Introduction to Databases and Data Modeling (Database Design)

An important aspect of most every business is record keeping. In this information society, this has become an important aspect of business, and much of the world's computing power is dedicated to maintaining and using databases. All kinds of data, from emails and contact information to financial data and records of sales, are stored in some form of a database. The quest is on for meaningful storage of less-structured information, such as subject knowledge. This guide describes some of the key elements of the technology with an emphasis on database normalization and a more practical approach to database design, as well as provide an introduction to the SQL language.

What is a Database?

A database is a logically structured set of data organized to allow efficient storage and retrieval. Databases permit manipulation or retrieval of their contained data in usable manners. There are many forms of database that store and organize information using different structures.

Microsoft SQL Server is a database management system or DBMS. This terminology should not be confused with the term, "database". A DBMS is the software that provides the facilities to create and maintain databases and manipulate the information stored within. In the case of SQL Server, the databases that are created are relational databases, so the product is often known as a relational database management system or RDBMS.

Relational Database

Relational databases are probably the most common type of database used for general-purpose tasks. In a relational database, information is grouped according to its type, generally in tables. For example, in a database designed to hold fleet information you may include a table of employees and a table of vehicles. These two types of structured data would be held separately as they hold fundamentally different information.

In addition to separating information according to its data structure, a relational database allows *relationships* to be created between the data. A relationship defines a possible link between data types; the actual linkage of data is dependent upon the information held. For example, in the fleet database there may be a relationship between employees and vehicles, indicating that one or more employees drive a particular vehicle.

To achieve a clear data separation and relationships, we need to pass through rigorous steps of data modeling/ database design.

Data Modeling Overview

A data model is a representation of the data structures that are required by a database. The data structures include the data objects, the associations between data objects, and the rules which govern operations on the objects. As the name implies, the data model focuses on what data is required and how it should be organized rather than what operations will be performed on the data. Data model is equivalent to an architect's building plans. A data model is independent of hardware or software constraints, i.e. focuses on representing the data as the user sees it in the "real world". It serves as a bridge between the concepts that make up real-world events and processes and the physical representation of those concepts in a database.

Data Modeling/ Database Design

While there are two major methodologies used to create a data model: Entity-Relationship (ER) approach and the Object Model, here we use the Entity-Relationship approach. Database design is defined as: **"Design the logical and physical structure of one or more databases to accommodate the information needs of the users in an organization for a defined set of applications".** Thedesign process roughly follows five steps:

1. Planning and analysis
2. Conceptual design
3. Logical design
4. Physical design
5. Implementation

The data model is one part of the conceptual design process. The other, typically is the functional model. The data model focuses on what data should be stored in the database while the functional model deals with how the data is processed. To put this in the context of the relational database, the data model is used to design the relational tables. The functional model is used to design the queries and application logic which will access and perform operations on those tables. Figure 1 below shows how functional and database requirements are handled in organizational software systems development/upgrade efforts.

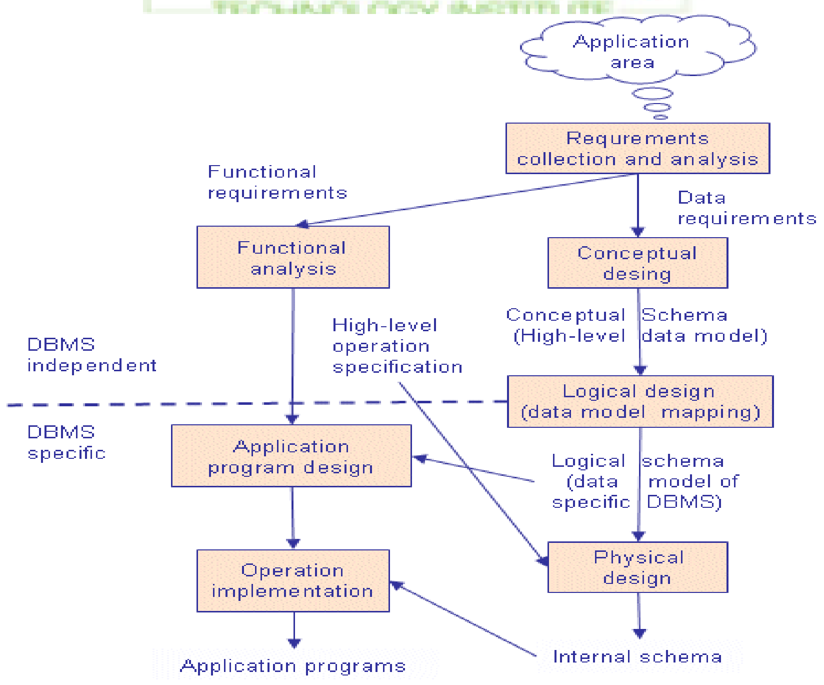


Figure 1: Functional design and Database design

Components of A Data Model

The data model gets its inputs from the planning and analysis stage. Here the modeler, along with analysts, collects information about the requirements of the database by reviewing existing documentation and interviewing end-users. The data model has two outputs. The first is an entity-relationship diagram which represents the data structures in a pictorial form. Because the diagram is easily learned, it is valuable tool to communicate the model to the end-user. The second component is a data document. This document that describes in details the data objects, relationships, and rules required by the database. The dictionary provides the detail required by the database developer to construct the physical database.

Why is Data Modeling Important?

Data modeling is probably the most labor intensive and time consuming part of the development process. Why bother especially if you are pressed for time? A common response by practitioners who write on the subject is that you should not build a database without a model than you should build a house without blueprints. The goal of the data model is to make sure that all the data objects required by the database are completely and accurately represented. Because the data model uses easily understood notations and natural language, it can be reviewed and verified as correct by the end-users. The data model is also detailed enough to be used by the database developers to use as a "blueprint" for building the physical database. The information contained in the data model will be used to define the relational tables, primary and foreign keys, stored procedures, and triggers – all these will be discussed later. A poorly designed database will require more time in the long-term. Without careful planning you may create a database that omits data required to create critical reports, produces results that are incorrect or inconsistent, and is unable to accommodate changes in the user's requirements.

In summary, a data model is a plan for building a database. To be effective, it must be simple enough to communicate to the end user the data structure required by the database yet detailed enough for the database design to use to create the physical structure. The Entity-Relation Model (ER) is the most common method used to build data models for relational databases. The next section provides a brief introduction to the concepts used by the ER Model

The Entity-Relationship Model

The Entity-Relationship (ER) model is a conceptual data model that views the real world as entities and their relationships. A basic component of the model is the Entity-Relationship diagram which is used to visually represents data objects.

For the database designer, the ER model:

- Maps well to the relational model. The constructs used in the ER model can easily be transformed into relational tables.
- Is simple and easy to understand with a minimum of training. Therefore, the model can be used by the database designer to communicate the design to the end user and
- Can be used as a design plan by the database developer to implement a data model in a specific database management software.

Basic Constructs of E-R Modeling

The ER model views the real world as a construct of entities and association (relationship) between entities.

Entities

Entities are the principal data object about which information is to be collected. Entities are usually recognizable concepts, either concrete or abstract, such as person, places, things, or events which have relevance to the database. Some specific examples of entities are **Employee**, **Customer**, **Project**, **Student, Book**, **Course**, **Registration**, **Invoice**. An entity is analogous to a ***Table*** in the relational model.

Entities are classified as independent or dependent (in some methodologies, the terms used are strong and weak, respectively). An independent entity is one that does not rely on another for identification. A dependent entity is one that relies on another for identification. Considering the entities we discussed above, while **Student** and **Course** are independent entities, **Registration** is a dependent entity. The reason is **Registration** cannot exist in the absence of **Student** or **Course** entities.

An entity occurrence (also called an instance) is an individual occurrence of an entity. An occurrence is analogous to a row in the relational table. As such, *College Algebra (also known as Math 102)* is an Instance of the **Course** Entity.

Special Entity Types

Associative entities (also known as intersection entities) are entities used to associate two or more entities in order to reconcile a many-to-many relationship (more on this later).Subtypes entities are used in generalization hierarchies to represent a subset of instances of their parent entity, called the super-type, but which have attributes or relationships that apply only to the subset. For example, **Doctor** may be a **Sub type** of **Employee** entity.

Data modeling is preceded by planning and analysis. The effort devoted to this stage is proportional to the scope of the database. The planning and analysis of a database intended to serve the needs of an enterprise will require more effort than one intended to serve a small workgroup.

The information needed to build a data model is gathered during the requirements analysis. Although not formally considered part of the data modeling stage by some methodologies, in reality the requirements analysis and the ER diagramming part of the data model are done at the same time.

Requirements Analysis

The goals of the requirements analysis are to:

- determine the data requirements of the database in terms of primitive objects
- classify and describe the information about these objects
- identify and classify the relationships among the objects
- determine the types of transactions that will be executed on the database and the interactions between the data and the transactions

- identify rules governing the integrity of the data

The modeler(s), works with the end users of an organization to determine the datarequirements of the database. Information needed for the requirements analysis can be gathered in several ways:

- Review of existing documents - such documents include existing forms and reports, written guidelines, job descriptions, personal narratives, and memoranda. Paper documentation is a good way to become familiar with the organization or activity you need to model.
- Interviews with end users - these can be a combination of individual or group meetings. Try to keep group sessions to be small. If possible, try to have everyone with the same function in one meeting. Use a whiteboard, flip charts, or overhead transparencies to record information gathered from the interviews.
- Review of existing automated systems - if the organization already has an automated system, review the system design specifications and documentation
- Perform observation of the business process while the business is live

The requirements analysis is usually done at the same time as the data modeling. As information is collected, data objects are identified and classified as entities, attributes, or relationship; assigned names; and, defined using terms familiar to the end-users. The objects are then modeled and analyzed using an ER diagram. The diagram can be reviewed by the modeler and the end-users to determine its completeness and accuracy. If the model is not correct, it is modified, which sometimes requires additional information to be collected. The review and edit cycle continues until the model is accepted as complete sufficient to go forward with the implementation.

Three points to keep in mind during the requirements analysis are:

- Talk to the end users about their data in "real-world" terms. Users do not think in terms of entities, attributes, and relationships but about the actual people, things, and activities they deal with daily.
- Take the time to learn the basics about the organization and its activities that you want to model. Having an understanding about the processes will make it easier to build the model.
- End-users typically think about and view data in different ways according to their function within an organization. Therefore, it is important to interview the largest number of people as time permits.

Steps in Building the Data Model

While ER model lists and defines the constructs required to build a data model, there is no standard process for doing so. Some methodologies specify a bottom-up development process were the model is built in stages. Typically, the entities and relationships are modeled first, followed by key attributes, and then the model is finished by adding non-key attributes. Other experts argue that in practice, using a phased approach is impractical because it requires too many meetings with the end-users.

The following sequence is a general guide as a steps to build data model:

1. Identification of data objects (entities) and relationships

2. Drafting the initial ER diagram with entities and relationships
3. Refining the ER diagram
4. Add key attributes to each entity in the diagram (if the entities can have one)
5. Adding non-key attributes
6. Diagramming Generalization Hierarchies
7. Validating the model through normalization
8. Adding business and integrity rules to the Model

In practice, model building is not a strict linear process. As noted above, the requirements analysis and the draft of the initial ER diagram often occur simultaneously. Refining and validating the diagram may uncover problems or missing information which require more information gathering and analysis.

In summary, Data modeling must be preceded by planning and analysis. Planning defines the goals of the database, explains why the goals are important, and sets out the path by which the goals will be reached. Analysis involves determining the requirements of the database. This is typically done by examining existing documentation and interviewing users.

An effective data model completely and accurately represents the data requirements of the end users. It is simple enough to be understood by the end user yet detailed enough to be used by a database designer to build the database. The model eliminates redundant data, it is independent of any hardware and software constraints, and can be adapted to changing requirements with a minimum of effort.

Data modeling is a bottom up process. A basic model, representing entities and relationships, is developed first. Then detail is added to the model by including information about attributes and business rules. Below we will discuss identifying Data Objects and Relationships.

Identifying Data Objects (Entities) and Relationships

In order to begin constructing the basic model, the modeler must analyze the information gathered during the requirements analysis for the purpose of:

- Classifying data objects as either entities or attributes
- Identifying and defining relationships between entities
- Naming and defining identified entities, attributes, and relationships
- Documenting this information in the data document

To accomplish these goals, the modeler must analyze narratives from users, notes from meeting, policy and procedure documents, and, if lucky, design documents from the current information system.

Although it is easy to define the basic constructs of the ER model, it is not an easy task to distinguish their roles in building the data model. What makes an object an entity or attribute? For example, given the statement "employees work on projects". Should employees be classified as an entity or attribute? Very often, the correct answer depends upon the requirements of the database. In some cases, employee would be an entity, in some it would be an attribute. Same is true with the project.

While the definitions of the constructs in the ER Model are simple, the model does not address the fundamental issue of how to identify them. Some commonly given guidelines are:

- Entities contain descriptive information
- Attributes either identify or describe entities
- Relationships are associations between entities

These guidelines are discussed in more detail below.

- Entities
- Attributes
    - Validating Attributes
    - Derived Attributes and Code Values
- Relationships
- Naming Data Objects
- Object Definition
- Recording Information in Design Document

An **Entity** is any distinguishable object like person, place, thing, event, or concept, about which information is kept and represented in a database. It is a thing which can be distinctly identified. As such, **entity** is anything about which we store information (e.g. supplier, machine tool, employee, utility pole, airline seat, etc.). For each entity type, certain attributes are stored.

The above definition if an entity depicts:

- An **Entity** is a *"thing", "concept"* or, *object"*. However, entities can sometimes represent the relationships between two or more objects. This type of entity is known as an associative entity.

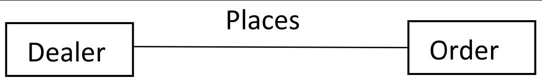
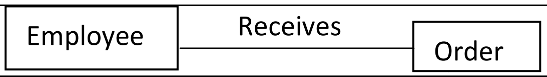
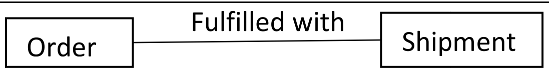
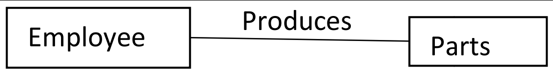
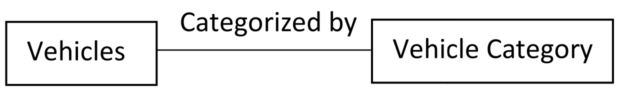
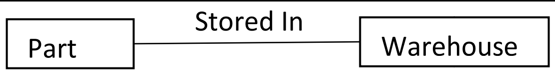
- **Entities** are objects which contain descriptive information. If data object you have identified is described by other objects, then it is an entity. *Note that, if there is no descriptive information associated with the item, it is not an entity. Whether or not a data object is an entity may depend upon the organization or activity being modeled.*
- An entity represents many things which share properties. They are not single things. For example, King Lear and Hamlet are both plays which share common attributes such as name, author, and cast of characters. The entity describing these things would be PLAY, with King Lear and Hamlet being instances of the entity.
- Entities which share common properties are candidates for being converted to generalization hierarchies
- Entities should not be used to distinguish between time periods. For example, the entities 1st Quarter Profits, 2nd Quarter Profits, etc. should be collapsed into a single entity called Profits. An attribute specifying the time period would be used to categorize by time
- Not everything the users want to collect information about will be an entity. A complex concept may require more than one entity to represent it. Others "things" users think important may not be entities.

Let’s consider some examples of entities in different scenarios

1. A Vehicle manufacturer, BMW, wants a database to keep track of its production, sales, orders and shipments. We can identify the following entities and their relationships:

Entities: Employees, Vehicle, Vehicle Category, Dealer, Order, Supply, Machine, Warehouse, Shipment, Parts

Relationships:

Verbal description	Entity-Relationship (E-R) Diagram
Dealers place Orders	
Employees receive Orders	
Orders are fulfilled with Shipment	
Employees produce Parts	
Vehicles are categorized by Automotive Category	
Parts are stored in a Warehouse	

Exercises:

- Identify more entities for BMW
- Think of or visit a nearby business and learn their business process by which they offer service or produce products. E.g. Department Stores, pharmacies, convenient stores, doctors' offices, schools, DMV, …
- Identify entities that participate in the service provision or production and name them (identify as many entities)
- Identify the relationship that exists between the entities and name them
- Draw the Entity-Relationship diagram

Logical Database Design

Logical database design is the process of describing each piece of information needed to track the relationships among entities or the business rules that govern, those pieces of information. As described above, entities (in ER model) are referred to as Tables in RDBMS, we will continue using the name Table from this point onwards.

Once a logical database design is created, you can verify with users and management that the design is complete (that is, it contains all of the data that must be tracked) and accurate (that is, it reflects the correct table relationships and enforces the business rules).

Creating a logical data design is an information-gathering, iterative process. It includes the following steps:

- Refine the entities defined in the conceptual database design phase based on the information your business requires.
- Refine the relationships between the tables
- Determine the attributes (columns) of each table
- Normalize the tables to at least the third normal form.
- Determine the primary keys and the column domain *.

*A domain is the set of valid values for each column. For example, the domain for the customerId can include all positive integers.

After the entities/tables defined at the conceptual database design phase are refined, by adding more tables or removing some, the most important step of identifying the attributes of the entities will follow.

Attributes

Attributes are characteristics or traits that describe, specify, qualify or quantify an entity.

Identification of attributes for every entity (table) is followed by definition of Attribute names, domains and constraints on the attributes.

Examples:

Entity	Attributes
Vehicle	VIN, VehicleType, model, make, modelYear
Employee	EmployeeId, firstName, lastName, SSN,
Parts	PartNumber, PartName, purpose, madeBy, madeInCountry, partSize,…
Supply	Supplier Name, supplied date, supplied item,

Exercises:

1. Identify attributes for all the other entities of BMW described above
2. Identify attributes for all the entities you identified in the previous exercise

Keys

Key attribute represents primary key is an attribute, that has distinct/unique value for each entity/element in an entity set. It is also possible that two or more attributes together can be used for this distinct identification. When two or more attributes are used as a key together, it’s called a **Composite Key**

Non-key attributes are attributes other than candidate key attributes in a table. For example Firstname is a non-key attribute as it does not distinctly identify an individual (as two or more people can have the same Firstname).

Normalization

Normalization is an iterative process during which you structure your database to reduce redundancy and increase stability. During the normalization process, you determine in which table a particular piece of data belongs based on the data itself, its meaning to your business, and its relationship to other data. Normalizing your database results in a data-driven design that is more stable over time.

Normalization requires that you know the business and know the different ways you want to relate the data in the business. When you normalize your database, you eliminate columns that:

- Contain more than one value
- Are duplicates or repeat
- Do not describe the table in which they currently reside
- Contain redundant data
- Can be derived from other columns

The result of each iteration of the normalization process is a table that is in a ***Normal Form***. After one complete iteration, your table is said to be in first normal form; after two, second normal form; and so on. The sections that follow describe the rules for the First, Second, and Third normal forms.

First Normal Form

The first rule of normalization is that you must remove duplicate columns or columns that contain more than one value to a new table. The **columns** of a table in the first normal form have these characteristics:

- They contain only one value
- They occur once and do not repeat

Let's consider the following example: **Table 1:**Dealer table (**Un-normalized** table)

Dealer Number	Company Name	street	AutoParts
101	Derik and sons	77 U st	Gasket, Spark plug, Brake pad, Belt
102	Downy Inc	321 Automobile blvd	Belt, Air filter, Sensors, Bearing

Here, the Parts Ordered column has more than one entry. This makes it very difficult to perform even the simplest tasks, such as deleting an order, finding the total number of orders for a dealer, or printing orders in sorted order. You can eliminate the complexity by normalizing the table so that each column in a table consists of exactly one value.

Table 2, below is the same Dealer table in a different **un-normalized** format which contains only one value per column, but this time the columns AutoPart1, AutoPart2, … are repeated

Dealer Number	Company Name	street	AutoPart1	AutoPart2	AutoPart3	AutoPart4
101	Derik and sons	77 U St	Gasket Belt	Spark plug	Brake pad	
102	Downy Inc.	321 Auto Blvd	Belt	Air filter	Sensors	Bearing

Here, instead of a single AutoPart column, there are three separate but duplicate columns for multiple orders. This format is also not efficient. What happens if a customer has more than three orders? You must either add a new column or clear an existing column value to make a new entry. It is difficult to estimate a reasonable maximum number of orders for a customer.

In order to reduce the Dealer table to the first normal form, split it into two smaller tables, one table to store only Dealer information and another to store only AutoPart Order information. Below Table 3 shows Normalized Dealer table and Table 4 is normalized AutoPart Order table

Table 3: Normalized **Dealer** table

Dealer Number (Primary key)	Company Name	Street
101	Derik and sons	77 U St
102	Downy Inc.	321 Automobile Blvd

Table 4: Normalized **AutoPartOrder** table

Dealer Number (Foreign Key)	AutoPart
101	Gasket Belt
101	Spark plug
101	Brake Pad
102	Belt
102	Air filter
102	Sensors
102	Bearing