CHAPTER 3: PRINCIPLES AND TECHNIQUES OF DATABASE DESIGN

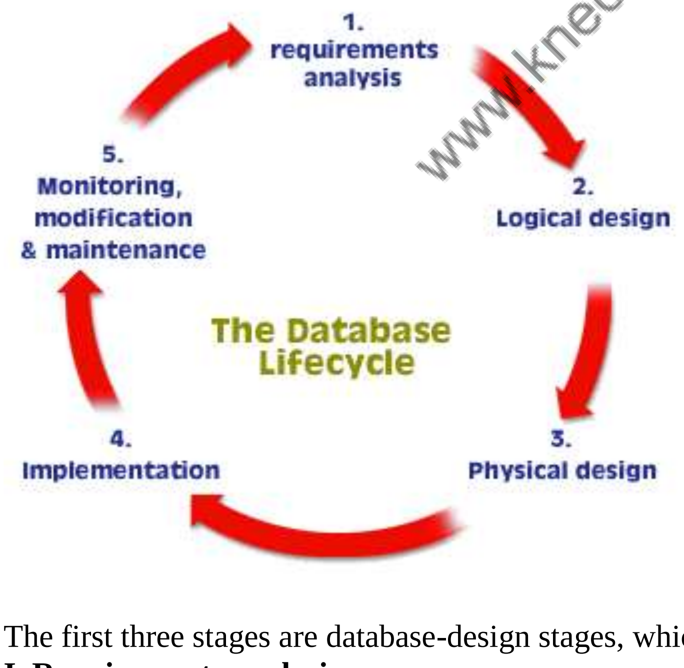
Introduction to Database design Principles

Is the process of producing a detailed data model of a database. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a data definition language, which can then be used to create a database.

The database life cycle (DBLC)

The database life cycle (DBLC) defines the stages involved in getting any type of database off the drawing board and up and running.
In fact, the DBLC never ends because database monitoring, modification, and maintenance are part of the life cycle, and these activities continue long after a database has been implemented.
Put simply, the DBLC encompasses the lifetime of the database.
The five stages in the database life cycle are:

1. Requirements analysis
2. Logical design
3. Physical design
4. Implementation
5. Monitoring, modification, and maintenance



The first three stages are database-design stages, which are briefly described below.
I. Requirements analysis
Requirements Analysis is the first and most important stage in the *Database Life Cycle*.
It is the most labor-intensive for the database designer.

This stage involves assessing the informational needs of an organization so that a database can be designed to meet those needs.

II. Logical design

During the first part of Logical Design, a *conceptual model* is created based on the needs assessment performed in stage one. A conceptual model is typically an *entity-relationship (ER) diagram* that shows the tables, fields, and primary keys of the database, and how tables are related (linked) to one another.

The tables sketched in the ER diagram are then *normalized*. The *normalization* process resolves any problems associated with the database design, so that data can be accessed quickly and efficiently.

1. *conceptual model:* A description of the structure of a database.
2. *entity-relationship (ER) diagram:* A diagram used during the design phase of database development to illustrate the organization of and relationships between data during database design.
3. *normalization:* The process of applying increasingly stringent rules to a relational database to correct any problems associated with poor design.

III. Physical design

The Physical Design stage has only one purpose: to maximize database efficiency.

This means finding ways to speed up the performance of the RDBMS. Manipulating certain database design elements can speed up the two slowest operations in an RDBMS: retrieving data from and writing data to a database.

IV. Implementation and Monitoring, Modification, and Maintenance

The final two stages in the DBLC, Implementation and Monitoring, Modification, and Maintenance, occur after the database design is complete.

The following paragraphs discuss these stages in detail.

<u>*Implementation*</u>

During the implementation stage of the DBLC, the tables developed in the ER diagram (and subsequently normalized) are converted into SQL statements and “fed” into the RDBMS to create a database. By this stage in the DBLC, the *System Administrator* has installed and configured an RDBMS.

<u>*System administrator:*</u>

The person responsible for administering a multi-user computer system; duties range from setting up and configuring system components (e.g., an RDBMS) to performing maintenance procedures (e.g., database backups) on the system.

Certain database design books consider converting an ER diagram into SQL statements to be the final task in the logical-design stage. According to such books, implementation is just a matter of feeding SQL statements into an RDBMS and populating the database with data. The difference is not especially important.

Monitoring, modification, and maintenance
A successfully implemented database must be carefully *monitored* to ensure that it's functioning properly and that it's secure from unauthorized access. The RDBMS usually provides utilities to help monitor database functionality and security.
Database *modification* involves adding and deleting records, importing data from other systems (as needed), and creating additional tables, user views, and other objects and tools. As an organization grows, its *information system* must grow to remain useful.
information system: Interrelated components (e.g., people, hardware, software, databases, telecommunications, policies, and procedures) that input, process, output, and store data to provide an organization with useful information. Well-designed Database

A well-designed database enhances the organization's ability to expand its information system.

Ongoing *maintenance* procedures include periodic database backups, for example, an important and ongoing maintenance procedure. Again, the RDBMS provides utilities to assist in this task.

Basic Design Principles

Avoid redundant information

I.e., keep duplication of data to a minimum. The process to find such a design is called normalization. The normalization process can be formalized to a set of strict rules, and the level of normalization can be expressed in terms of different normal forms, e.g., First Normal Form (1NF), Second Normal Form (2NF), Third Normal Form (3NF), Boyce-Codd Normal Form (BCNF), Fourth Normal Form (4NF), and Fifth Normal Form (5NF). However, in most cases it is enough to follow the simple rule "avoid redundant information" with your common sence. This is the approach in this material.

Make the database structure flexible to meet the future needs

Flexibility here means scalability and such design that new tables or table columns need not to be created after the database has been set up. The aim is to be able to handle all situations just by manipulating data, making new data definitions (creation of new tables or fields) unnecessary.

Define data types and constraints

The three basic data type categories are string, numeric, and date/time*. Choose numeric for fields including values that are measures. For example, phone numbers and zip codes are not a measures, and the appropriate data type is thus string. Date and time data is not string data, even though it is most often represented as strings to the user. After the data type cathegory has been chosen, next task is to find the appropriate (maximum) length for the data items, for numeric and date/time data types also the accuracy.

The main conciderations with respect to defining constraints:

- is the field value always required, i.e., are NULL values denied (NOT NULL constraint in SQL)**
- should the field values be unique (PRIMARY KEY constraint, UNIQUE constraint)***
- what set of values is allowed (CHECK constraint, FOREIGN KEY constraint)
- is there some more complex constraints or business rules to be set up (ASSERTION constraint, triggers)

* Techically speaking, date and time data types fall into the cathegory of numeric data types.
** Make sure that you understand the difference between empty string and NULL. With string data types they can look the same from the user's point of view, but they behave differently with search conditions. To avoid confusion, some developers advice to deny NULLs in fields with string data type.
*** Table can have only one primary key but many alternate keys, i.e., keys defined with the UNIQUE constraint. Notice that both constraints can involve a set of fields rather than only one field. Make sure that you understand the difference between these two:
a) table has two UNIQUE constraints: UNIQUE(a), UNIQUE(b);
b) table has one UNIQUE constraint: UNIQUE(a,b).

Optimize physical performance

Speed is a very improtant factor in database applications. Indexes help the database management system (DBMS) to search data from the database tables. Indexes are data structures maintained automatically by the DBMS after they have been created. Primary indexes, indexes for the primary key fields, are automatically created. Normally, other indexes have to be created manually. It is not always easy to analyse what indexes are worth creating. Namely, even though indexes speed up search queries*, they slow down update operations (because each update means that also the index has to be updated).
* However, not in all cases. For example, if all rows or relatively very large number of rows in a table are selected to the result set of the query, it is obviously more efficient to search the file from the beginning to the end sequentially without using any indexes.

Another way to speed up data searches is denormalization, in the case that table joins should be made in the query. Denormalization simply means joining two or more tables into one table. Then the join operation hasn't have to be done by the query, which saves time. Again, this means slowing down data update operations, because in denormalized tables there are redundancies: same information has to be updated in many places.

This material concentrates on designing operational databases, databases where update operations take place often, and it is important that updates are made fluently. Thus, denormalization is not normally a good option. If we were designing data warehouse databases, denormalization would be a preferred operation. In data warehouses, data searches to a huge amount of history data have to be carried out as fast as possible, but data updates are batch processed, e.g., at nights, and thus can be more slow.

Types of Database Modeling Techniques

Below is a list of the most common database modeling methods. Do note that, depending on the type of data and end user needs when accessing the database, it’s possible to employ multiple models to create a more sophisticated database design. Of course, in either scenario, the production of database diagrams would be required to establish and maintain high operational standards.

From the below mentioned models the relational model is the most commonly used model for most database designs. But in some special cases other models can be more beneficial.