationship notation can be used to simulate disjoint subclasses. For example, in Figure 10, we used an exclusive arc to represent the disjoint subclasses, as in Figure 6-3 of [12].

In Figure 10, we represented the ternary relationship by converting it into an entity type (called intersection entity) and adding a binary relationship between the intersection entity and other entities. Note that the notion of weak entity is not directly represented in Oracle CASE\*Method. Rather, it can be simulated by a bar and a little diamond. A bar in many-side

entity represents the situation that the primary key of one-side entity contributes the identifier of the many-side entity. The diamond represents the non-transferability, which means that the entity in many-side, once connected, cannot be reconnected to another entity in one side. This property is similar to existence dependency in typical ER modeling. Many-to-many relationships are allowed, but they are usually decomposed into two one-to-many relationships. Qualified limits of degree are represented by =, >, ≥, <, ≤ to define cardinality constraints.

### 3.8 Information Engineering Notation

Information Engineering (IE) method was originally developed by Martin & Finklestein. It was later revised by Martin [13]. Our discussion is mainly based on Martin's revised notation [13]. The IE method is also a binary method which does not allow a ternary relationship nor does it show attributes related to a relationship. Both cardinality and participation constraints are combined into min/max (bar and crowfoot) notation, and are represented with the Look Across convention.

Figure 11 shows the example ERD using the Information Engineering notation. Entity type is represented as a box. The relationship is shown as a line connecting two associated entities and given a name. Cardinality is depicted as follows:

one and only one	Two bars at end of line or single bar
zero or one	Hollow dot and one bar
one or more	One bar and crowfoot
zero, one or more	Hollow dot and crowfoot
more than one	Crowfoot.

The relationship between mutually exclusive entity types is represented with a black dot (See Figure 11). Entity subtypes are created when they have different associations to other entity types. Entity subtypes are shown in inner boxes within the super class entity type and subdivided by solid lines. The solid line represents disjoint subclasses. Overlapping subclasses can be represented by using a dashed line between two subclasses. A blank subtype box indicates that there are other subtypes not shown on the entity-relationship diagram, showing a partial specialization. When the specialization hierarchy is complex, a decomposition diagram can also be used. Instead of using inner boxes for subclasses, they are modeled as rectangles (entities)

outside the superclass and connected by lines. Disjoint subclasses are represented by a black dot. This is illustrated in FOUNDATION and CORPORATE subclasses in Figure 11. An open circle is added near the black dot when not every superclass entity instance participates in one of the subclasses, showing a partial specialization.

The popular CASE tools using the IE notation are IEF [26] and ADW [27]. In ADW, they use the term *Fundamental* entity for regular entity in other methods, the *Associative* entity for a relationship-converted entity, and the *Attributive* entity for dependent which serves to describe another entity type. In the IE method, attributes are not usually directly shown on the ER diagram, but they are entered into an data dictionary. As previously stated, the Information engineering method is a binary-modeling technique. So, this method does not allow a relationship to have an attribute. Attributes belonging to one-to-many relationships are modeled under the many-side entity type. When a many-to-many relationship has at least one descriptive attribute, the relationship is modeled as an entity type. ADW call this new entity type an associative entity, and adds a diamond inside the rectangle. (See WORKS\_ON in Figure 11). Note that in this case, the associative entity always has a many side. Since IE does not allow a ternary relationship either, the ORDER relationship in the sample ERD was represented as an associative entity in Figure 11. In the IE method, as in Oracle CASE\*Method, weak entities are not directly represented. However, the identifier dependency can be shown in IEF [26]. In IEF, we can superimposes an I near the dependent entity to represent the fact that the identifier of the dependent entity is the combination of the partial key of the dependent entity and the identifier of the other side entity type (See DEPENDENT in Figure 11).

We note that IE notation shows the relationship names in both directions. A label above a horizontal line is used when the relationship is read from left to right. A label below a horizontal line is used when the relationship is read from right to left. In Figure 11, however, we did not use the two-way naming practice of the IE method in order not to create any additional labels.

### 3.9 IDEF1X Information Model Notation

Figure 12 shows the example ERD using the IDEF1X Information Model notation [14]. IDEF1X is a binary model which does not allow n-ary relationships or many-to-many relationships with non-key attributes. Thus, in IDEF1X, any object with at least one information-bearing attribute is modeled as an entity type. The regular entity type is called the *independent entity*. It is

represented by a closed box with the name of the entity at the top. The attributes of the entity are listed inside the box. The primary keys are listed in the top section of the box. The data (non-primary-key) attributes are noted in the bottom section of the box. Independent entities or parent entities are entities that do not depend on another entity for its identification. This is represented with a square cornered box. In IDEF1X, most relationships are either one-to-one, one-to-many or many-to-many relationships without non-key attributes. Whenever a many-to-many relationship has at least one non-key attributes, it is modeled as an entity type called an associative entity. A ternary relationship is also modeled as an associative entity as in Figure 1. *Dependent entity* or child entity depends on another entity for its identification. A dependent entity is represented by a round cornered box. IDEF1X attribute notation conventions are detailed below:

attribute(FK)	Foreign Key
role-name.attribute(FK)	Role name (new name for FK)
attribute(AKn)	Alternate key
attribute(IEn)	Inversion entry (non-unique access identifier)
group(c1,c2,c3)	Group attribute
attribute(fk1, fk2)(FK)	Unified FK.

Relationship notation is subdivided into *identifying* and *non-identifying* associations between entities. An identifying relationship is a relationship in which all primary key attributes of the parent entity become part of the primary key attributes of the child entity. This simulates the notion of the weak entity of the ER model. A non-identifying relationship is a relationship in which the primary key of the parent entity does not become part of the primary key of the child entity but a foreign key in the child entity. A identifying relationship is shown as a solid line connecting entities while a non-identifying relationship is depicted by a dotted line.

In IDEF1X, the cardinality constraint and participation constraint are combined into min/max constraint style. These min/max constraints are represented using the Look Across notation. Both graphical symbols and textual notations are used to represent the min/max constraints. The single line, either solid or dotted, represents EXACTLY ONE. The closed dot represents ZERO OR MORE. The closed dot with P near the dot represents ONE OR MORE. The N represents EXACTLY N. The Z near the dot represents ZERO OR ONE.

Note that IDEF1X distinguishes between identifying and nonidentifying relationships.

Identifying relationships always begin with cardinality exactly one as in EMPLOYEE entity to DEPENDENT entity in Figure 12, since the primary key of the parent entity always become a part of the primary key of the child entity. However, in nonidentifying relationships, the primary key of the parent entity does not become a part of the primary key of the child entity. Instead, it simply becomes a foreign key on the child entity. Thus, the parent entity may or may not participate in the relationship with the child entity. For this problem, IDEFIX used a little diamond to represent optional participation of ZERO OR ONE. This is illustrated in DEPARTMENT entity, which means that an employee can have zero or one department.

IDEFIX can distinguish between overlapping and disjoint subentities in a generalization. It can also distinguish between a complete and an incomplete classification of subentities. Overlapping is represented by multiple classification lines from the super entity (e.g., EMPLOYEE and PROJECT in Figure 12), while disjoint is represented by a single line from the super entity (e.g., FUNDED PROJECT). A single line underneath a circle specifies a partial specialization (meaning that not all categories are shown), while double line specifies an complete specialization (meaning that all categories are shown). IDEFIX directly models foreign keys at the ERD level. This notation is used in ERWin CASE Tool.

### 3.10 Bachman Notation

Figure 13 shows the example ERD using the Bachman Case tool of Bachman's notation [16]. Bachman's method is also a binary model. An entity is represented by a box. The relationship is depicted as a line connecting the associated entities. The relationship is given a phrase to describe the association at both ends of the line. Cardinality constraints use Look Across notation and participation constraints use Look Here notation. Cardinality is shown by an arrow for many and a single line for one. An open circle at the end of a relationship shows optional participation between any pair of instances of associated entities. A filled-in or black circle indicates a mandatory relationship between any pair of instances of the entities. When a many-to-many relationship does not have a non-key attribute, the relationship is represented as a line. When a many-to-many relationship has a non-key attribute, it is modeled as an entity type. The roles of attributes are annotated in front of their names as follows:

PK	Primary key
FK	Foreign key

- PFK Primary key and Foreign key
- I Inherited attribute from the superclass entity

Note that in Figure 12, a ternary relationship was represented in two steps. First, many-to-many relationship between SUPPLIER and PART was modeled as an associative entity SUPPLIED\_PART. Then there is many-to-many relationship between PROJECT and SUPPLIED-PART.

In Bachman notation, a subclass is represented as an inner box within the superclass. The notation, however, does not represent disjoint or completeness constraints in a specialization hierarchy. Gane [12] also uses the same notation as Bachman. However, Gane represents mutually exclusive relationships by connecting each subentity type with an arc. Entity subtypes are shown within inner boxes. Bachman's method, as in IDEF1X, directly models foreign keys at the ERD level. A little diamond near the arrow (many side) represents the fact that the primary key of the one-side entity is used as a foreign key in the many-side entity.

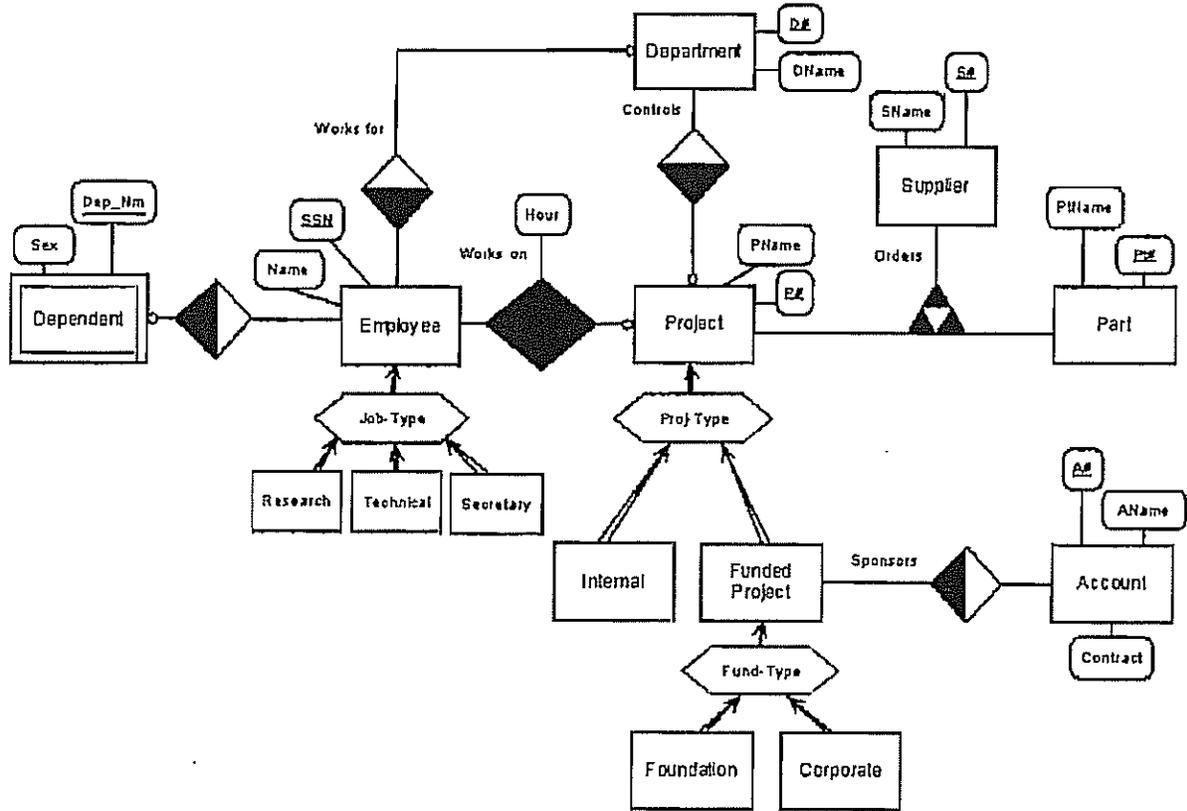


FIGURE 5: TEOREY's Notation

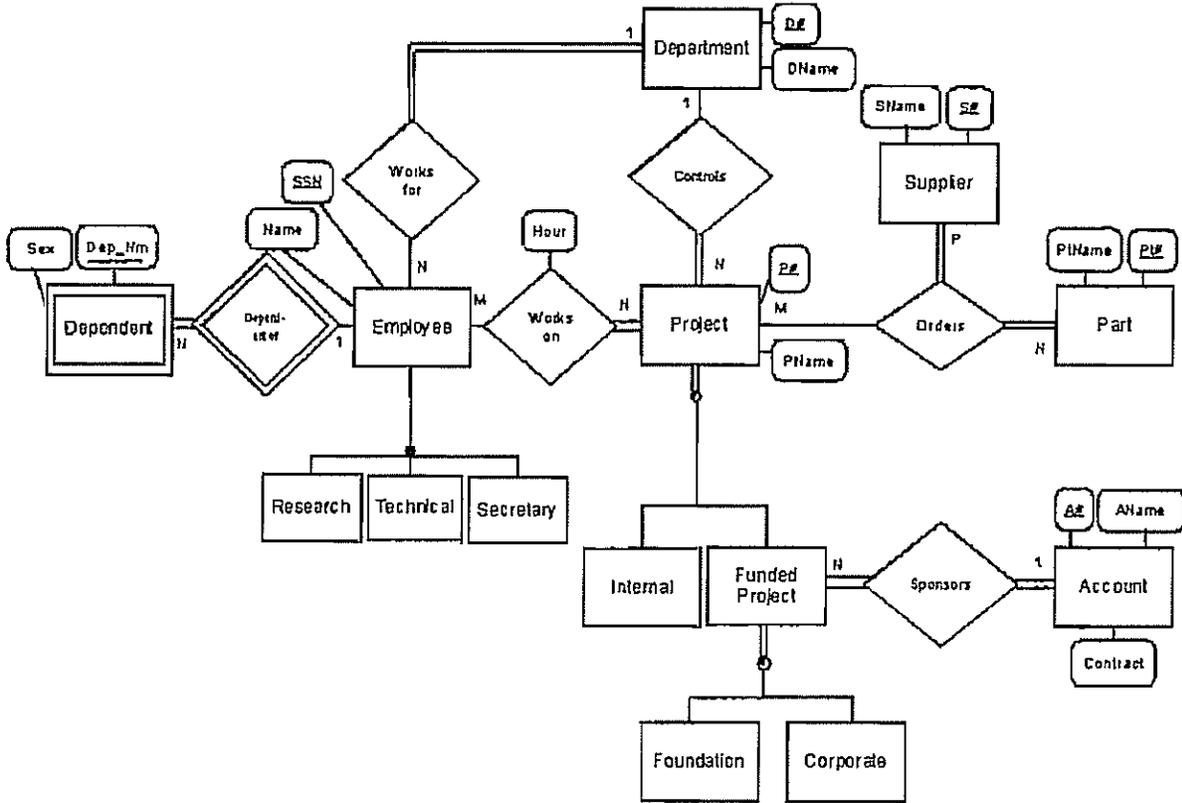


FIGURE 6: ELMASRI & NAVATHE's Notation

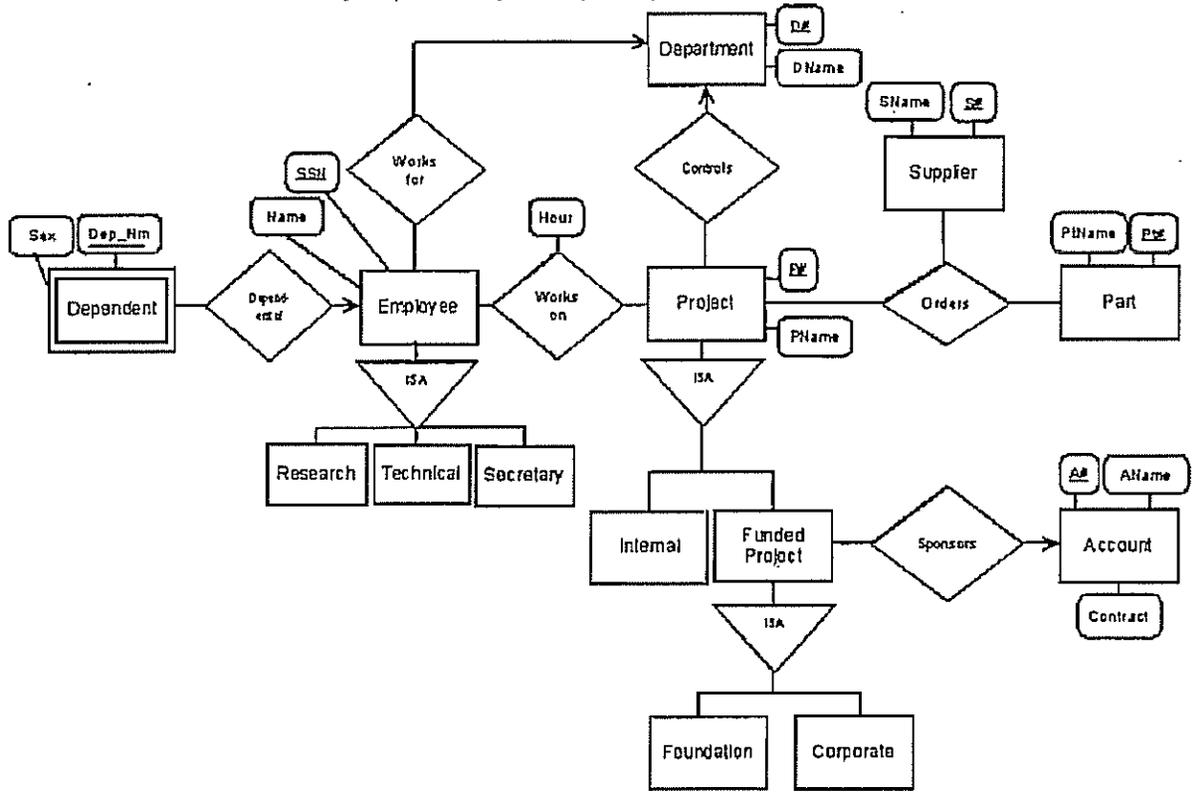


FIGURE 7: KORTH's Notation

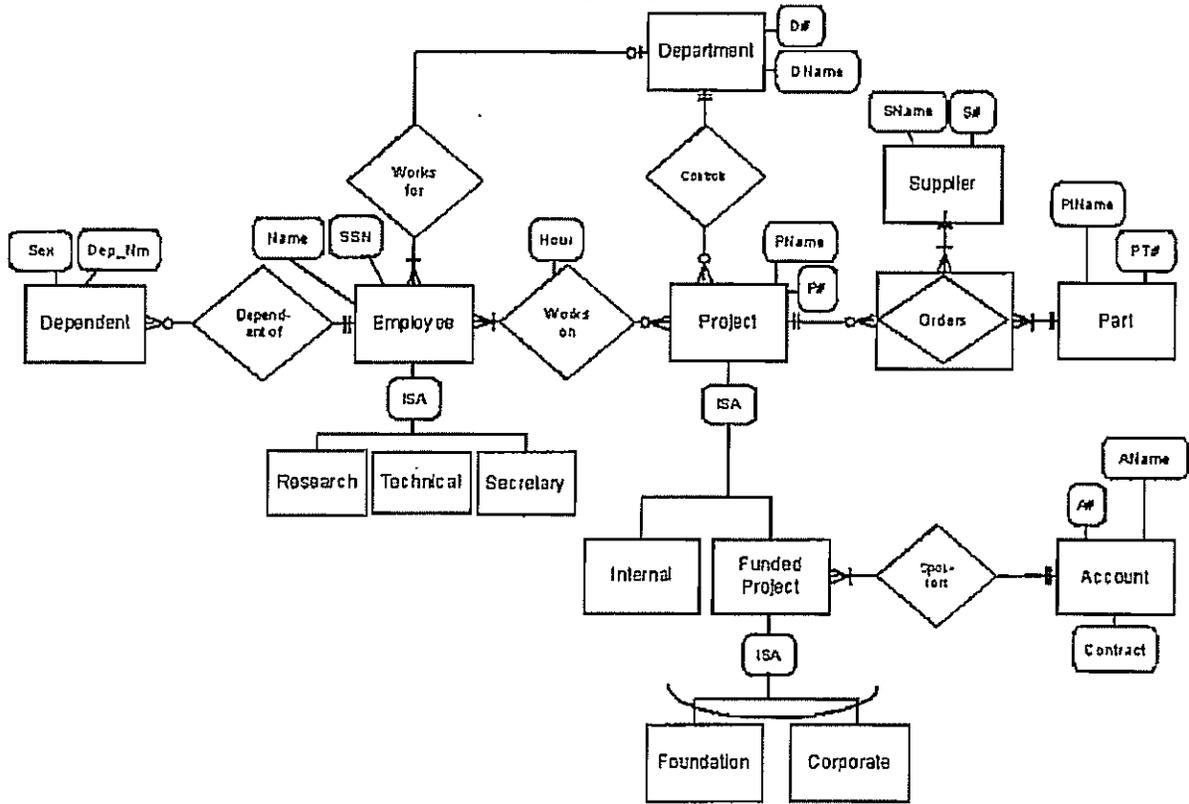


FIGURE 8: McFADDEN & HOFFER's Notation

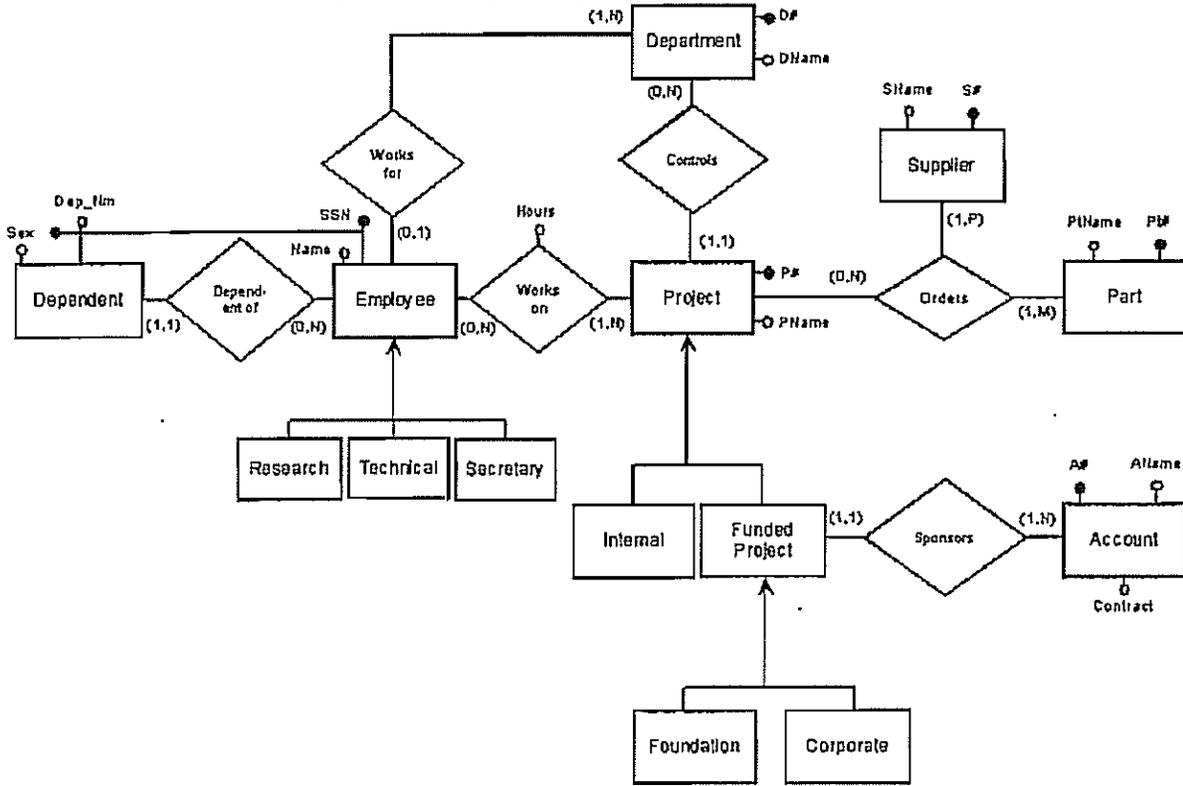


FIGURE 9: BATINI, CERI, & NAVATHE's Notation

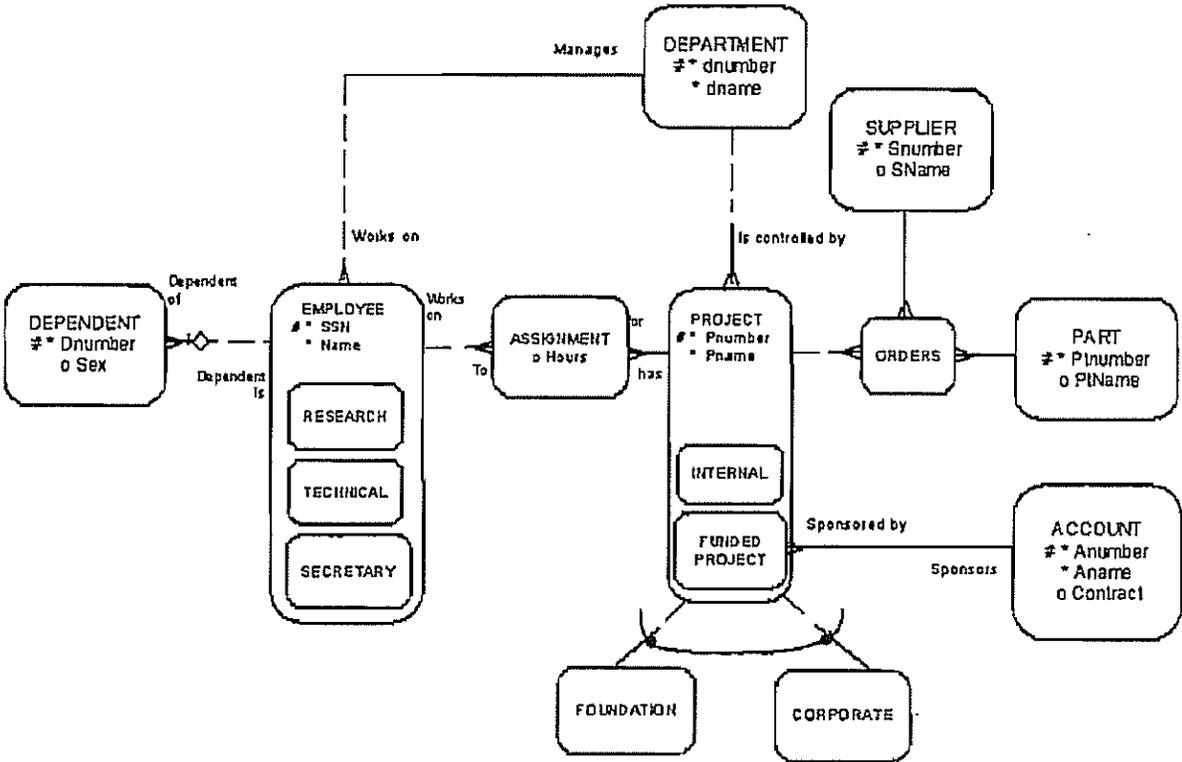


FIGURE 10: ORACLE's CASE METHOD Notation

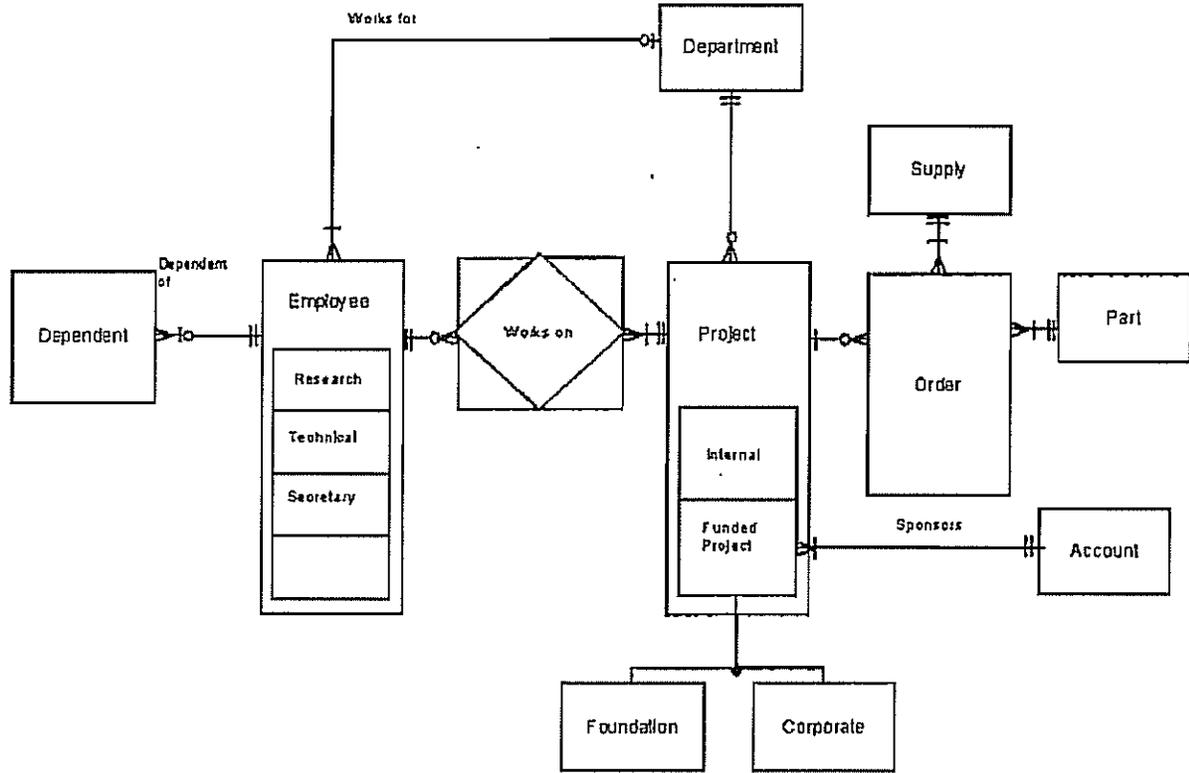


FIGURE 11: INFORMATION ENGINEERING Notation

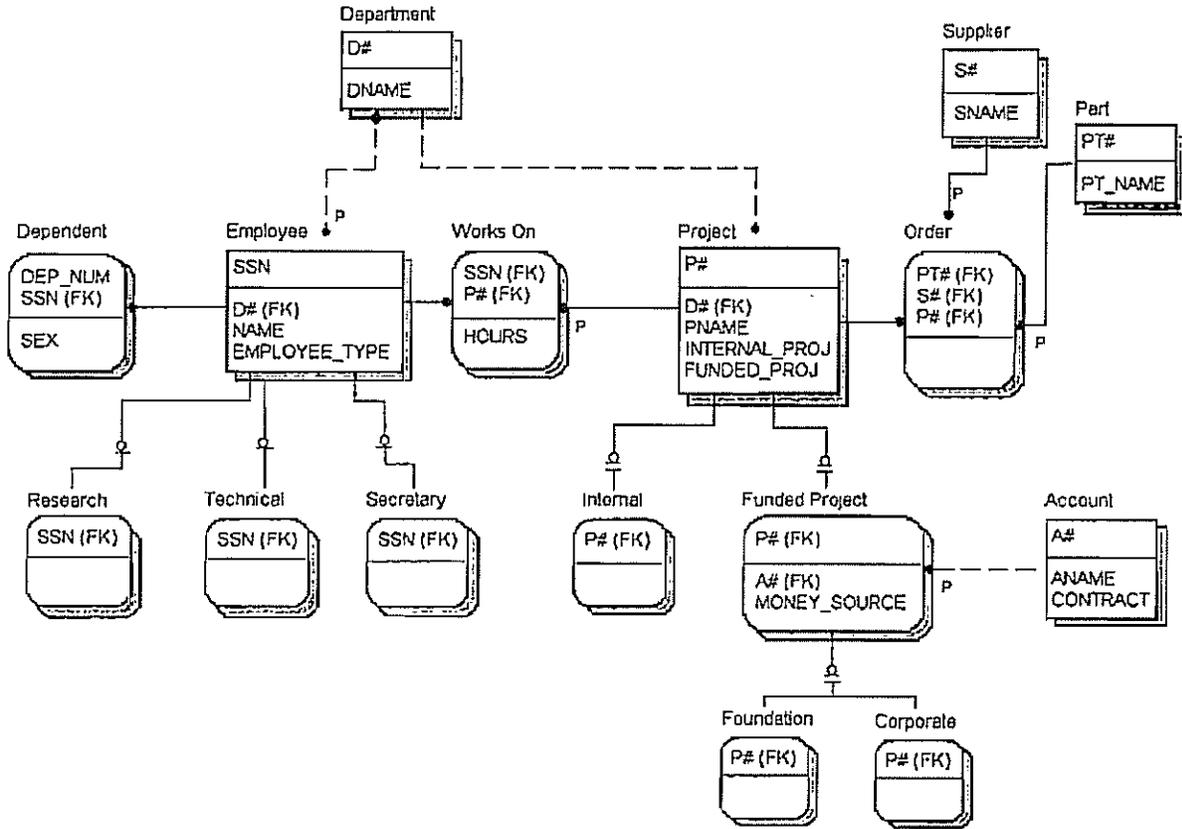


FIGURE 12: ERwIn/ERX 2.0 Notation

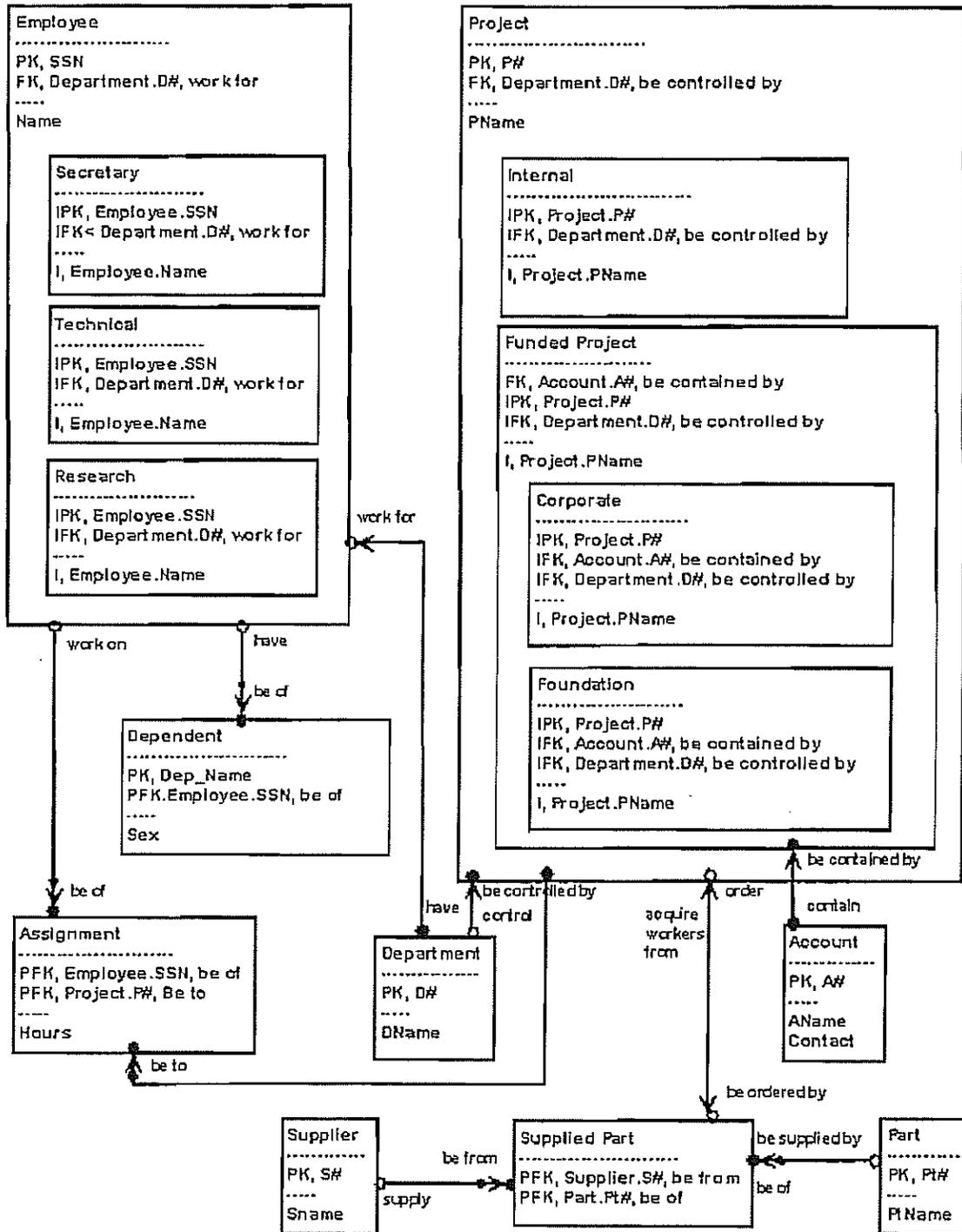


FIGURE 13: BACHMAN Notation

## 4 ANALYSIS OF NOTATIONS

In Section 4.1, we summarize features of ten ERD methods discussed in this paper. We further analyze those features in Section 4.2 and discuss the limitations of CASE tools using ERDs in Section 4.3.

### 4.1 Summary of ERD Methods

The criteria we used for comparing ERD methods include binary or n-ary relationships, relationships with or without attributes, cardinality & participation constraints, Look Across and Look Here notations, disjoint and completeness constraints in generalization/specialization, and the direct modeling of foreign keys at the ERD level. Table I classifies N-ary models from binary relationships. Table II summarizes the various ways of representing cardinality and participation constraints. Table III distinguishes ERD methods that model foreign keys at the ERD level. Table IV summarizes all the features in detail.

<b>N-ary</b>	Chen; Teorey; Elmasri & Navathe; Korth & Silberschatz; McFadden & Hoffer; Batini, Ceri, & Navathe
<b>Binary</b>	Oracle CASE*Methods; Information Engineering; IDEF1X; Bachman

**Table I Binary versus N-ary Notations**

Cardinality and Participation Constraints can be represented as (Min, Max) notation.	
(Min, Max) Look Here	Batini, Ceri, & Navathe.
(Min, Max) Look Across	Teorey <sup>1</sup> ; McFadden & Hoffer; Information Engineering; IDEF1X.
Participation Constraints: Look Here Cardinality Constraints: Look Across	Chen; Elmasri & Navathe; Oracle CASE*Method; Bachman.

Cardinality Constraints: Look Across No participation constraint notation	Korth & Silberschatz.
--	-----------------------

**Table II: Cardinality & Participation versus (Min,Max) Constraints**

<sup>1</sup> Total participation is not shown.

<b>No Foreign Key at the ERD level</b>	Chen; Teorey; Elmasri & Navathe; Korth & Silberschatz; McFadden & Hoffer; Batini, Ceri & Navathe; Oracle CASE*Method; Information Engineering.
<b>Modeling Foreign Key at the ERD level</b>	IDEF1X, Bachman.

**Table III: Modeling Foreign Key at the ERD level**



## **4.2 Analysis of ERD Methods**

We found that most ERD methods used in textbooks and CASE tools can be clearly classified as either binary models or n-ary models.

The following characteristics summarize the binary models examined:

- Any object with an information-bearing attribute becomes an entity,
- Ternary relationships are not allowed,
- Attributes in a relationship are not allowed,
- Symbols (e.g., a diamond) are not used for a relationship,
- Many-to-many relationships are allowed at an earlier analysis stage, but are encouraged to be decomposed into two one-to-many relationships,
- Many-to-many relationships with non-key attributes and ternary relationships are converted into entity types called intersection entities or associative entities.

The characteristics of n-ary models, most of all, are natural and allow direct modeling of ternary relationships and many-to-many relationships. For example, when a many-to-many or a ternary relationship does not need a unique identifier, we don't have to create an artificial entity as in binary models. As discussed in Section 2.1, the binary models have at least two weaknesses. The first one is that it cannot represent the semantics of ternary relationships correctly when the ternary relationships are not many-to-many-many. The second one is that not every binary representation of ternary relationships are functional-dependency preserving [20]. Rigorous analysis of binary relationships and ternary relationships can be found in Song & Jones [19, 20] and Jones & Song [28]. The two advantages of binary models are (1) the distinction between entities and relationships is clear since any object with at least one descriptive attribute is an entity; (2) the distinction between binary and ternary is simpler since there is no ternary relationships.

We also found that there are a variety of notations for cardinality and participation constraints. We found that ERD methods that uses Look Across convention for participation constraints cannot correctly represent semantics of a ternary relationship. See the discussion on Teorey's and McFadden and Hoffer's. This problem can be solved by converting a ternary relationship into a gerund.

ERD methods that do not directly model foreign keys at the ERD level need an extra step to convert ERDs to a relational schema. ERD methods that directly model foreign keys at the ERD level need more effort at the analysis stage, but they can be readily converted into a relational schema.

One notation cannot be forced over another. Each notation offers their own advantages and disadvantages. Many variations exist of the same modeling method and it is useful to know how to convert from one notation to another. In order to convert from one notation to another, less powerful method must be extended by concepts and notations. However, the semantics of the similar constructs must be interpreted carefully and documented by additional constraints. For example, the notion of weak entity is supported in Chen's and Elmasri and Navathe's notations. The weak entity not only implies ID dependency (the primary key of the weak entity is the combination of the primary key of parent entity and partial key of the weak entity), but also supports existence dependency (whenever an instance of the parent entity *s* is removed, the associated weak entity instances must be removed). The IE and IDEFIX methods only support ID dependency, which may or may not incur existence dependency. Oracle CASE\*METHOD supports both ID dependency and non-transferability, which can be considered to be similar to the notion of weak entity. The decision about what notation to use must be decided based on knowledge of the data modeler for the selected notation, corporate modeling history, and the availability of CASE tools. However, the pattern of many organizations is to stay with one ERD methodology because of the investment in CASE software, application development and training of systems personnel.

### **4.3 CASE Tools for ERD Methods**

Most CASE tools supporting data modeling still mainly supports diagram editing and at most the identification of cardinality constraints. They still lack the ability to check the correctness of the diagram at the semantics of application domains. For example, they do not give any clue whether a ternary relationship or a set of binary relationships must be used. They do not identify any redundant relationships, either, and does not optimize ERDs. For these problems, most CASE tools rely on the knowledge of data modeler. CASE tools also need to support more diverse notations semantics for flexibility

## 5 CONCLUSION

In this article, we compared ten different notations for ER diagrams which are widely used in database textbooks and CASE tools for modeling and designing relational databases. According to our investigation, we found that ERDs differ based on whether they allow n-ary relationships; whether they allow attributes in a relationship; where and how they represent cardinality and participation constraints; how they depict overlapping and disjoint subclass entity types; and whether they model foreign keys at the ERD level. The result of these comparisons were summarized in the section above. Each diagram was explained and illustrated using a common problem domain.

Some areas that need more research in ER modeling include the development of more modeling heuristics, the identification and removal of redundant relationships, optimization of ERDs, optimal use of specialization hierarchy (now this is in the realm of object-oriented database design), and objective measures of quality of ERDs (see [29] for example). This issue must be discussed in the context of the various cardinality and participation constraints and related integrity constraints.

From the mid-70's through the 1980's new entity-relationship methodologies offered semantic solutions to the shortcomings of previous methodologies. With the saturation of ER modeling techniques in the research community, new methods are not as enthusiastically received unless the modeling designer proves how his new method provide more semantic power. Extensions to the entity-relationship diagram continue to evolve to include new symbols to model object-oriented concepts. Some of them are allowed to have non-atomic attributes for modeling complex objects [4, 5]. Some of them are extended to include new semantics to model object oriented concepts, such as methods, operations, and messages [30]. This only re-enforces the flexibility and expressive power of this modeling technique.

### *Acknowledgments*

Authors would like to thank you many students of Drexel University who generously provided several iterations of the diagrams used in this article, including Ed. Forbes for Bachman Notation James McNeil for five diagrams, and Xin Sun for reformatting the references.

## References

1. R. Elmasri, S. Navathe, *Fundamentals of Database Systems*. 2nd ed., Benjamin/Cummings, Redwood City, CA., 1993.
2. Michael Kushner, Il-Yeol Song, and Kyu-Young Whang, "A Comparative Study of Three Object-Modeling Methodologies," *Systems Development Management*, 1994, 34-03-40 (1-22). (Also in *Data Base Management*, 26-01-10).
3. Janet Lind, Il-Yeol Song, and E.K. Park, "Object-Oriented Analysis: A Study in Diagram Notations," *Journal of Computer and Software Engineering*, Vol. 3, No. 1 (Winter 1995), pp. 133-165.
4. K. R. Dittrich, W. Gotthard, and P. Lockemann, "Complex Entities for Engineering Applications", in *Proceedings of the International Conference on the ER Approach*, North-Holland, (1987), pp. 421-440.
5. C. Parent and S. Spaccapietra, "About entities, complex objects and object-oriented data models," in *Information System Concepts: An In-depth Analysis*, ED.. Falkenberg and P. Lindgreen (eds.), North-Holland, 1989, pp. 193-223.
6. P. P-S. Chen, "The entity-relationship model-toward a unified view of data," *ACM Transactions on Database Systems*, 1,1 (March 1976), pp. 9-36.
7. D. Reiner, M. Brodie, and G. Brown, et al. (eds.). "The Database design and evaluation workbench (DDEW) project at CCA." *Database Engineering*, 7,4 (1985).
8. T. J. Teorey, *Database Modeling and Design: The Entity-Relationship Approach*. Morgan Kauffmann, San Mateo, CA. 1991.
9. H. Korth and A. Silberschatz, *Database System Concepts*. 2nd ed., McGraw-Hill, New York, N.Y., 1991.
10. F. McFadden and J. Hoffer, *Modern Database Management*. Benjamin/Cummings

11. C. Batini, S. Ceri, and S. Navathe, *Concatual Database Design: an Entity- Relationship Approach*. Benjamin/Cummings Publishing, Redwood City, CA., 1992.
12. R. Barker, *CASE\*METHOD<sup>TM</sup>: Entity Relationship Modeling*. Addison-Wesley Publishing Company, New York, New York, 1990.
13. J. Martin, *Information Engineering: Planning & Analysis, Book II*. Prentice-Hall, Englewood Cliffs, NJ, 1990.
14. T. Bruce, *Designing Quality Databases with IDEF1X Information Models*. Dorset House Publishing, New York, New York, 1992.
15. C. Gane, *Rapid System Development: Using Structured Techniques and Relational Technology*. Prentice Hall, Englewood Cliffs, N.J., 1989.
16. Bachman, *Bachman Analyst*, Bachman Information Systems Incorporated. 1992.
17. T. J. Teorey and J. Fry & Yang, "A logical design methodology for relational databases using the extended entity-relationship model," *ACM Computing Survey*, 18,2 (June 1986), pp197-222.
18. P. P-S. Chen, *The ER Designer: Reference Manual*. Chen & Associates, 1987.
19. I. Y. Song and T. J. Jones, "Analysis of binary relationships within ternary relationships in ER Modeling," In *Proc. of the 12th International Conference on Entity-Relationship Approach*, Dallas, TX., Dec. 15-17, 1993, pp. 265-276.
20. T. Jones and I.-Y. Song, "Binary Representations of Ternary Relationships in ER Conceptual Modeling," in *14th International Conference on Object-oriented and Entity-Relationship Approach*, Gold Coast, Australia, Dec. 12-15, 1995.
21. S. Ferg, "Cardinality concepts in entity-relationship modeling," in *Proceedings of 10th*

*Journal of Computer and Software Engineering*, Vol. 3, No.4 (1995), pp. 427-459  
*International Conference on Entity-Relationship Approach*, (Oct. 23-25, 1991, San Mateo, CA) T. Teorey (editor), pp. 1-30.

22. Il-Yeol Song and Kristin Froehlich, "Entity-Relationship Modeling: A Practical How-to Guide," *IEEE Potentials*, Vol. 13, No. 5, Dec/Jan 1994-1995, pp. 29-34.
23. P. Scheuermann, G. Scheffner, and H. Weber, "Abstraction capabilities and invariant properties modeling within the ERA," In *the Proceedings of the 1st International Conference on Entity-Relationship Approach*. P. Chen (Editor), North-Holland, Elsevier, Netherlands, 1980, pp. 121-140.
24. T. J. Teorey, *Database Modeling and Design: The Fundamental Principles*, 2nd ed., Morgan Kaufmann, San Francisco, CA, 1994.
25. R. Elmasri, T. Weddreyer, and A. Hevner, "The Category Concept: an extension to the entity-relationship model," *Data and Knowledge Engineering*, 1,1, (June 1985), pp75-116.
26. *IEF Technology Overview*, Texas Instrument, 1990.
27. ADW Case Tool Seminar, KnowledgeWare, 1991.
28. T. J. Jones and I. Y. Song, "Binary Imposition Rules and Ternary Decomposition", *The Proc. of InfoScience '93*, Oct. 21-23, 1993, Seoul, Korea, pp. 267-274.
29. D. L. Moody and G. G. Shanks, "What Makes a Good Data Model? Evaluating the Quality of Entity Relationship Models," in *Proc. of 13th International Con. on Entity-Relationship Approach*, pp. 94-111, 1994.
30. G. Gorman and J. Chovbinch, "The Object-oriented entity-relationship model (OOERM). *Journal of Management Information Systems*, 7,3, (Winter 1990-1991), pp41-65.

# Entity-Relationship Modeling: Historical Events, Future Trends, and Lessons Learned

Peter P. Chen

Computer Science Department  
Louisiana State University  
Baton Rouge, LA 70803, USA  
E-mail: [pchen@lsu.edu](mailto:pchen@lsu.edu)

**Abstract.** This paper describes the historical developments of the ER model from the 70's to recent years. It starts with a discussion of the motivations and the environmental factors in the early days. Then, the paper points out the role of the ER model in the Computer-Aided Software Engineering (CASE) movement in the late 80's and early 90's. It also describes the possibility of the role of author's Chinese culture heritage in the development of the ER model. In that context, the relationships between natural languages (including Ancient Egyptian hieroglyphs) and ER concepts are explored. Finally, the lessons learned and future directions are presented.

## 1 Introduction

Entity-Relationship (ER) modeling is an important step in information system design and software engineering. In this paper, we will describe not only the history of the development of the ER approach but also the reactions and new developments since then. In this perspective, this paper may be a little bit different from some other papers in this volume because we are not just talking about historical events that happened twenty or thirty years ago, we will also talk about the consequences and relevant developments in the past twenty-five years. At the end, we will talk about lessons learned during this time period. In particular, we intend to show that it is possible that one concept such as the ER concept can be applied to many different things across a long time horizon (for more than twenty-five years) in this fast-changing Information Technology area.

This paper is divided into 8 sections. Section 1 is the Introduction. In Section 2, the historical background and events happened around twenty-five years ago will be explained. For example, what happened at that time, what the competing forces were, and what triggered researchers like the author to work on this topic will be explained. Section 3 describes the initial reactions in the first five years from 1976 to 1981. For example, what the academic world and the industry viewed the ER model initially? Section 4 states the developments in the next twenty years from 1981 to 2001. In particular, the role of the ER model in the Computer-Aided Software Engineering (CASE) will be discussed. Section 5 describes a possible reason for the author to come up with the ER modeling idea, that is, the author's Chinese culture heritage. The author did not think about this particular reason until about fifteen years ago. Section 6 presents our view of the future of ER modeling. Section 7 states the lessons learned. For those of you who have similar experience in the past twenty-five years, you probably have recognized similar principles and lessons in this section. For those who just started their professional careers recently, we hope the lessons learned by the author will be helpful to those readers. Section 8 is the conclusion.

## 2 Historical Background

In this section, we will look at the competing forces, the needs of the computer industry at that time, how the ER model was developed, and the main differences between the ER model and the relational model.

### 2.1 Competing Forces

First, Let us look at the competing forces in the computer software area at that time. What are the competing forces then? What triggered people like the author to work on this area (data models) and this particular topic (ER modeling)? In the following, we will discuss the competing forces in the industry and in the academic world in the early 70's,

**Competing Forces in the industry.** There were several competing data models that had been implemented as commercial products in the early 70's: the file system model, the hierarchical model (such as IBM's IMS database system), and the Network model (such as Honeywell's IDS database system). The Network model, also known as the CODASYL model, was developed by Charles Bachman, who received the ACM Turing Award in 1973. Most organizations at that time used file systems, and not too many used database systems. Some people were working on developing better data or index structures for storing and retrieving data such as the B+-tree by Bayer and McGreight [1].

**Competing Forces in the Academic World.** In 1970, the relational model was proposed, and it generated considerable interest in the academic community. It is correct to say that in the early 70's, most people in the academic world worked on relational model instead of other models. One of the main reasons is that many professors had a difficult time to understand the long and dry manuals of commercial database management systems, and Codd's relational model paper [2] was written in a much more concise and scientific style. For his contributions in the development of the relational model, Codd received ACM Turing Award in 1981.

*Most People were working on DBMS Prototypes.* Many people at that time in the academic world or in the industry worked on the implementation of database management system prototypes. Most of them were based on the relational model.

*Most Academic People were investigating the definitions and algorithms for the Normal Forms of Relations.* A lot of academic people worked on normalization of relations because only mathematical skills were needed to work on this subject. They could work on the improvement of existing algorithms for well-defined normal forms. Or, they could work on new normal forms. The speed of research moved very fast in the development of normal forms and can be illustrated by the following scenario. Let us say that several people were ready to publish their results on normal forms. Assuming that one person published a paper on 4<sup>th</sup> normal form and another person who had written a paper on 4<sup>th</sup> normal form but had not published it yet, the 2<sup>nd</sup> person would have changed the title of the paper from 4<sup>th</sup> normal form to 5<sup>th</sup> normal form. Then, the rest would work on the 6<sup>th</sup> normal form. This became an endless game till one day somebody wrote a paper claiming that he had an infinity-th normal form and arguing that it did not make any sense to continue this game. Most practitioners also said loudly that any relational normal form higher than 3<sup>rd</sup> or 4<sup>th</sup> won't have practical significance. As a result, the game of pursuing the next normal form finally ran out of steams.

### 2.2 Needs of the System Software in the Early 70's

**The Needs of the Hardware/Software Vendors.** In terms of software vendors at that time, there were urgent needs for (1) integration of various file and database formats and (2) incorporating more "data semantics" into the data models.

**The Needs of the User Organizations.** For user organizations such as General Motors and Citibank, there were urgent needs for (1) a unified methodology for file and database design for various file and database system available in the commercial market and (2) incorporation of more data semantics including business rules into the requirements and design specifications.

### 2.3 How the ERM was Developed

Here, we will give some personal history of the development of the ER model: where the author was and what the author did in the early 70's, particularly on how the author developed the ER model.

**Harvard (Sept. '69 to June '73).** After the author got a B.S. in Electrical Engineering from National Taiwan University in 1968, the author received a fellowship to study Computer Science (at that time, it was a part of Applied Mathematics) at Harvard graduate school. The author received the Ph.D. degree in 1973. The thesis was very mathematically oriented – focusing on the file allocation problems in a storage hierarchy using the queuing theory and mathematical programming techniques. The knowledge the author learned in EE, CS and applied math was crucial in the development of the ER model in subsequent years.

**Honeywell and Digital (June '73 to August '74).** The author joined Honeywell Information Systems in Waltham, MA in June '73. He participated in the “next-generation computer system” project to develop a computer system based on distributed system architecture. There were about ten people in the team, and most of them were at least twenty years senior than the author. The team consisted of several well-known computer experts including Charles Bachman. One of the requirements of such a “distributed system” was to make the files and databases in different nodes of the network compatible with each other. The ER model was motivated by this requirement. Even though the author started to crystallize the concepts in his mind when he worked for Honeywell, he did not write or speak to anyone about this concept then. Around June of 1974, Honeywell abandoned the “next-generation computer system” project, and all the project team members went different ways. The author then spent three months at Digital Equipment Corporation in Maynard, MA to develop a computer performance model for the PDP-10 system.

**MIT Sloan School of Management (1974 – 1978).** In September 1974, the author joined MIT Sloan School of Management as an Assistant Professor. This was the place that he put the ER ideas down into an article. Being a professor in a business/management school provided the author many opportunities to interact with the user organizations. In particular, he was particularly impressed by a common need of many organization to have a unified methodology for file structure and database design. This observation certainly influenced the development of the ER model. As a result, the first ER paper was first presented at 1st International Conference on Very Large Databases in 1975 and subsequently published in the first issue of ACM Transactions on Database Systems [3] in March of 1976.

#### 2.4 Fulfilling the Needs

How did the ER model fulfill the needs of the vendor and user organizations at that time? We will first start with the graphical representation and theoretical foundations of the ER model. Then, we will explain the significant differences between the ER model and the relational model.

**The Concepts of Entity, Relationship, Types, and Roles.** In Fig. 1, there are two entities; both of them are of the “Person” type. There is a relationship called, “is-married-to,” between these two persons. In this relationship, each of these two Person entities has a role. One person plays the role of “husband,” and another person plays the role of “wife.”

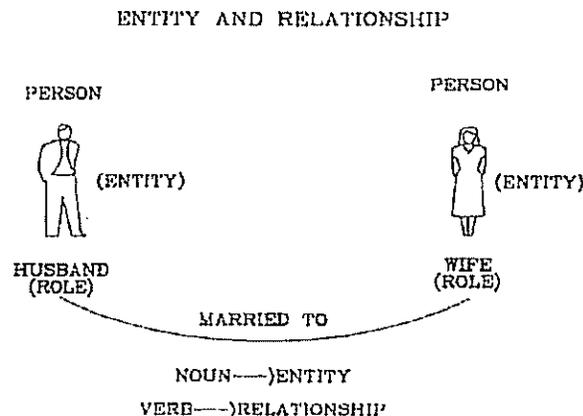


Fig. 1. The Concept of Entity and Relationship

**The Entity-Relationship (ER) Diagram.** One of the key techniques in ER modeling is to document the entity and relationship types in a graphical form called, Entity-Relationship (ER) diagram. Figure 2 is a typical ER diagram. The entity types such as EMP and PROJ are depicted as rectangular boxes, and the relationship types such as WORK-FOR are depicted as a diamond-shaped box. The value sets (domains) such as EMP#, NAME, and PHONE are depicted as circles, while attributes are the "mappings" from entity and relationships types to the value sets. The cardinality information of relationship is also expressed. For example, the "1" or "N" on the lines between the entity types and relationship types indicated the upper limit of the entities of that entity type participating in that relationships.

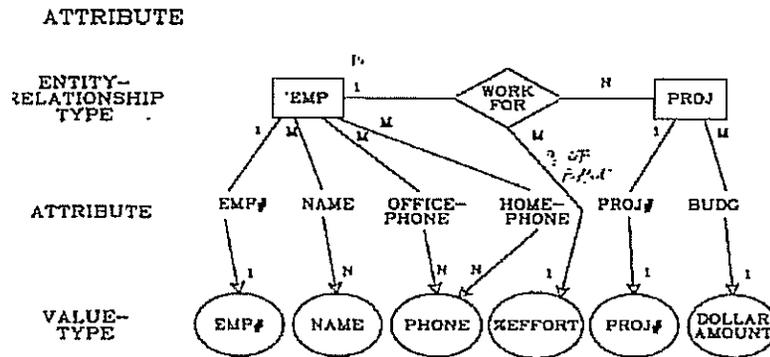


Fig. 2. An Entity-Relationship (ER) Diagram

**ER Model is based on Strong Mathematical Foundations.** The ER model is based on (1) Set Theory, (2) Mathematical Relations, (3) Modern Algebra, (4) Logic, and (5) Lattice Theory. A formal definition of the entity and relationship concepts can be found in Fig. 3.

**SET THEORY (DEFINITIONS)**

ENTITY	$e$
ENTITY SET	$E; e \in E$
VALUE	$v$
VALUE SET	$V; v \in V$
RELATIONSHIP	$r$
RELATIONSHIP SET	$R; r \in R$

A RELATIONSHIP SET IS DEFINED AS A 'MATHEMATICAL RELATION' ON ENTITY SETS

$$R = \{r_1, r_2, \dots, r_n\}$$

$$r_i = [e_1, e_2, \dots, e_n] | e_1 \in E_1, \dots, e_n \in E_n$$

Fig. 3. Formal Definitions of Entity and Relationship Concepts

**Significant Differences between the ER model and the Relational Model.** There are several differences between the ER model and the Relational Model:

*ER Model uses the Mathematical Relation Construct to Express the Relationships between Entities.* The relational model and the ER model both use the mathematical structure called Cartesian product. In some way, both models look the same – both use the mathematical structure that utilizes the Cartesian product of something. As can be seen in Figure 3, a relationship in the ER model is defined as an ordered tuple of "entities." In the relational model, a Cartesian product of data "domains" is a "relation," while in the ER model a Cartesian product of "entities" is a "relationships." In other words, in the relational model the

mathematical relation construct is used to express the "structure of data values," while in the ER model the same construct is used to express the "structure of entities."

*ER Model Contains More Semantic Information than the Relational Model.* By the original definition of relation by Codd, any table is a relation. There is very little in the semantics of what a relation is or should be. The ER model adds the semantics of data to a data structure. Several years later, Codd developed a data model called RM/T, which incorporated some of the concepts of the ER model.

*ER Model has Explicit Linkage between Entities.* As can be seen in Figures 2 and 4, the linkage between entities is explicit in the ER model while in the relational model is implicit. In addition, the cardinality information is explicit in the ER model, and some of the cardinality information is not captured in the relational model.

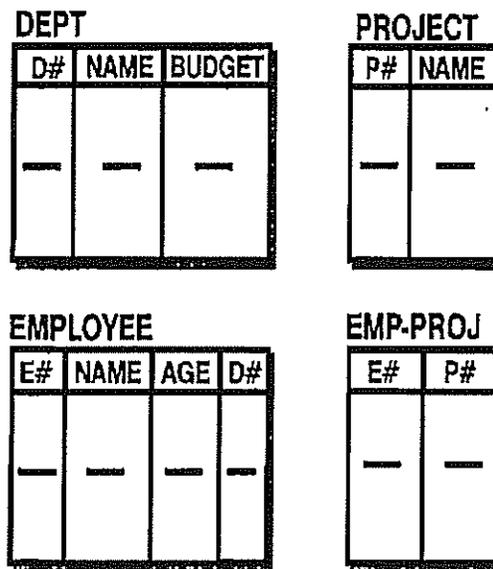


Fig. 4. Relational Model of Data

### 3. Initial Reactions & Reactions in the First Five Years (1976 – 1981)

#### 3.1 First Paper Published & Codd's Reactions

As stated before, the first ER model paper was published in 1976. Codd wrote a long letter to the editor of ACM Transaction on Database Systems criticizing the author's paper. The author was not privileged to see the letter. The editor of the Journal told the author that the letter was very long and single-spacing. In any case, Dr. Codd was not pleased with the ER model paper. Ironically, several years later, Codd proposed a new version of the relational data model called RM/T, which incorporated some concepts of the ER model. Perhaps, the first paper on the ER model was not as bad as Codd initially thought. Furthermore, in the 90's, the Codd and Date consulting group invited the author to serve as a keynote speaker (together with Codd) several times in their database symposia in London. This indicates that the acceptance of ER model was so wide spread so that initial unbelievers either became convinced or found it difficult to ignore.

#### 3.2 Other Initial Reactions and Advices

During that time, there was a "religious war" between different camps of data models. In particular, there was a big debate between the supporters of the Relational model and that of the Network model. Suddenly, a young assistant professor wrote a paper talking about a "unified data model." In some sense, the author

was a “new kid on the block” being thrown into the middle of a battle between two giants. The advice the author got at that time was: “why don't you do the research on the n-th normal form like most other researchers do? It would be much easier to get your normal form papers published.” That was an example of the type of advices the author got at that time. Even though those advices were based on good intentions and wisdom, the author did not follow that type of advices because he believed that he could make a more significant contribution to the field by continuing working on this topic (for example, [4-13]). It was a tough choice for a person just starting the career. You can imagine how much problems or attacks the author had received in the first few years after publishing the first ER paper. It was a very dangerous but a very rewarding decision the author made that not only had a significant impact on the author's career but also the daily practices of many information-modeling professionals.

### **3.3 IDEF, ICAM, and Other Believers**

There were a small but growing number of believers of the ER or similar data models. For example, Mike Hammer, who was an Assistant Professor at the EECS department of MIT, developed the Semantic Data Model with his student, Dennis McCleod. Later on, Hammer applied the idea in reverse engineering in the IT field to organization restructuring and became a management guru. Outside of the academic world, the industry and government agencies began to see the potential benefits of ER modeling. In the late 70's, the author served as a consultant in a team that developed the data modeling methodology for the ICAM (Integrated Computer-Aided Manufacturing) project sponsored by the U.S. Air Force. One of the objectives was to develop at least two modeling methodologies for modeling the aircraft manufacturing processes and data: one methodology for process modeling and one for data modeling. The data modeling methodology was called IDEF1 methodology and has been used widely in US military projects.

### **3.4 Starting a Series of ER Conferences**

The first ER conference was held in UCLA in 1979. We were expecting 50 people, but 250 to 300 people showed up. That was a big surprise. Initially, the ER conference was a bi-annual event, but now it is an annual event being held in different parts of the world [14]. In November of this year (Year 2001), it will be held in Japan [15], and next year (Year 2002) it will be held in Finland. This series of conferences has become a major annual forum for exchanging ideas between researchers and practitioners in conceptual modeling.

## **4 The Next Twenty Years ('81 –'01)**

### **4.1 ER Model Adopted as a Standard for Repository Systems and ANSI IRDS.**

In the 80's, many vendors and user organizations recognized the need for a repository system to keep track of information resources in an organization and to serve as the focal point for planning, tracking, and monitoring the changes of hardware and software in various information systems in an organization. It turned out that the ER model was a good data model for repository systems. Around 1987, ANSI adopted the ER model as the data model for Information Resource Directory Systems (IRDS) standards. Several repository systems were implemented based on the ER model including IBM's Repository Manager for DB2 and DEC's CDD+ system.

### **4.2 ER Model as a Driving Force for Computer-Aided Software Engineering (CASE) tools and Industry**

Software development has been a nightmare for many years since the 50's. In the late 80's, IBM and others recognized the needs for methodologies and tools for Computer-Aided Software Engineering (CASE). IBM proposed a software development framework and repository system called, AD Cycle and the Repository Manager that used the ER model as the data model. The author was one of the leaders who actively preached the technical approach and practical applications of CASE. In 1987, Digital Consulting Inc. (DCI) in Andover, Mass., founded by Dr. George Schussel, organized the 1<sup>st</sup> Symposium on CASE in Atlanta and invited the author to be one of the two keynote speakers. To everybody's surprise, the symposium was a huge commercial success, and DCI grew from a small company to a major force in the symposium and trade show business.

### 4.3 Object-Oriented (OO) Analysis Techniques are Partially Based on the ER Concepts

It is commonly acknowledged that one major component of the object-oriented (OO) analysis techniques are based on the ER concepts. However, the "relationship" concept in the OO analysis techniques are still hierarchy-oriented and not yet equal to the general relationship concept advocated in the ER model. It is noticeable in the past few years that the OO analysis techniques are moving toward the direction of adopting a more general relationship concept.

### 4.4 Data Mining is a Way to Discover Hidden Relationships

Many of you have heard about data mining. If you think deeply about what the data mining actually does, you will see the linkage between data mining and the ER model. What is data mining? What does the data mining really is doing? In our view, it is a discovery of "hidden relationships" between data entities. The relationships exist already, and we need to discover them and then take advantage of them. This is different from conventional database design in which the database designers identify the relationships. In data mining, algorithms instead of humans are used to discover the hidden relationships.

## 5 In Retrospect: Another Important Factor – Chinese Culture Heritage

### 5.1 Chinese Culture Heritage

Many people asked the author how he got the idea of the Entity-Relationship model. After he kept on getting that kind of questions, the author thought it might be related to something that many people in Western culture may not have. After some soul searching, the author thought it could be related to his Chinese culture heritage. There are some concepts in Chinese character development and evolution that are closely related to modeling of the things in the real world.

Here is an example. Figure 5 shows the Chinese characters of "sun", "moon, and "person". As you can see, these characters are a close resemblance of the real world entities. Initially, many of the lines in the characters are made of curves. Because it was easier to cut straight lines on oracle bones, the curves became straight lines. Therefore, the current forms of the Chinese characters are of different shapes.

<u>Original Form</u>	<u>Current Form</u>	<u>Meaning</u>
		Sun
		Moon
		Person

Fig. 5. Chinese Characters that Represent the Real-World Entities

Chinese characters also have several principles for "composition." For example, Figure 6 shows how two characters, SUN and MOON, are composed into a new character. How do we know the meaning of the new character? Let us first think: what does sun and moon have in common? If your answer is: both reflect lights, it is not difficult to guess the meaning of the new character is "brightness." There are other principles of composing Chinese characters [10].

日 (sun) + 月 (moon) = 明 (Bright/ Brightness by light)

Fig. 6. Composition of Two Chinese Characters into a New Chinese Character

What does the Chinese character construction principles have to do with ER modeling? The answer is: both Chinese characters and the ER model are trying to model the world – trying to use graphics to represent the entities in the real world. Therefore, there should be some similarities in their constructs.

### 5.2 Ancient Egyptian Hieroglyphs

Besides Chinese characters, there are other languages have graphic characters. Ancient Egyptian language is one of them. It turns out that there are several characters in ancient Egyptian characters are virtually the same as the Chinese characters. One is "sun", another is "mouth, and the third one is "water." It is amazing that both the Egyptian people and the Chinese people developed very similar characters even though they were thousands of miles away and had virtually no communication at that time. Ancient Egyptian Hieroglyphs also have the concept of composition. Interested readers should refer to [11].

Hieroglyph	Meaning	Hieroglyph	Meaning
(a) 	lower arm	(f) 	man
(b) 	mouth	(g) 	woman
(c) 	viper	(h) 	sun
(d) 	owl	(i) 	house
(e) 	sieve	(j) 	water

Fig. 7. Ancient Egyptian Hieroglyphs

## 6 The Future

### 6.1. XML and ER Model.

In the past few years, the author has been involved in the developing the "standards" for XML. He has participated in two XML Working Groups of the World Wide Web Consortium (W3C) as an invited expert. During this involvement, some similarities between XML and the ER model were discovered including the following:

**RDF and the ER Model.** There are several components in the XML family. One of them is RDF, which stands for Resource definition Framework. This is a technology that Tim Berners-Lee, the Director of W3C, pushes very hard as a tool for describing the meta-data in the web. There are some similarities and differences between RDF and the ER model, and Mr. Berners-Lee has written several articles discussing this issue. In a joint meeting of the RDF and Schema Working Groups over one year ago, they issued the Cambridge Communiqué [16] that states: "...RDF can be viewed as a member of the Entity-Relationship model family..."

**XLink and the ER model.** Most of us are familiar with the hyperlink in HTML. The XLink Working Group of W3C has been trying to do is to develop a new kind of hyperlink for XML. In HTML, the hyperlink is basically a "physical pointer" because it specifies the exact URL of the target. In XLink, the new link is one step closer to a "logical pointer." In the evolution of operating systems, we have been moving from physical pointers to logical pointers. The XLink Working Group proposed a new structure called, "extended link." For example, Fig. 8 is an extended link for five remote resources. The extended link concept in XML is very similar to the n-ary relationship concept in the ER model. Figure 8 can be viewed as a relationship type defined on 5 entity types.

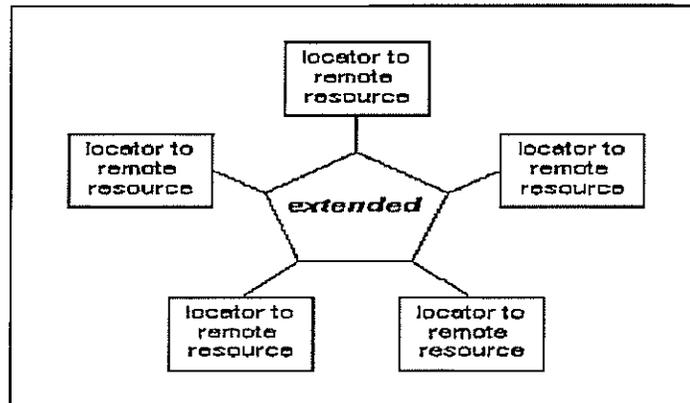


Fig. 8. "Extended Link" in XML is Similar to the N-ary Relationship Concept in the ER Model

## 6.2. Theory of the Web

One thing that is still missing today is the theory of the web. The ER model could be one of the foundations for the theory of the Web. The author plans to work on that topic and would encourage the readers to work on the subject, too.

## 7 Lesson Learned

### 7.1 Reflections on Career choices

In the past twenty-five years, the author made some tough career choices as some of the other authors in this volume did. It is the hope of the author that our experience will be useful to some other people who just started their professional careers and are making their career choices. Here are some reflections based on the author's own experience:

**Right idea, right place, right time, and belief in yourself.** In order to have your idea be accepted by other people, you need not only to have the right idea but also to present them at the right place and right time. You also need "persistence." In other words, you need to believe in yourself. This is probably the most difficult part because you have to endure some unnecessary pressures and criticisms when you are persistent on your idea and try to push it forward. Hopefully, some days in the future, you will be proved to be right. At that time, you will be happy that you have persisted.

**Getting Fresh Ideas from Unconventional Places.** After working on a particular area for a while, you may run out of "big" ideas. You may still have some "good" ideas to get you going, but those ideas are not "earth-breaking." At that time, you need to look for ideas in different subject areas and to talk to new people. For example, most of us are immersed in Western culture, and learning another culture may trigger new ways of thinking. Similarly, you may look into some fields outside of information technology such as Physics, Chemistry, Biology, or Architecture to find fresh ideas. By looking at the theories, techniques, and approaches used in other fields, you may get very innovative ideas to make a breakthrough in the IT field.

### 7.2 Implications of the Similarity and differences between the Chinese Characters and Ancient Egyptian Hieroglyphs on Software Engineering and Systems Development Methodologies

As we pointed out earlier, there are several Chinese characters that are almost the same as their counterparts in ancient Egyptian hieroglyphs. What does this mean? One possible answer is that human beings think alike even though there was virtually no communication between ancient Chinese people and ancient Egyptian people. It is very likely that the way to conceptualize basic things in the real world is

common to most of the races and cultures. As was discussed earlier, the construction and developments of other characters are different in Chinese and in Ancient Egyptian Hieroglyphs. It is valid to say that the language developments were dependent on the local environment and culture. What is the implication of the similarities and differences in character developments on the development of software engineering and information system development methodologies? The answer could be: some basic concepts and guidelines of software engineering and system development methodologies can be uniformly applied to all people in the world while some other parts of the methodologies may need to be adapted to local cultures and customs.

## **8. Conclusions**

The author was very fortunate to have the opportunity to meet the right people and to be given the opportunity to develop the Entity-Relationship (ER) model at the time and environment such a model was needed. The author is very grateful to many other researchers who have continued to advance the theory of the ER approach and to many software professionals who have practiced ER modeling in their daily jobs in the past twenty-five years. We believe that the concepts of entity and relationship are very fundamental concepts in software engineering and information system development. In the future, we will see new applications of these concepts in the Web and other new frontiers of the software world.

## References

1. Bayer, R. and McGreight, E., "Organization and Maintenance of Large Ordered Indexes," Acta Informatica, Vol. 1, Fasc. 3, 1972, pp. 173-189.
2. Codd, E. F. "The Relational Model of Data for Large Shared Data Banks," Comm. of the ACM, Vol. 13 (6), 1970, pp. 377-387.
3. Chen, P.P., "The Entity-Relationship Model: Toward a Unified View of Data," ACM Trans. on Database Systems, Vol.1, No.1, March 1976, pp. 1-36.
4. Chen, P. P., "An Algebra for a Directional Binary Entity-Relationship Model," IEEE First International Conference on Data Engineering, Los Angeles, April 1984, pp. 37-40.
5. Chen, P. P., "Database Design Using Entities and Relationships," in: S. B. Yao (ed.), Principles of Data Base Design, Prentice-Hall, NJ, 1985, pp. 174-210.
6. Chen, P. P., "The Time-Dimension in the Entity-Relationship Model," in: Information Processing '86, H. -J. Kugler (ed.), North-Holland, Amsterdam, 1986, pp. 387-390.
7. Chen, P. P. and Zvieli, A., "Entity-Relationship Modeling of Fuzzy Data," Proceedings of 2nd International Conference on Data Engineering, Los Angeles, February 1986, pp. 320-327.
8. Chen, P. P. and Li, M., "The Lattice Concept in Entity Set," in: Entity-Relationship Approach, S. Spaccapietra (ed.), North-Holland, Amsterdam, 1987, pp. 311-326.
9. Chandrasekaran, N., Iyengar, S.S., and Chen, P. P., "The Denotational Semantics of the Entity-Relationship Model," International Journal of Computer Mathematics, 1988, pp. 1-15.
10. Chen, P. P., "English, Chinese and ER Diagrams," Data & Knowledge Engineering, Vol. 23, No. 1, June 1997, pp. 5-16.
11. Chen, P. P., "From Ancient Egyptian Language to Future Conceptual Modeling," in: Conceptual Modeling: Current Issues and Future Directions, Chen, P.P., et al. (eds), Springer-Verlag, Berlin, Lecturing Notes in Computer Sciences, No. 1565, 1998, pp. 57-66.
12. Yang, A. and Chen, P. P., "Efficient Data Retrieval and Manipulation using Boolean Entity Lattice," Data & Knowledge Engineering, Vol. 20, 1996, pp.211-226.
13. <http://www.csc.lsu.edu/~chen/>
14. <http://www.er2000.byu.edu/>
15. <http://www.arislab.dnj.vnu.ac.jp/ER2001>
16. <http://www.w3.org/TR/schema-arch>

Documentation: Aqua Data Studio 6.5 - [HTML Documentation download, ZIP: ads-docs.zip \(Size: 84.7MB\) / ads-docs-no-shots.zip \(Size: 1.2MB\)](#)  
 New Features

- [Getting Started](#)
- [Installation](#)
- [JDBC Drivers](#)
  - Oracle 8i
  - Oracle 9i
  - Oracle 10g
  - DB2 7.2
  - DB2 8.1
  - DB2 9.0
  - SQL Server
  - Sybase ASE
  - Sybase Anywhere
  - Sybase IQ
  - Informix
  - PostgreSQL
  - MySQL
- [Memory Configuration](#)
- [Character sets](#)
- [Registering Servers](#)
  - MSDE 2000
  - SQL Express 2005
  - PostgreSQL SSL
- [Application](#)
  - Browser and Scripting
  - Visual Editing
  - SQL History and Archive
  - Shortcut Toolbar
  - Key Mappings
  - Key Mapping Assistant
  - Custom Keymap
  - Profiles
  - Options
  - Settings
  - New Frame
  - Window
  - Details View
  - Command Line
- [Query Window](#)
  - Basics
  - Toolbar
  - Server Side
  - Comments

**Tools - Entity Relationship The (ER) Diagram Generator**

Entity Relationship (ER) Diagram Generator helps users achieve a better understanding of their database schema by displaying the structure in a graphical format.

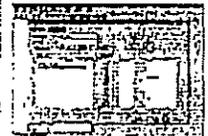
- Tools - Entity Relationship The (ER) Diagram Generator

To view an ER Diagram:

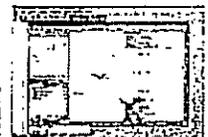
1. Launch the ER Diagram Generator dialog by:
  - o Selecting *Tools -> ER Diagram Generator* from the Menu Bar OR
  - o Selecting *Tools -> ER Diagram Generator* from the right-click pop-up menu on a database object in the Schema Browser
2. Select a Database and Schema (or *All Schemas*)
3. In the left panel, select one or more types of objects to be scripted (Tables, Views or Both).  
*Click the green check button to select all and the red X button to deselect all.*
4. All objects matching the selected schema & object types will appear in the right panel. Select one or more objects to be scripted.  
*Click the green check button to select all and the red X button to deselect all.*
5. Click *Next* to generate the diagram
6. Once the diagram is generated, the ER Diagram Viewer will open and display the new diagram.
7. To change the layout of the diagram, select an entry from the *Layout* dropdown list.  
 Supported layouts include:
  - o *Box*
  - o *Circle*
  - o *Hierarchical*
  - o *Tree*
  - o *Radial Tree*
  - o *Moer*
  - o *Fruchterman-Reingold (FR)*
  - o *Annealing*
  - o *Inverted Self Organising Map (ISOM)*
8. The *Entity View* drop-down list determines the amount of detail displayed for each entity in the diagram. Select *Full* to view all attributes or *Header* to view only the entity name.
9. To save an image of the current diagram, click *Save Image As*.



ER Diagram Menu



ER Diagram Generator



ER Diagram Window

SQL Automation  
Aqua Commands  
Morph to  
Delimited List  
Results  
Save Results  
Auto-completion  
Popup Menu  
SQL Formatter  
Permissions  
Parameterized  
Pivot Results  
(Grid and Chart)  
Grid Results (Grid  
and Chart)

#### Editors

SQL Editor  
Text Editor  
HTML Editor  
XML Editor  
Regular  
Expressions

#### Image Viewer

#### Visual Explain Plan

Basics  
Explain Plan  
Explain Diagram  
Whiteboard

#### Procedure/Package Editor

#### SQL Debugger

Sybase Debugger

#### Tools

Table Data Editor  
Schema Script  
Generator  
Server Script  
Generator  
ER Diagram  
Query Builder  
Import Tool  
Export Tool  
Execution Monitor

#### Compare Tools

Schema Compare  
Tab Compare  
Directory  
Compare  
File Compare  
Copy History  
Compare  
Results Compare

#### Oracle DBA Tools

Instance Manager  
Storage Manager  
Rollback Manager  
Log Manager  
Security Manager

10. To print a hard-copy of the current diagram, click *Print*.
11. Click on an entity to display more column & constraint detail in the lower left panel.

#### ER Diagram Generator Enhancements:

- General
  - ER Diagram: Indexed columns are now identified with an icon in the diagram.
  - ER Diagram Generator Dialog - Added "Reverse Selection" to object selection, to reverse the current selection

Session Manager  
SGA Manager  
Server Statistics

**SQL Server DBA Tools**

Instance Manager  
Storage Manager  
Security Manager  
Session Manager  
SQL Agent  
Manager

**Sybase DBA Tools**

Instance Manager  
Storage Manager  
Security Manager  
Session Manager

**Source Control**

Subversion  
CVS

---

[Home](#) • [Screenshots](#) • [Downloads](#) • [Documentation](#) • [Support](#) • [Licensing](#) • [Contact](#)

Copyright © 2001-2007 AquaFold, Inc. All Rights Reserved