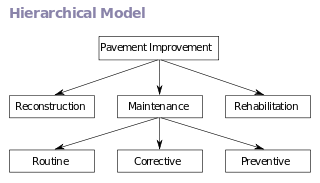
Database Systems Overview Exercises

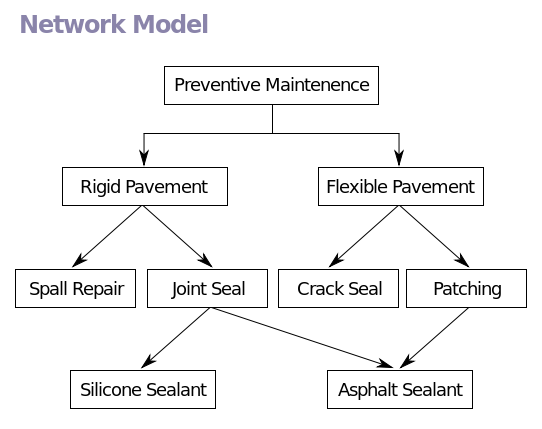
# What database models do you know?

A **database model** is a type of data model (objects with their properties and relationships) that determines the logical structure of a database and fundamentally determines in which manner data can be stored, organized, and manipulated.

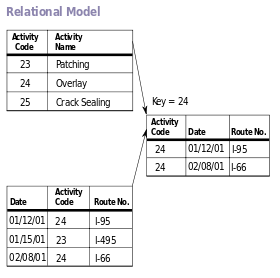
* A **hierarchical database model** is a data model in which the data is organized into a **tree-like** structure. The data is stored as **records** which are connected to one another through **links**. A record is a collection of fields, with each field containing only one value. The **entity type** of a record defines which fields the record contains. One of the main problems of this model is data repetition on various levels.



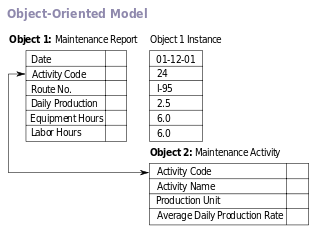
* The **network model** presents a flexible way of representing objects and their relationships. It makes it possible for there to be one-to-many relationships between different levels.



* The **relational model** for database management (RDBMS) is a database model in which all data is represented in tables that can be related to one another. Widely used.



* An **object-oriented database management system** is a database management system in which information is represented in the form of objects as used in object-oriented programming. Object databases are different from relational databases which are table-oriented. Object-relational databases are a hybrid of both approaches.



Source: [Wikipedia](http://en.wikipedia.org/wiki/Database_model)

# Which are the main functions performed by a Relational Database Management System (RDBMS)?

* Creating / altering / deleting tables and relationships between them (database schema);
* Adding, changing, deleting, searching and retrieving of data stored in the tables;
* Support for the SQL language;
* Transaction management (optional);

# Define what is "table" in database terms

The data in RDBMS is stored in database objects called tables. The table is a *collection of related data entries -* a physical representation of an entity or object that is in a tabular format consisting of columns and rows. Columns are the fields of a record or the attributes of an entity. The rows contain the values or data instances; these are also called records or tuples.

# Explain the difference between a primary and a foreign key.

Primary keys and foreign keys are two types of constraints that can be used to enforce data integrity in SQL Server tables.

The **primary key** of a relational table uniquely identifies each record **in the table**. It can either be a normal attribute that is guaranteed to be unique or it can be generated by the DBMS. Primary keys may consist of a single attribute or multiple attributes in combination.

A **foreign key** is a field (or collection of fields) in one table that uniquely identifies **a row of another table**. In other words, a foreign key is a column or a combination of columns that is used to establish and enforce a link between two tables.

# Explain the different kinds of relationships between tables in relational databases.

In**One-To-Many relation** a row in one of the tables can have many matching rows in the second table, but a row in the second table can match only one row in the first table.

Example: Country-Towns: One country has many towns. Each town is in one country only.

In **Many-To-Many relation** many rows from the first table can match many rows in the second and the other way around. To define this type of relation you need a third table whose primary key is composed of the 2 foreign keys from the other 2 table.

Example Students-Courses: A course has many students. A student is signed up for many courses.

In the **One-To-One relation** each row in the first table may match only one row in the second and the other way around. This relationship is very uncommon simply because if you have this type of relation you may as well keep all the info in one single table. Used to model inheritance between tables.

Example: Persons-Student: Every student is a person. Each person can be just one student.

# When is a certain database schema normalized? What are the advantages of normalized databases?

**Database normalization** is the process of organizing the fields and tables of a relational database to minimize redundancy. A database schema is normalized when the data is divided into separate tables with relationships between them, so that it doesn’t contain repetitive data.

The **advantage** that normalization gives is that the data is isolated so that additions, deletions, and modifications of a field can be made in just one table and then propagated through the rest of the database using the defined relationships.

# What are database integrity constraints and when are they used?

**Integrity constraints** ensure data integrity in the database tables. **Data integrity** refers to maintaining and assuring the accuracy and consistency of data over its entire life-cycle, and is a critical aspect to the design, implementation and usage of any system which stores, processes, or retrieves data. Integrity constraints enforce data rules which cannot be violated.

* **Primary key constraint** ensures that the primary key of a table has unique value for each table row.
* **Unique key constraint** ensures that all values in a certain column (or a group of columns) are unique. It is similar to a primary key in that the value in that column for every row of data in the table must have a unique value. Although a primary key constraint is placed on one column, you can place a unique constraint on another column even though it is not actually for use as the primary key.
* **Foreign key constraint** ensures that the value in given column is a key from another table. A foreign key constraint is the main mechanism used to enforce referential integrity between tables in a relational database. A column defined as a foreign key is used to reference a column defined as a primary key in another table.
* **Check constraint** ensures that values in a certain column meet some predefined condition.

Source: <http://www.informit.com/articles/article.aspx?p=1216889&seqNum=4>

# Point out the pros and cons of using indexes in a database.

Pros:

* Used for quick access to a database table specific information. Increase speed of data retrieval (searching).

Cons:

* Use of indexes slows down inserting rows, adding, deleting and modifying data.

# What's the main purpose of the SQL language?

**SQL (Structured Query Language**) is a standardized declarative language for manipulation of relational databases. A special-purpose programming language, designed for managing data held in a relational database management system (RDBMS).

SQL consists of:

* DDL – Data Definition Language

CREATE, ALTER, DROP commands

* DML – Data Manipulation Language

SELECT, INSERT, UPDATE, DELETE commands

# What are transactions used for? Give an example.

Transactions are a sequence of database operations which are executed as a single unit: either all of them execute successfully or none of them is executed at all.

Transactions in a database environment have two main purposes:

* To provide reliable units of work that allow correct recovery from failures and keep a database consistent even in cases of system failure, when execution stops (completely or partially) and many operations upon a database remain uncompleted, with unclear status.
* To provide isolation between programs accessing a database concurrently. If this isolation is not provided, the program's outcome is possibly wrong.

Example:

We need to transfer 100$ from account A to account B

* Account A -= 100;
* Account B +=100;

If something goes wrong between the first and the second operation in the pair we have a problem - either 100$ have disappeared or they have appeared out of nowhere.

A transaction is a mechanism that allows us to mark a group of operations and execute them in such a way that either they all execute (commit) or the system state will be as if they have not started to execute at all (rollback).

beginTransaction;  
Account B += 100;  
Account A -= 100;  
commitTransaction;

This will either transfer 100$or leave both account in the initial state.

# What is a NoSQL database?

A **NoSQL** or **Not Only SQL** database provides a mechanism for storage and retrieval of data that is modelled in means other than the tabular relations used in relational databases. Motivations for this approach include simplicity of design, horizontal scaling and finer control over availability. The data structure (e.g. key-value, graph, or document) differs from the RDBMS, and therefore some operations are faster in NoSQL and some in RDBMS. There are differences though, and the particular suitability of a given NoSQL DB depends on the problem it must solve (e.g., does the solution use graph algorithms?).

NoSQL databases are increasingly used in big data and real-time web applications. NoSQL systems are also called "Not only SQL" to emphasize that they may also support SQL-like query languages.

# Explain the classical non-relational data models.

* Document model - set of documents, e.g. JSON strings;
* Key-value model - set of key-value pairs;
* Hierarchical key-value - hierarchy of key-value pairs;
* Wide-column model - key-value model with schema;
* Object model - set of OOP-style objects;

# Give few examples of NoSQL databases and their pros and cons.

## MongoDB

Document model

**Pros**:

* Open-source;
* Support for concurrent manipulation of data;
* Durability;
* Stored procedures;

**Cons**:

* No triggers;
* No foreign keys;
* No transactions;

## Cassandra

Wide-column model

**Pros**:

* Triggers;
* Support for concurrent manipulation of data;
* Durability;

**Cons**:

* No foreign keys;
* No transactions;
* No stored procedures;

## Redis

Key-value model

**Pros**:

* Stored procedures;
* Support for concurrent manipulation of data;
* Durability;
* Transactions;

**Cons**:

* No triggers;
* No foreign keys;