Journal of Computer and Software Engineering, Vol. 3, No.4 (1995), pp. 427-459

A Comparative Analysis of Entity-Relationship Diagrams[1]

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

[1]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

Journal of Computer and Software Engineering, Vol. 3, No.4 (1995), pp. 427-459

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^+ 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.

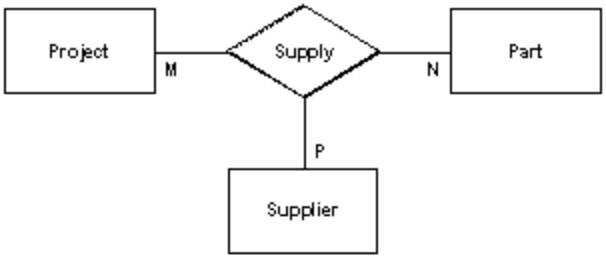


FIGURE 1 (a)
m:n:p Tenary Relationship

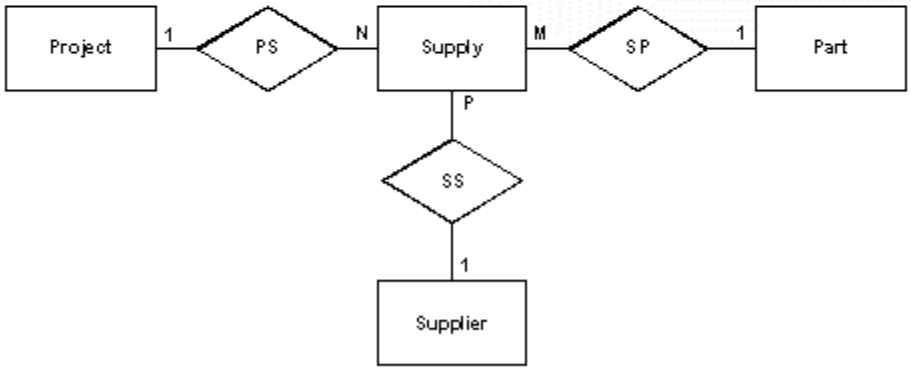


FIGURE 1 (b)
Representing Fig 1 (a) in binary models

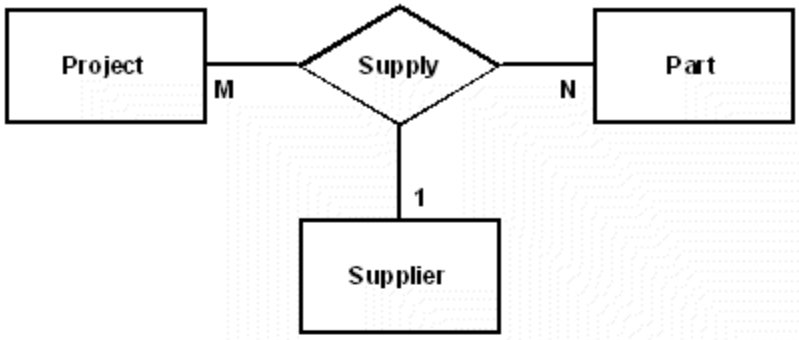


FIGURE 1 (c)
m:n:1 Tenary Relationship

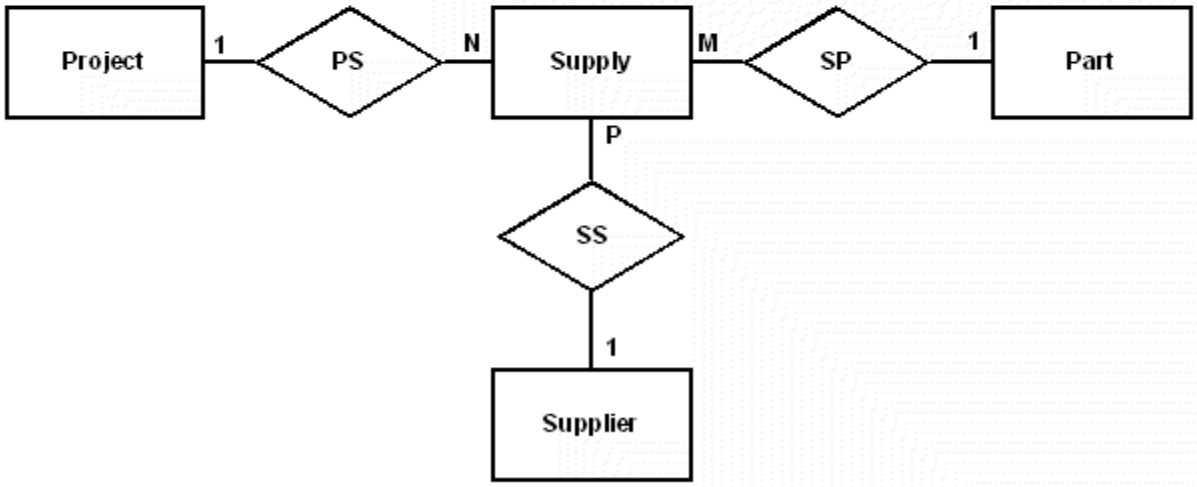
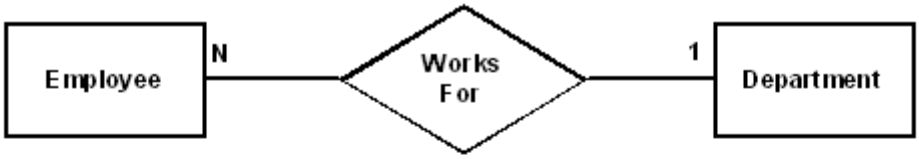


FIGURE 1 (d)
Representing Fig 1 (c) in binary models

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.



(a) Look Across

(b) Look Here

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.

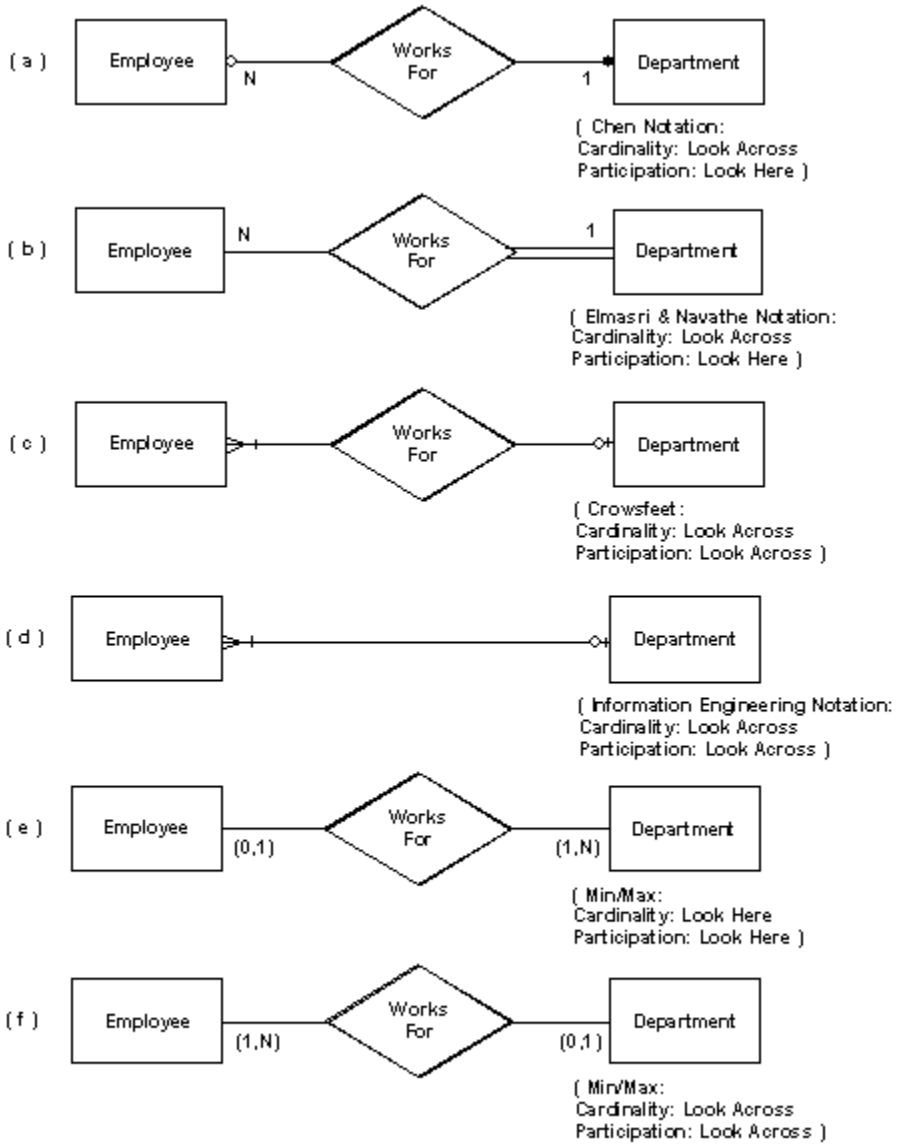


FIGURE 3
Various Notations for Cardinality and Participation Constraints showing "one employee can work for zero or one department and a department can have one or more employees."

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 $=,>,\geq,<,\leq$ 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, IDEF1X 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.

IDEF1X 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). IDEF1X 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.

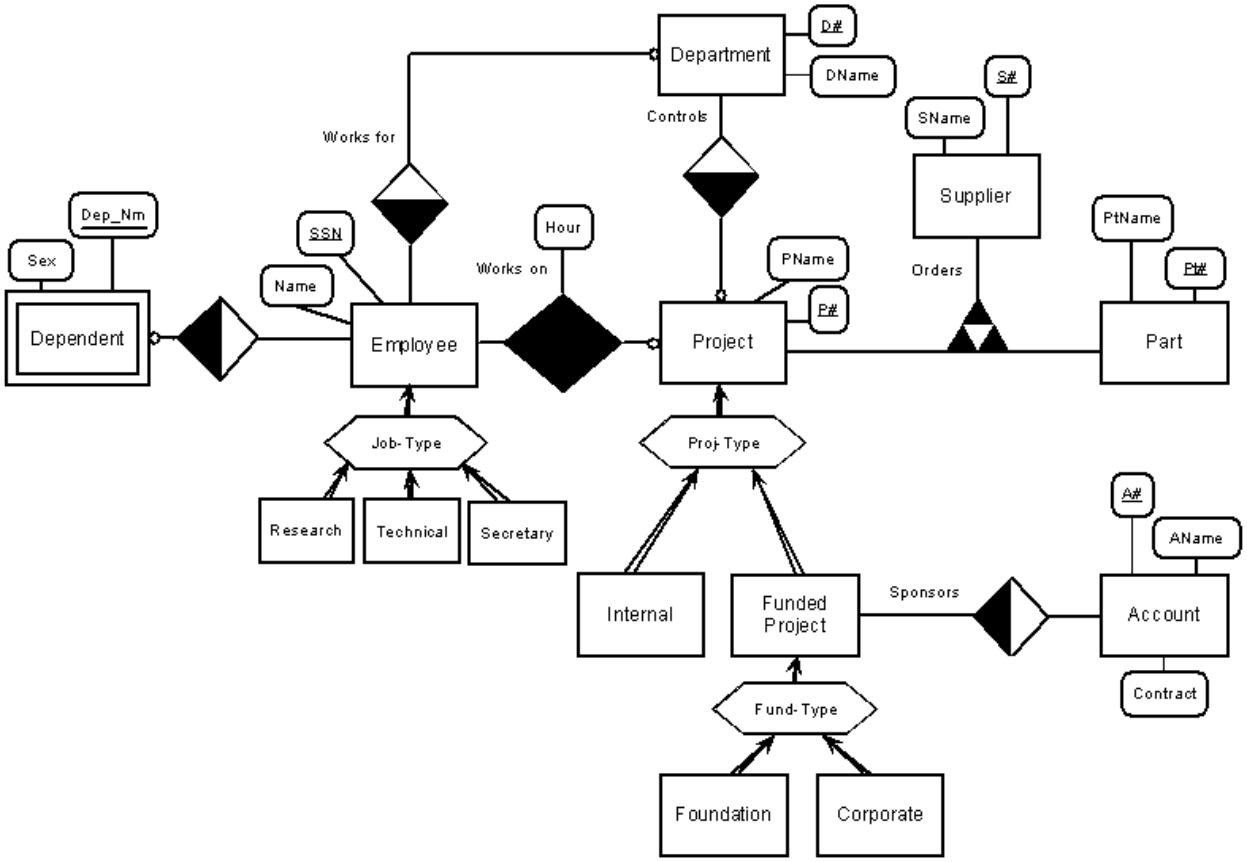


FIGURE 5: TEOREY's Notation

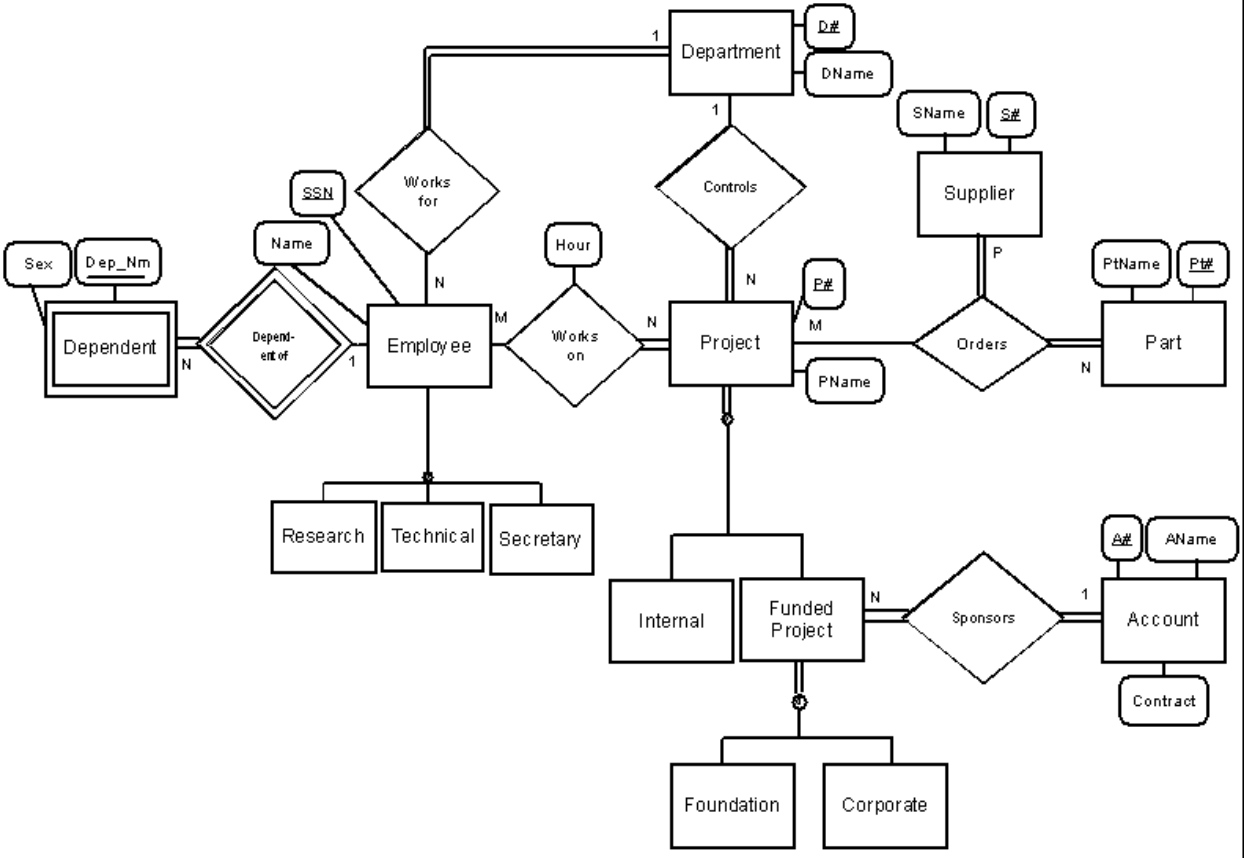


FIGURE 6: ELMASRI & NAVATHE's Notation

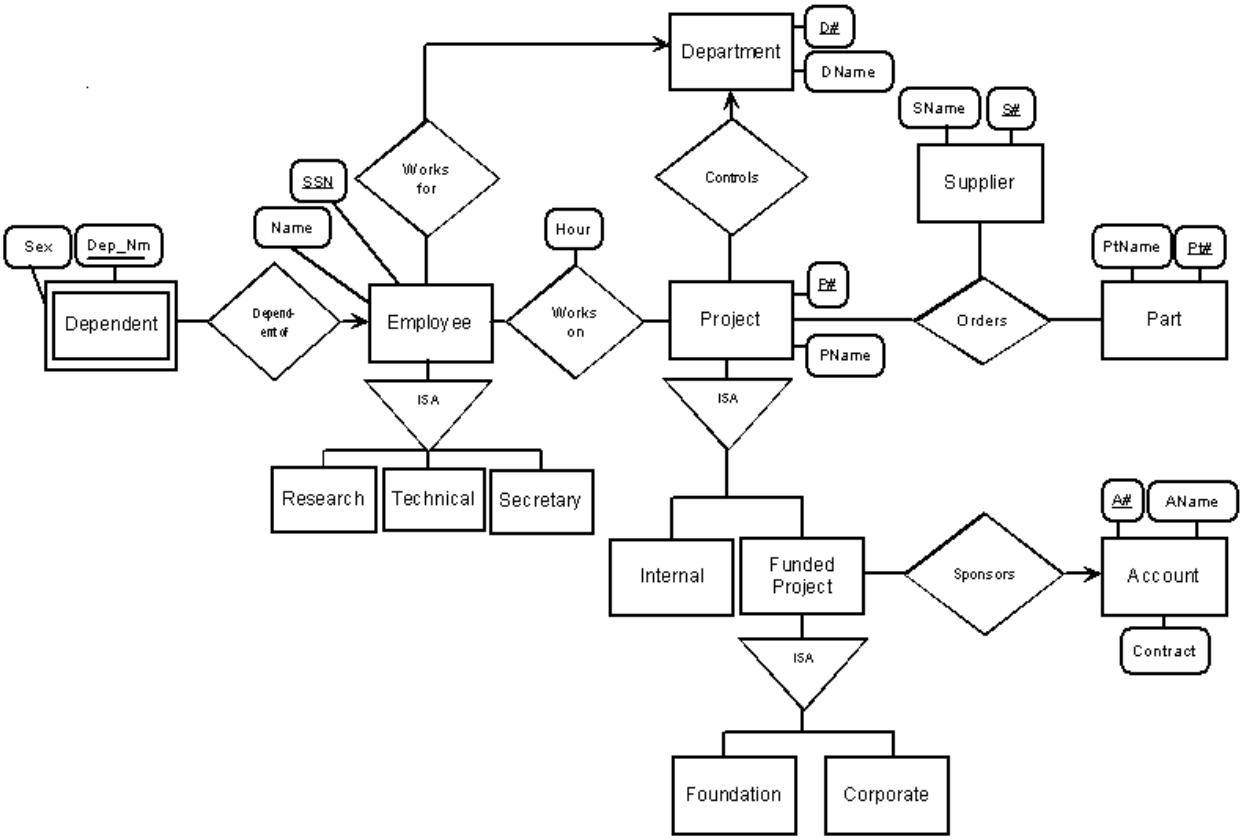


FIGURE 7: KORTH's Notation

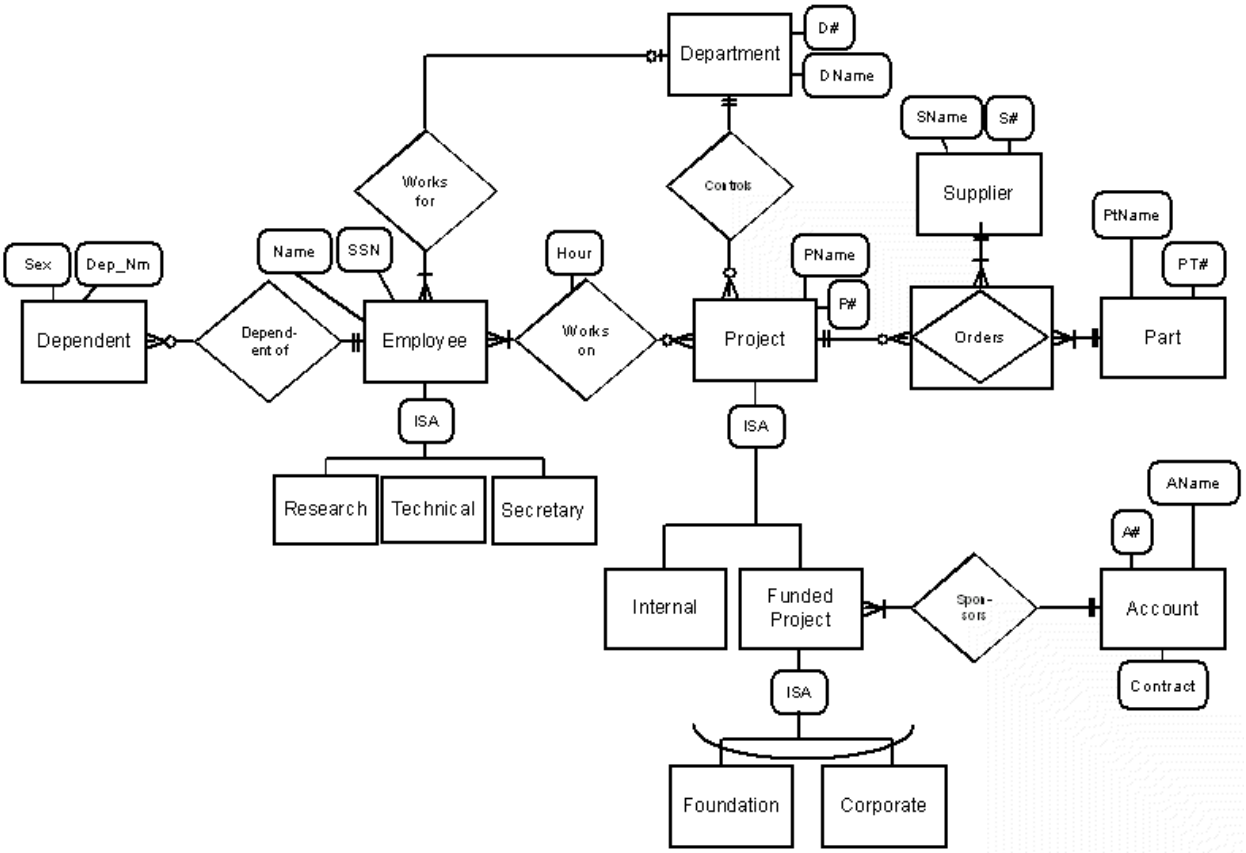


FIGURE 8: McFADDEN & HOFFER's Notation

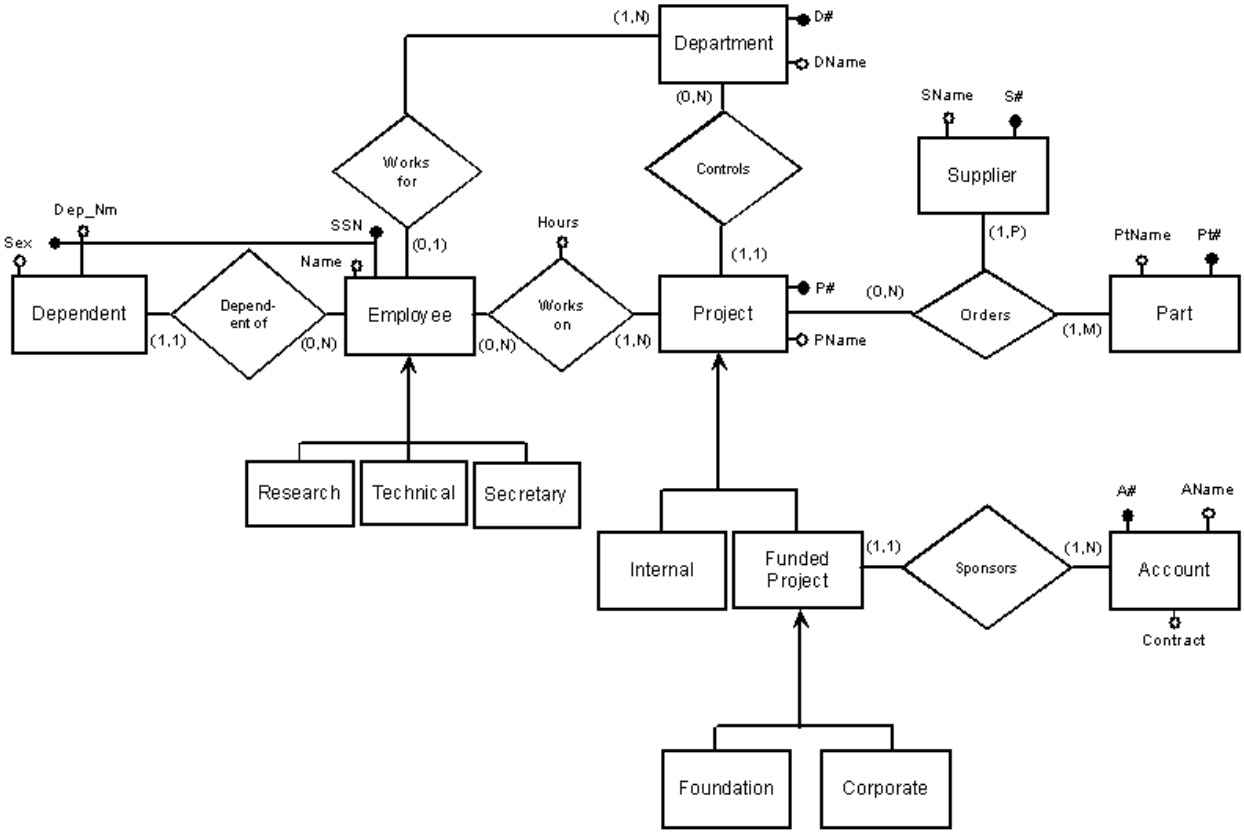


FIGURE 9: BATINI, CERI, & NAVATHE's Notation

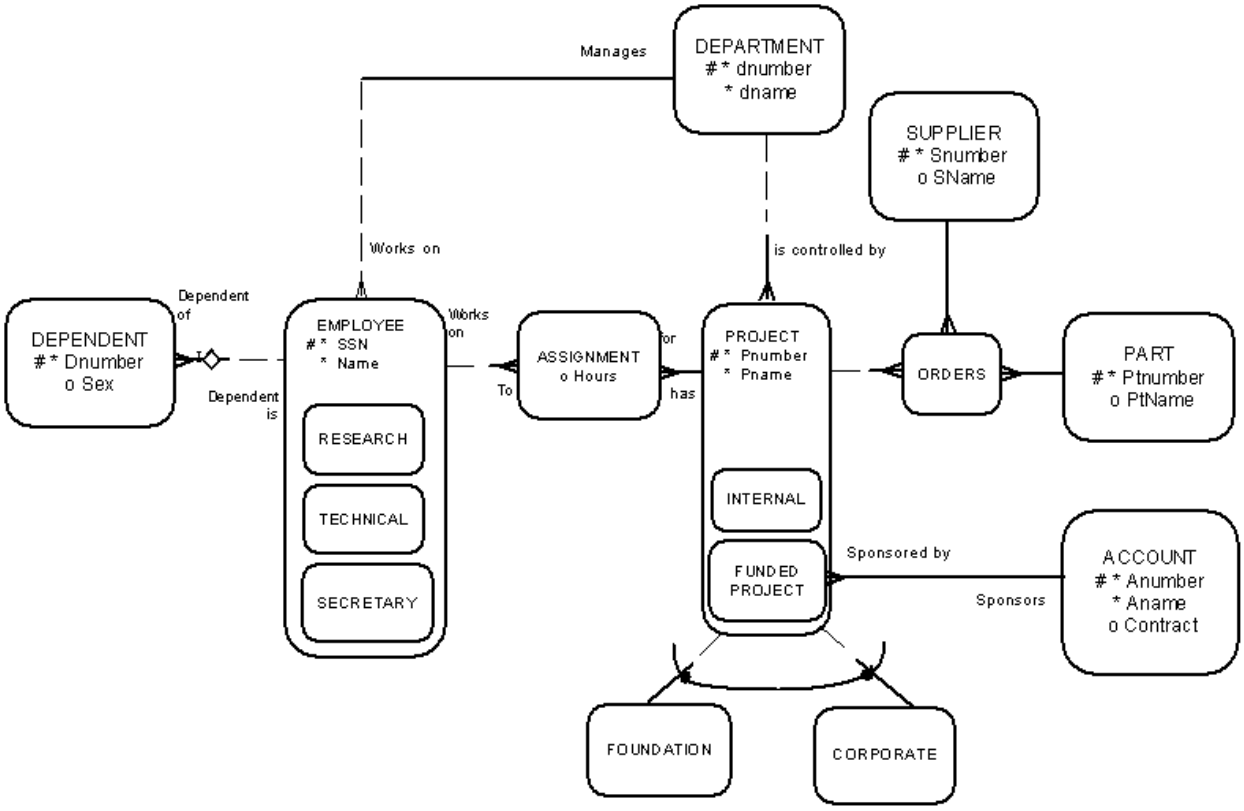


FIGURE 10: ORACLE's CASE* METHOD Notation

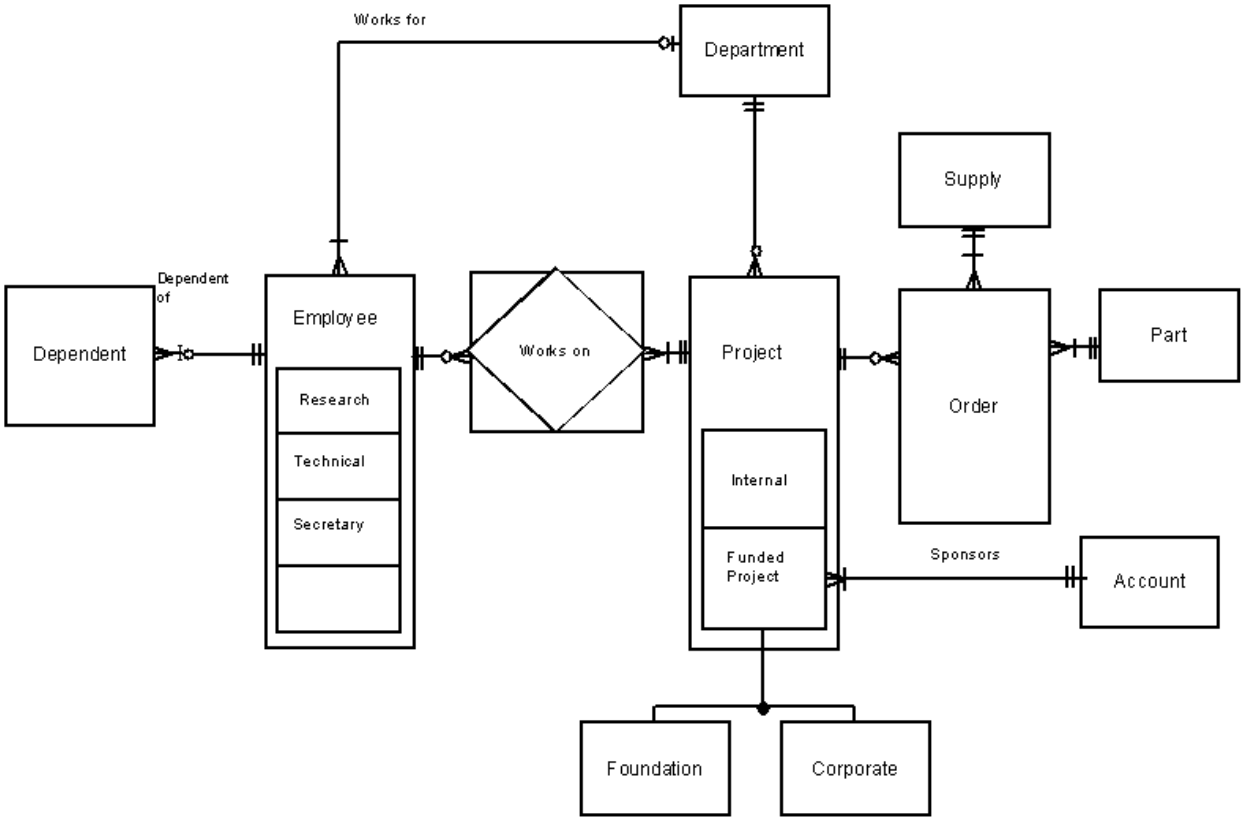


FIGURE 11: INFORMATION ENGINEERING Notation

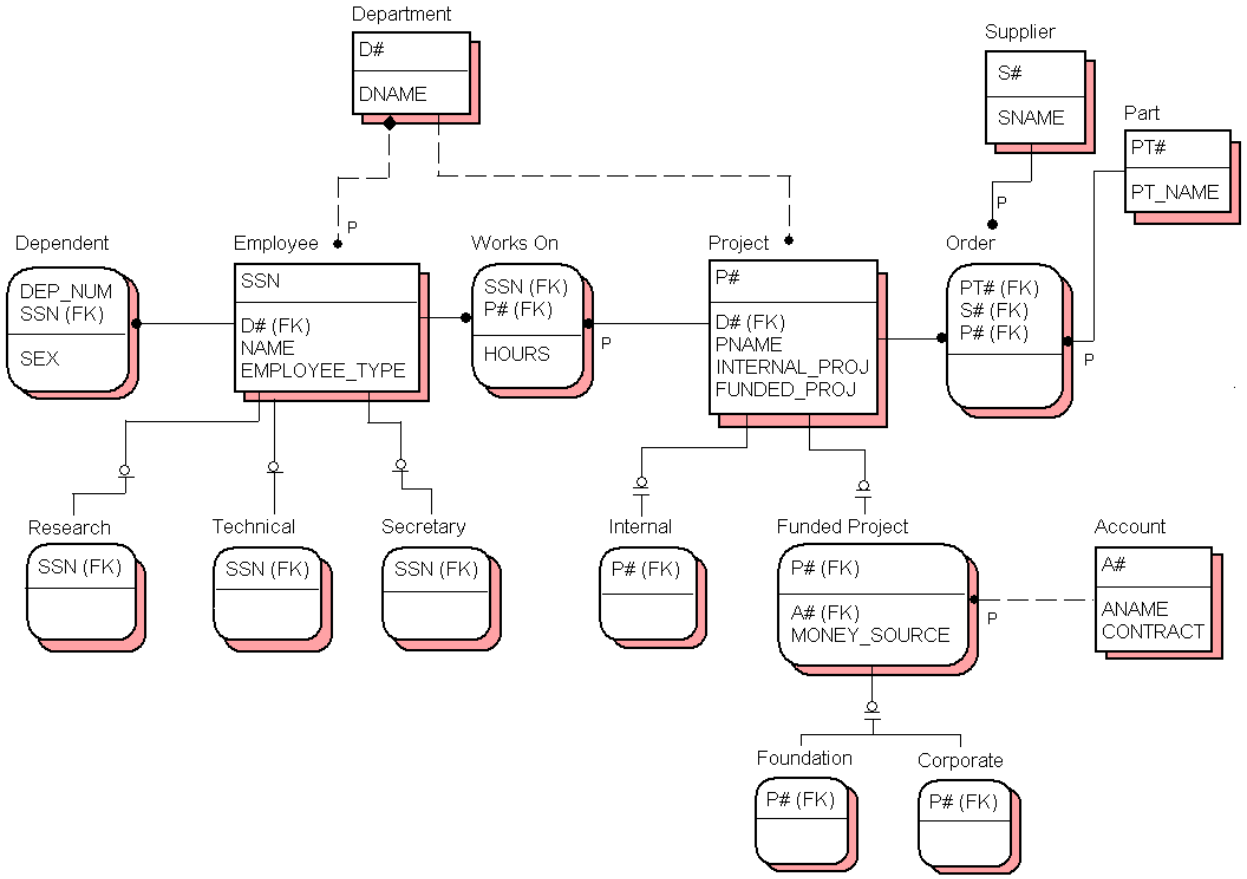


FIGURE 12: ERwin/ERX 2.0 Notation

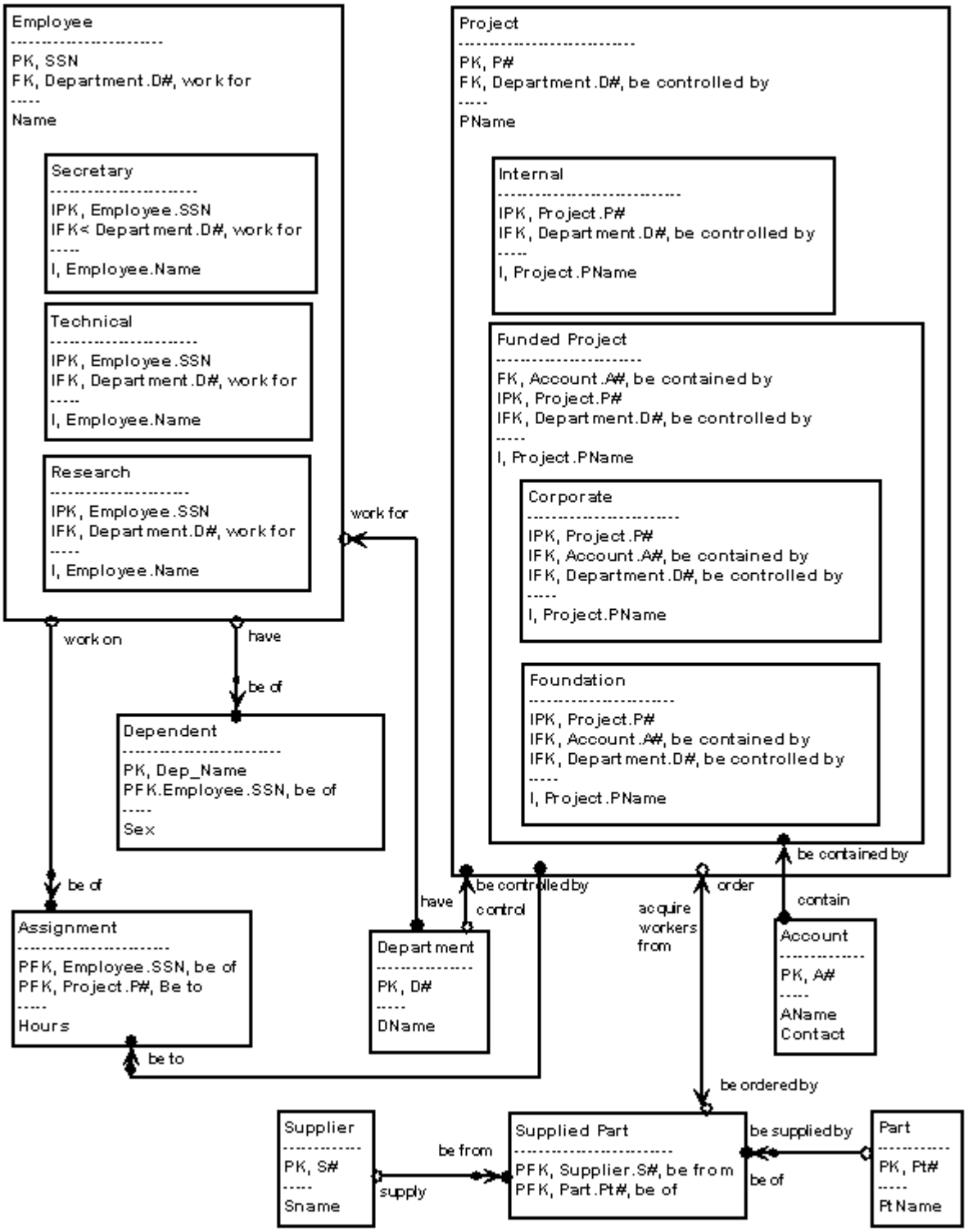


FIGURE 13: BACHMAN Notation

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 1 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.

N-ary	Chen; Teorey; Elmasri & Navathe; Korth & Silberschatz; McFadden & Hoffer; Batini, Ceri, & Navathe
Binary	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[1]; 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

[1] Total participation is not shown.

No Foreign Key at the ERD level	Chen; Teorey; Elmasri & Navathe; Korth & Silberschatz; McFadden & Hoffer; Batini, Ceri & Navathe; Oracle CASE*Method; Information Engineering.
Modeling Foreign Key at the ERD level	IDEF1X, Bachman.

Table III: Modeling Foreign Key at the ERD level

Journal of Computer and Software Engineering, Vol. 3, No.4 (1995), pp. 427-459

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 IDEF1X 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

Publishing, Redwood City, CA., 1994. 4th ed..

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*^{TM}*: 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*

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, Elsvter, Netherlands, 1980, pp. 121-140.

24. T. J. Teorey, *Database Modeling and Design: The Fundamental Principles*, 2nd ed., Morgan Kauffmann, 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. Chovbineh, "The Object-oriented entity-relationship model (OOERM). *Journal of Management Information Systems*, 7,3, (Winter 1990-1991), pp41-65.