**INTRODUCTION**

A database is a structured collection of data, and is usually stored electronically to facilitate computer-assisted search and retrieval. The electronic format allows different users to have views of the data that serve their particular needs. Furthermore, the database management software can ensure the consistency, integrity, and security of the database, provided the data needs of the organization have been correctly modeled and implemented.

**DATA MODELING**

Although the field of information systems engineering is young relative to most other engineering disciplines, it does apply formal modeling methods to study and predict aspects of a planned system. Different kinds of models may exhibit the user interfaces, behaviours, or data interchanges of the planned system.

Modern businesses depend heavily on timely, trustworthy, and relevant data to provide a competitive advantage (Lau, 2010). As a result, organizations invest in modeling their enterprise data. Data models describe the logical and physical structure of data that the information systems maintain and process.

The main goal of data modeling is to identify all of the data elements that the enterprise uses to manage its business, along with the structure and interrelationships of these elements. Data modeling typically follows activity modeling, which defines the functional requirements for the system. Its focus is to build a standard representation for the system’s data elements with little regard for when components of the system may create, modify, or access the data. This representation affords all applications a consistent, integrated view of the corporate information.

The entity-relationship diagram (ERD) is the most commonly used model for designing databases. The ERD model identifies the important data elements and organizes them into a collection of entities and relationships. An entity represents a separately identifiable subject of interest to the enterprise, typically a person, place, thing, or event. The ERD not only includes entities and relationships but also may include a set of attributes that describes each entity and identifies primary and foreign keys.

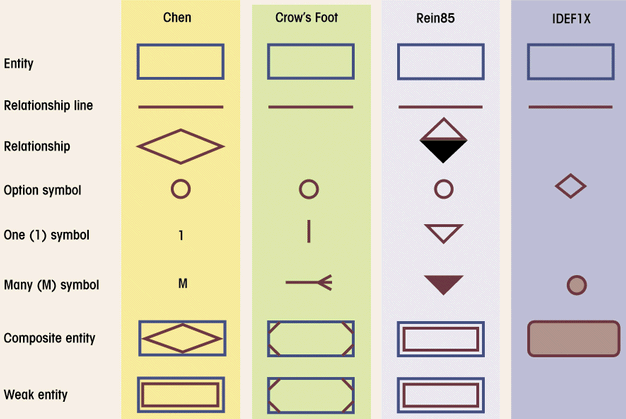
**ENTITIES**

Entities are concepts within the ERD model. Each entity is represented by a box within the ERD. An entity represents a separately identifiable subject of interest to the enterprise, typically a person, place, thing, or event. An entity might be considered a container that holds all of the instances of a particular thing in a system. Entities are equivalent to database tables in a relational database, with each row of the table representing an instance of that entity.

Remember that each entity represents a container for **instances** of the thing in question. The diagram below has an entity for *student* and another for *school.* This indicates that the system being modeled **may** contain one or more students and one or more schools.

****

There are several notations to represent ERD diagrams as indicated in the diagram below. The most popular notation is Crow’s foot as it is supported by many popular software tools.



**RELATIONSHIPS**

Relationships are represented by lines between entities. Relationship lines indicate that each **instance** of an entity may have a relationship with instances of the connected entity, and vice versa.



The diagram above now indicates that students may have some relationship with schools. More specifically, there may be a relationship between a particular student (an **instance** of the student entity) and a particular school (an **instance** of the school entity).

If necessary, a relationship line may be labeled to define the relationship. In this case, one can infer that a student may attend a school, or that a school may enroll students. But if necessary, this relationship could be labeled for clarification:



Read the first relationship definition, *attends*, when tracing the relationship left to right or top to bottom. Read the second definition, *enrolls*, when tracing the relationship right to left or bottom to top.

**OPTIONALITY AND CARDINALITY (Using Crow’s Foot)**

Symbols at the ends of the relationship lines indicate the *optionality* and the *cardinality* of each relationship. Optionality expresses whether the relationship is optional or mandatory. “Cardinality” expresses the maximum number of relationships.

As a relationship line is followed from an entity to another, near the related entity two symbols will appear. The first of those is the optionality indicator. A circle indicates that the relationship is optional—the minimum number of relationships between each instance of the first entity and instances of the related entity is zero. One can think of the circle as a zero, or a letter “O” for optional. A stroke ( | ) indicates that the relationship is mandatory—the minimum number of relationships between each instance of the first entity and instances of the related entity is one.

The second symbol indicates cardinality. A stroke ( | ) indicates that the maximum number of relationships is one. A crow’s-foot indicates that many such relationships between instances of the related entities might exist.

The following diagram indicates all of the possible combinations:



In our model, we wish to **indicate** that **each** school may enroll **many** students, or **may not** enroll **any** students at all. We also wish to indicate that each student attends exactly one school. The following diagram indicates this optionality and cardinality:



It is important to note that relationship optionality and cardinality constraints apply specifically to the **system being modeled**, not to all possible systems. According to the example modeled above, a school might not enroll any students—that relationship is optional. A school without students isnot much of a school, and indeed if the system being modeled were a school system enrollment database, the relationship would probably be mandatory. However, if the system being modeled is an extracurricular honors program, there may be schools that have no students currently participating in the program. Consider the function of the system and consult the other documents in the data model to **clarify modeling decisions**.

The ERD may include attributes, primary and foreign keys in addition to the entities and relationships represented in the ERD diagram.

Each attribute represents a characteristic that should be recorded for each entity. For example, a **Customer** entity may have attributes that include **Customer Name**, **Address**, **Phone Number**, and **Frequent Shopper Number**. The term ‘entity’ (or ‘entity type’) refers to the general notion or class, in this case, the idea of customers, but not particular values for the attributes.

The term ‘instance’ (or ‘entity instance’) refers to an individual member of the class. Each instance of **Customer** would have specific values for its attributes—for example, **John Eng**, **1 Any Street**, **03-77-53622**, and **985410**. The logical data model describes and explains the intended purpose or meaning of the entities and attributes. Because entities describe things, their names are usually nouns.

The combination of attribute values for any entity instance must be unique for that entity type. Typically, one or two attributes guarantee uniqueness, and these are called the identifier (or key) for the entity. For example, each **Customer** could have a unique **Customer ID** assigned. Several customers may have the same name or phone number, so these cannot be used as identifiers. Similarly, unless every customer is automatically assigned a unique **Frequent Shopper Number**, this attribute cannot be used as an identifier.

Over the years, several different models have been popular for database implementations. The relational model is currently the best accepted. This model applies first-order predicate logic and collects data into a two-dimensional matrix called a table. Each row of a table represents an instance. Typically, the entities and attributes of the logical model become tables and fields (or columns), respectively, in the physical model. For example:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Title** | **Salary** | **Hire Date** |
| Corrie Dakan | Auditor | 31525 | 1-Jun |
| Annamarie Mender | Webmaster | 41500 | 16-Feb |
| Floy Pini | Buyer | 26660 | 3-Nov |
| Roma Prenger | Sr. Technician | 37602 | 7-Jul |
| Kently Schuffert | Manager | 34000 | 17-Apr |
| Thea Stremcha | Salesperson | 27000 | 3-Nov |

The identifying attributes of an entity become key fields of the table. The term **primary key** refers to the collection of key fields that uniquely identifies a row in a table.

The relational database represents relationships with common fields. For one-to-many relationships, you usually include a copy of the primary key of the ‘one’ side of the relationship in the table for the ‘many’ side. We call the copy a foreign key when used this way. In the following example, Corrie Dakan works for the Accounting department and Annamarie Mender works for the IT department:

**DeptNo** DeptName

1

2

3

4

5

8

Name

Corrie Dakan

Annamarie Mender

DeptNo

5

8

Facilities

Purchasing

Sales

Shipping

Accounting

IT

For one-to-one relationships, you can select primary key as a foreign key.

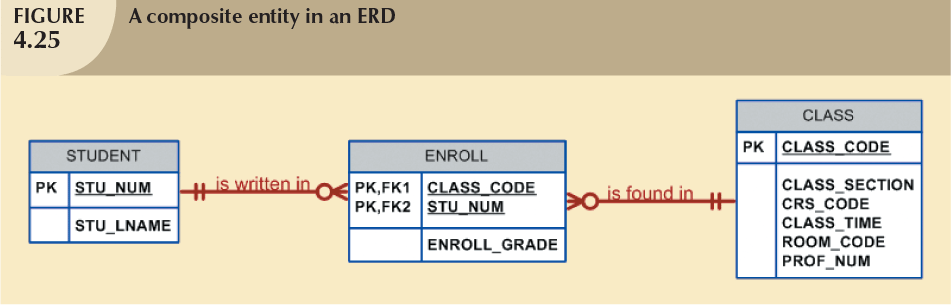
**Many-to-many relationships** typically require an **additional associative table** that combines the primary keys of the associated tables.

**ASSOCIATIVE TABLES**

Associative tables are also known as bridge tables. They are used to implement M:N relationships and they are composed of primary keys of each of the entities to be connected. They may also contain additional attributes that play no role in connective process. The example in 4.24 shows a many to many relationship between student and class.



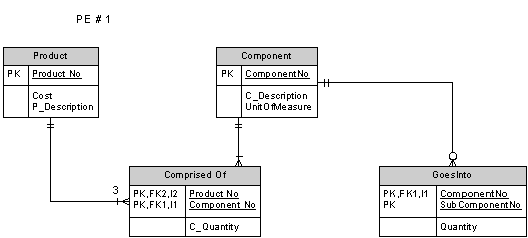
Figure 4.25 shows how an associative table can break a many to many relationship between student and class. The enroll table is composed of the primary keys of the student and class tables and includes the attribute enroll grade that is common to both tables (enroll and student).



In order to illustrate how to prepare a logical data model, the example below is used:

Assume that at Pine Valley Furniture each product (described by Product No., Description, and Cost) is comprised of at least three components (described by Component No., Description, and Unit of Measure) and components are used to make one or many products (i.e., must be used in at least one product). In addition, assume that components are used to make other components and that raw materials are also considered to be components. In both cases of components being used to make other components, we need to keep track of how many components go into making something else. Draw a logical data model diagram for this situation and place minimum and maximum cardinalities on the diagram.

Microsoft Visio was used to prepare the answer below. Please notice primary keys are indicated with the tag “PK” while foreign keys with the tag “FK”.

****

**DATA DICTIONARIES**

A data model may identify hundreds of different entities and attributes organizations typically gather the information about these elements (called metadata) in a data dictionary (Toombs, 2010). Although a single format has not been standardized for data dictionaries, for each attribute the database administrators may record its name, description, type, format, and length, constraints, and default value. The data dictionary helps users to understand the intended purpose of each attribute.

During conceptual data modeling, this authoritative and central record of data definitions helps ensure consistency. Because the data dictionary has names for all the entities and attributes defined so far, data modelers can follow the patterns when they create new names and avoid conflicts with existing names.

Likewise, data modelers can adopt the format of an existing attribute such as a date or customer name. This practice ensures consistent representations across all of the corporate data. Imagine the difficulties if the order-entry application allows 40 characters for a customer’s name but the billing system can use only 24 characters. The decision to have 24 or 40 characters is somewhat arbitrary, but a consistent standard is valuable. It also saves data modelers from the need to make similar decisions repeatedly.

A physical data model derives from the logical data model. While the logical model concentrates on the semantics of data elements, their meanings and interrelationships, the role of the physical data model is to define specific database structures to implement the concepts. The data dictionary can help guide this transition.

**DATA QUERIES**

Database users must often search for and retrieve stored information. Consider, for example, the question, ‘who of your employees are best paid?’ Although it seems straightforward, ‘best’ could mean many different things, such as the ones that earn the most, those who are paid best per hour, those who are the highest paid for their job title, or those that were paid the most during all their years of working for you. Similarly, we could answer ‘who’ with the employees’ unique numbers, but a more useful response would include the employee name, title, and department. We don’t even know how many employees to rank among the best paid!

This example should demonstrate that natural language is not precise enough to interpret clearly. If you were to ask an assistant this question, you would probably engage in a discussion to clarify your meaning. With a DBMS, you use a query language that makes your expectations clear. The standard language of relational databases, SQL, is both simple and precise. You specify the fields you want to select, the table(s) to select from, and the condition(s) where you want a row selected. One way to identify your best employees and their salaries could be the following (using the HR table above):

SELECT EmpName, Salary

FROM HR

WHERE Salary > 30000

This will produce the following table as an answer:

|  |  |
| --- | --- |
| **EmpName** | **Salary** |
| Corrie Dakan | 31525 |
| Annamarie Mender | 41500 |
| Roma Prenger | 37602 |
| Kently Schuffert | 34000 |

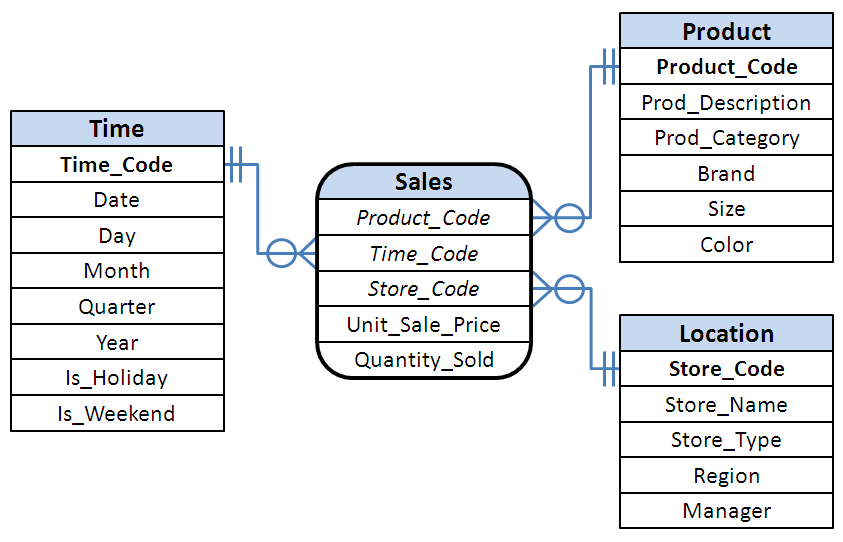
Queries can combine, or join, rows from related tables and they can summarize, or aggregate, data from multiple rows. For example, a query could count the number of employees in each of the company’s departments. Many Web-based tutorials, for example Refsnes Data (2010) and *SQL SELECT Statement* (2009), present both the fundamentals and the advanced features of SQL.

### **Data Warehousing**

A data warehouse is an integrated repository of corporate data, organised to facilitate decision making (Reed, 2002). Warehouses typically merge information from the disparate transactional systems used by different divisions and locations of the enterprise. So as to present a complete and accurate picture, data warehouses may also incorporate content from external sources, such as suppliers and even public records.

Transactional databases process primarily a standard menu of data modifications. They must make these real-time operations quick and consistent. In contrast, the data warehouse optimises ad hoc data retrieval queries and must often aggregate a collection of data (such as total sales volume by department over a given time interval). The relatively infrequent updates are background tasks that do not need instantaneous response. This difference in orientation has led many data warehouses to adopt the so-called star schema (Utley, 2008).

A star schema has two types of data (Borysowich, 2007)—a fact table that represents the factual or quantitative data, linked by one-to-many relationships to a collection of dimension tables that describe the subjects of the business. For example, a fact table that represents retail sales could have dimension tables for the product, store location, and time of each sale, as follows:



The dimension tables give decision makers and data mining software quick access to the data from a variety of perspectives—for example, comparing the weekend sales volume of casual shoes in each of the chain’s different types of stores, or the pattern of men’s dress shoe sales by month in each sales region.

**REFERENCE LIST**

Hillyer, M. (n.d.) *An introduction to database normalization*: http://ftp.nchu.edu.tw/MySQL/tech-resources/articles/intro-to-normalization.html (Accessed: 26 Sept. 2014).

Lau, K. (2010) ‘Three core pillars of a data-driven enterprise’, *ComputerWorld Canada* [Online]. http://www.itworldcanada.com/news/three-core-pillars-of-a-data-driven-enterprise/140662 (26 Sept. 2014).

Reed, M. (2002) [*A definition of data warehousing*](http://www.dwinfocenter.org/defined.html)  (Accessed: 26 September 2014).

Refsnes Data (2010) *SQL tutorial* [Online]. http://www.w3schools.com/sql (Accessed: 14 September 2014).

*SQL SELECT statement* (2009): http://beginner-sql-tutorial.com/sql-select-statement.htm (14 September 2014).

Toombs, E. (2010) *What is a data dictionary?* [Online]. https://suite.io/erin-toombs/3e0229x (Accessed: 13 Sept. 2014).