**COMP 122 Project Notes & Guidelines**

A Database exists as an important part of an information system. As you may recall, information systems also contain people, procedures, input data, output data software and hardware. Your database will be an essential component of an e-commerce solution. Its goal is to facilitate the production of a system that supports the business needs, business requirements, is easy to use, and meets users’ needs. Moreover, it must store all the relevant data and provide information in proper formats for customers, employees and vendors.

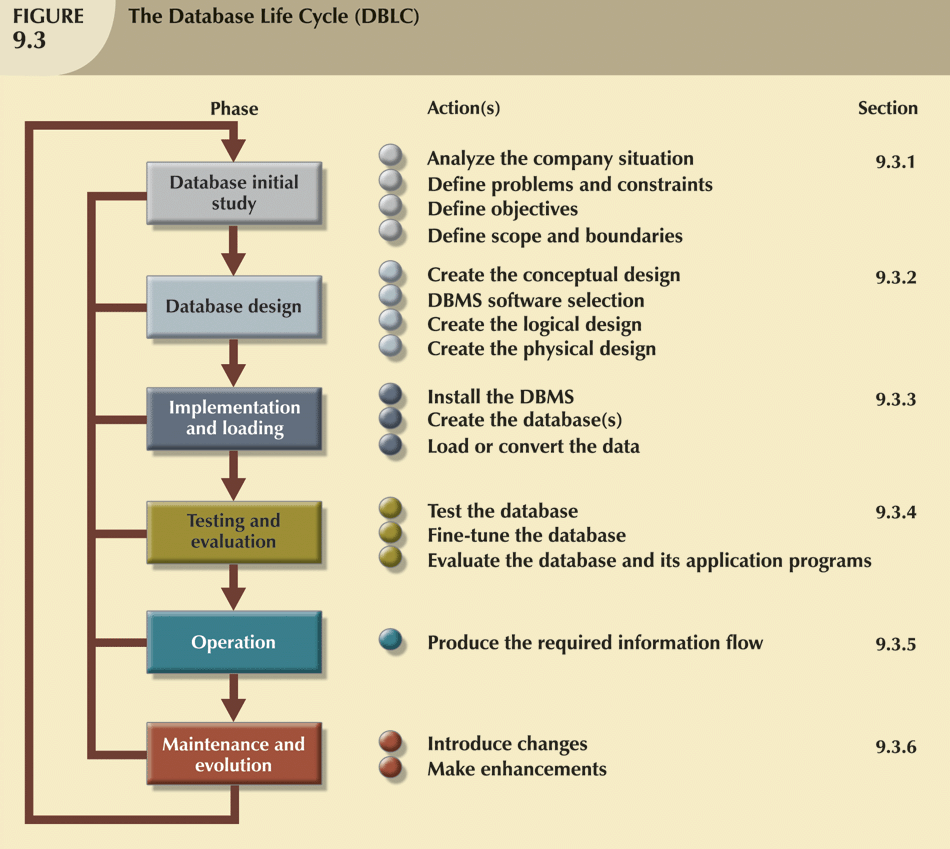
The first step in the database design process is to determine the data requirements which involve:

1. Collecting and evaluating existing data
2. Research missing or incomplete data
3. Talk to users to determine operation of the system, their needs, and the desired outputs

From your dealings with pharmacies, and the information gathered you will have an understanding of the data to store in the database and the entities (persons, places, things, or ideas) involved as briefly outlined below:

You will use Oracle on cencol or local and its to create and maintain the database.

You will follow the traditional systems d.evelopment life cycle for this project. Hence, your database design will encompass the activities associated with *requirements engineering* which consists of *requirements elicitation* and *requirements specification*.

Project Notes:

Database Development Phases

Conceptual Data

Modeling

Logical Database

Design

Distributed Database

Design

Physical Database

Design

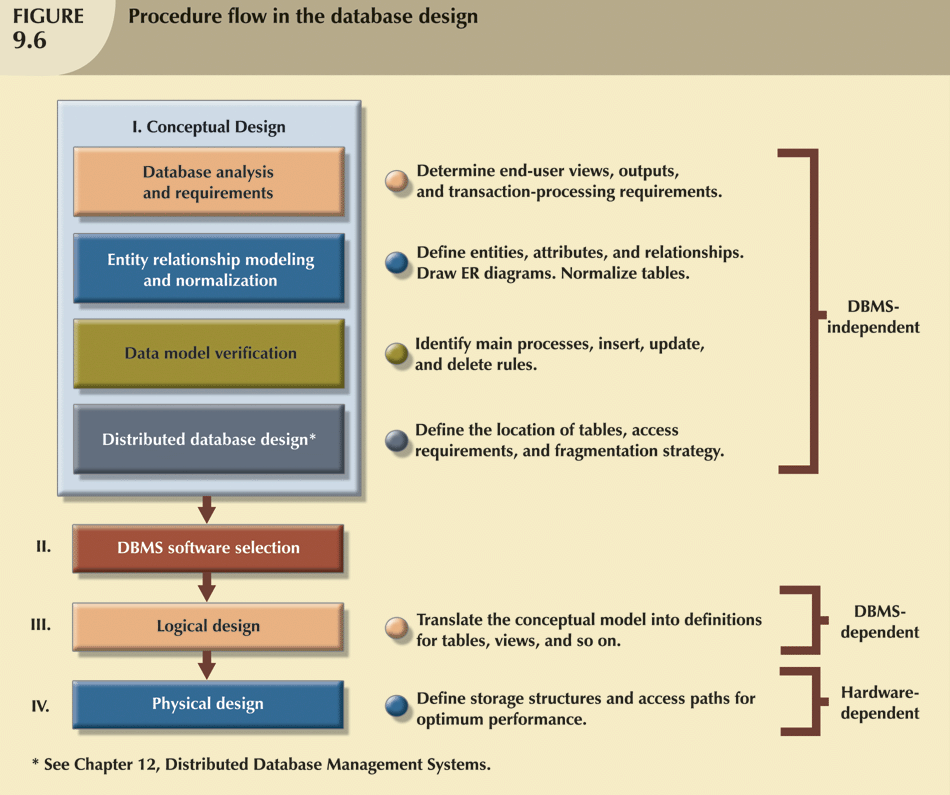
ERD

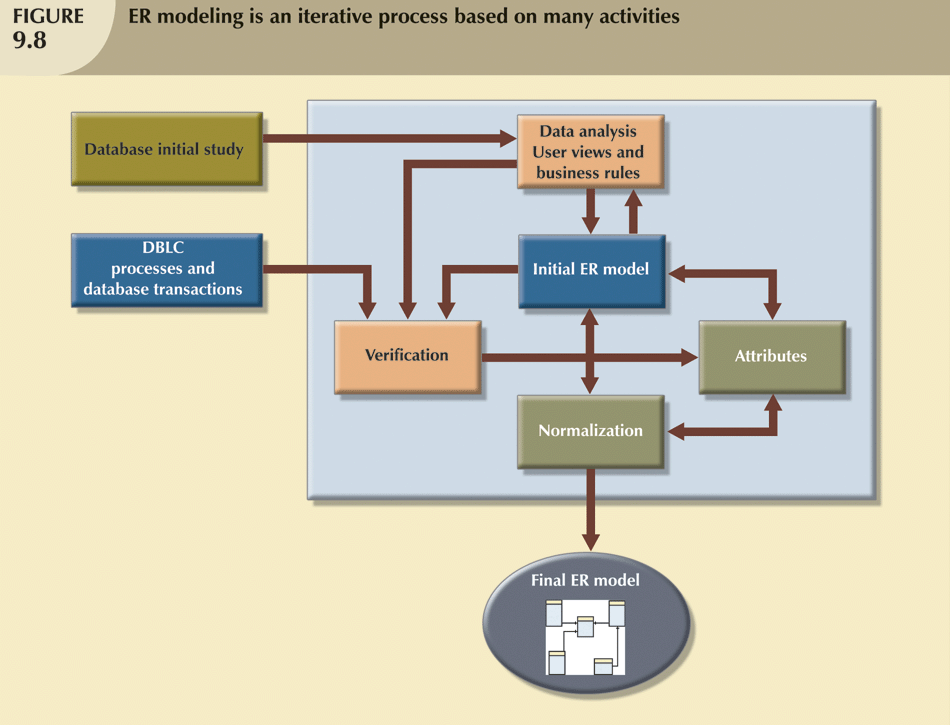
Tables

Distribution Schema

Internal Schema, Populated DB

Data requirements





1. **Data Requirements Phase**

The data requirements phase is concerned primarily with fact finding and would involve such things as interviews, surveys, research, etc which are done during *requirement elicitation*

1. **Conceptual Data Modeling Phase**

**Schema:** The schema contains descriptive information about the data stored in the physical data store and includes:

* Organization of individual stored data items in the tables
* Relationships / Associations among tables as shown in Entity Relationship diagrams (ERDs)
* Details of physical data store organization including field name, field types, field size and other related metadata.
* Access and content controls, allowable values used for specific data items, list of users allowed to read or update data items.

A generic description of the data that is not tied to any specific database software product such as Access, Oracle, SQL Server, SYBASE, DB2 etc. In this phase you will take the data collected in the data requirements phase and use it for the following schemas (database descriptions):

* 1. **External Schemas (End users Views of the database)**

Here you will focus on the particular usage of the database where each group of users can have a separate external view of the database tailored to meet their specific needs. Each external schema includes the appropriate entities, relationships, processes and constraints imposed by the business unit. Companies are generally divided into several business units such as sales, finance, marketing etc. Each business unit uses a data subset of the overall data in the organization. Therefore, end users working those business units view their data subsets as separate from, or external to other units within the organization. These views include forms, reports, etc. Some examples are:

* + - Employee assignment view (work schedule)
    - List of patients’ past prescriptions
    - Reports on possible side effects of a particular drug and adverse effect of each prescription dispensed.

The *external model* is the end users’ view of the data environment

* 1. **Conceptual Schema (Global view of the entire database)**

The conceptual model integrates all the external views (entities, relationships, constraints, and processes) into a single global view of the entire data in the organization, which is known as the *conceptual schema*. Thus, it is a representation of data as viewed by the entire organization. The conceptual model is independent of both hardware and software.

* + - For this project it defines entities such as doctors, prescriptions, drugs, health plan, customers, etc. and their relationship (named association between entity types). It produces Entity Relationship Models using Entity Relationship Diagrams (ERDs)

iii) **Internal Schema (representation of the database as seen by the Database Management System (DBMS)**

This represents the storage view of the database. It defines the files needed to store the tables, and the extra files needed to improve performance.

1. **Logical Database Design Phase**

The term *logical design* is used to refer to the tasks of creating a conceptual data model that could be implemented on any DBMS. Here you will transform the conceptual data model into a format that can be understood by a commercial DBMS. The logical design, for a database must specify the appropriate conceptual data model, inputs, processes, and expected output requirements. Once a specific DBMS has been selected, the internal model maps the conceptual model to the DBMS. Hence, the *internal model* is a representation of the database as “seen” by the DBMS. This, therefore, requires the designer to match the conceptual model’s characteristics and constraints to those of the selected implementation model. The *internal schema* depicts a specific representation of an internal model, using the database constructs supported by the chosen DBMS.

The following two refinements are performed during this phase:

1. **CONVERSION ACTIVITY**

Here you will transform the ERDs produced during the Conceptual Modeling phase into table design for each entity, and the associated relationship(s), using conversion rules. For a relational database, this phase produces a table design which includes: field definitions, primary keys, foreign keys, validation rules, properties settings, etc.

Below are the basic conversion rules:

1. **Entity Type Rule**

Each entity type becomes a table. The primary key of the entity type (except for weak entities) becomes the primary key of the table. The attributes of the entity type becomes the columns in the table.

1. **1-M Relationship Rule**

Each 1-M relationship becomes a foreign key in the table corresponding to the child entity type (entity type near the Crow’s Foot Symbol). When the minimum cardinality is one, the foreign key cannot accept Null Values.

1. **M-N Relationship Rule**

Tables in M-N relationship are not implemented because they create many redundancies and the contents and structure such tables make relational operations very complex. Therefore, each M-N relationship must be broken into two 1-M relationships.

This means that an M-N relationship will produce three tables i.e. one for each of the two entities(parent entities) and one for the entity (child entity) that came about as a result of the association of the parent entities and is also identification dependent on the parent entities. The primary key of the table that resulted from the relationship is a combined key, called a composite key, consisting of the primary keys of the entity types participating in the M-N Relationship.

1. **Identification Dependency Rule**

Each M-N relationship must be modeled as two 1-M relationships. The resulting identifying relationship (denoted by a solid relationship line) adds a component to primary key. The primary key of the table corresponding to the weak entity consists the following:

* 1. The underlined local key (if any) in the weak entity
  2. The primary key(s) of the entity type(s) connected by the identifying relationship(s)

These rules are used to convert the ERDs into table designs using the CREATE TABLE statement.

**Weak Entity:** A weak entity borrows all or part of its primary key from other entity type(s) in the identifying relationship(s).

**Identification Dependency**: When an entity is closely associated with other entities the resulting identifying relationship is denoted by a solid line. Identification dependency involves a weak entity and one or more identifying relationships. It is a form of existence dependency which yields a mandatory relationship (minimum cardinality of one)

**Identifying relationship**: A relationship that provides a component of the primary key to a weak entity

1. **NORMALIZATION ACTIVITY**

Recall that the goal of normalization is to split large tables into smaller related tables to remove redundancies. Redundancies can cause insert, update and delete operations to produce unexpected side effects which are referred to as anomalies.

Normalization requires a series of steps, each one building on the previous. These steps are called normal form e.g. first, second, and third normal forms which can be abbreviated as 1NF, 2NF and 3NF respectively.

VALIDATE THE LOGICAL MODEL USING NORMALIZATION

The logical design should contain only properly normalized tables. The process of mapping the conceptual model to the logical model may unveil some new attributes or the discovery of new multivalued or composite attributes. Therefore, it is very likely that new attributes may be added to tables or entire new tables added to the logical model. For each table you must ensure that all attributes are fully dependent on the identified primary key and that the tables are in at least third normal form (3NF).

Functional dependency

*Attribute A* is said to be functionally dependent on *attribute B* if for each value of *attribute B* there is only one corresponding value of *attribute A .*For example, *Course Description* is said to be functionally dependent on *Course Number*  if for each value of *Course Number*  there is only one corresponding value of *Course Description*



There are three other more sophisticated normal forms but they are rarely necessary and thus will not be covered in this course.

You will use Normalization to improve your table design as follows:

1. To reduce the space required to store data by eliminating duplicate data in the database.
2. To reduce inconsistent data in the database by storing data only once, thereby, reducing the chance of typographical, spelling, transposition errors, etc.
3. To remove modification anomalies caused by redundancies. These anomalies are referred to as: deletion anomaly, update anomaly, and insertion anomaly are further discussed below:

A **data anomaly** develops when all of the required changes in the redundant data are not made successfully.

**Deletion Anomaly**

This occurs when a user deletes data from a database and unintentionally deletes the only occurrence of that data in the database.

**Update Anomaly**

This usually occurs when there are redundant data (some data stored in more than one place) in the database and a user fails to update some records, or updates records erroneously.

**Insertion Anomaly**

This occurs when a user cannot add data to a database unless it is preceded by the entry of other data.

**Note:** The first two phases of the database development (conceptual data modeling, and logical database design are concerned with the information content of the database.

1. **DISTRIBUTED DATABASE DESIGN PHASE**

The Distributed Database and Physical Database Design phases are usually divided between the System Design phase and the Systems Implementation phase. Both distributed database and physical database designs are concerned with efficient implementation of the database.

This Distributed Database Design phase involves choices about the location of data processes in order to enhance performance by reducing response time, improve availability of data and improve control.

This phase will ***not*** be considered in the project.

1. **PHYSICAL DATABASE DESIGN PHASE**

This phase is concerned with the performance of the database at one computer location only. For a distributed database, physical design considerations are necessary for every location.

This phase yields the following:

1. The internal schema which represents the storage view of the database. It defines files, indexes and data placement.
2. A populated database that is functional.

**Project Deliverables**

**Conceptual modeling requirements**:

For all the entities you wish to store data about you must create ERDs to show the binary relationships between related entities, and then combine these ERDs to give the global view of the database system. All ERDs must be drawn by a modeling tool such as Visio and upload to the dropbox, a pdf or ms-word is also permitted.

**Database and tables requirements:**

1. Create the e-commerce database on oracle1.

Each table must include a primary key and all necessary fields. You must set the field properties as necessary to collect the current data. You must do a table design for each table showing a descriptive column name, data type, size, and constraints where necessary.

1. Use the prefix comm\_ for each of your table names.
2. All products must belong to a category and products can have attributes such as colour or size.
3. Products must include prices.
4. The shopping cart must indicate products, prices and quantities and a status, i.e., checked out or not.
5. Customers must have relevant contact information.
6. Shipping may be to the home address or any other address. There must be a shipping status with a discreet set of values.
7. Populate each table with at least 10 records.
8. Design 5 different queries you need to make your project functional. Use SQL to create the queries.
9. Upload your database script file .sql which will also include your 5 queries to the project dropbox . Also run all on Oracle and provide screen shots for all commands : create tables, constraints and inserts and 5 queries with results. Drop ms word or pdf in dropbox.

**Note:** Any form of plagiarism will be severely dealt with, and no mark will be awarded for the project. Please refer to the College policy outlined in the student’s rights and responsibilities provided to you by the college.