Problem 1. What database models do you know?

Perform a research (e.g. in Google or Wikipedia) and provide an information about different type of database models. Provide detailed information about one of the database models by your choice different than relational model.

There are several types of database models:

- Hierarchical Model
- Network Model, Relational Model
- Object/Relational Model
- Object-Oriented Model

Hierarchical Model:

The hierarchical data model organizes data in a tree structure. There is a hierarchy of parent and child data segments. This structure implies that a record can have repeating information, generally in the child data segments. Data in a series of records, which have a set of field values attached to it. It collects all the instances of a specific record together as a record type. These record types are the equivalent of tables in the relational model, and with the individual records being the equivalent of rows. To create links between these record types, the hierarchical model uses Parent Child Relationships. These are a 1:N mapping between record types. This is done by using trees, like set theory used in the relational model, "borrowed" from maths. For example, an organization might store information about an employee, such as name, employee number, department, salary. The organization might also store information about an employee's children, such as name and date of birth. The employee and children data forms a hierarchy, where the employee data represents the parent segment and the children data represents the child segment. If an employee has three children, then there would be three child segments associated with one employee segment. In a hierarchical database the parent-child relationship is one to many. This restricts a child segment to having only one parent segment. Hierarchical DBMSs were popular from the late 1960s, with the introduction of IBM's Information Management System (IMS) DBMS, through the 1970s.

Problem 2. Which are the main functions performed by a RDBMS?

Perform a research (e.g. Google or Wikipedia) and provide an information about relational database management systems and which are their main functions.

Main functions of RDBMS:

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

Problem 3. Define what is “table” in database terms.

Perform a research (e.g. Google or Wikipedia) and provide an information about database table. What is table? How information is stored in tables? What is the difference between tables and relations?

A table is a collection of related data held in a structured format within a database. It consists of fields (columns), and rows. A table has a specified number of columns, but can have any number of rows. Each row is identified by one or more values appearing in a particular column subset. The columns subset which uniquely identifies a row is called the primary key. The difference between table and relation is that a table is usually a multiset (bag) of rows where a relation is a set of tuples and does not allow duplicates.

Problem 4. Explain the difference between a primary and foreign key.

Perform a research (e.g. Google or Wikipedia) and provide an information about primary and foreign keys. What is the primary key? Why do we need it? What is the foreign key? What is the purpose of foreign keys? What is the difference between a primary and foreign key?

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 (such as Social Security Number in a table with no more than one record per person) or it can be generated by the DBMS (such as a globally unique identifier, or GUID, in Microsoft SQL Server). 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 simpler words, the foreign key is defined in a second table, but it refers to the primary key in the first table. The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table.

Problem 5. Explain the different kinds of relationships between tables in relational databases.

Perform a research (e.g. Google or Wikipedia) and provide an information about table relationships in relational databases. Provide more information about every relationship.

In most cases, the relationship matches the primary key from one table, which provides a unique identifier for each row, with an entry in the foreign key in the other table. There are three types of relationships between tables. The type of relationship that is created depends on how the related columns are defined.

- One-to-Many Relationship
- Many-to-Many Relationships
- One-to-One Relationships

One-to-Many Relationships

A one-to-many relationship is the most common type of relationship. In this type of relationship, a row in table A can have many matching rows in table B, but a row in table B can have only one matching row in table A.

Many-to-Many Relationships

In a many-to-many relationship, a row in table A can have many matching rows in table B, and vice versa. You create such a relationship by defining a third table, called a junction table, whose primary key consists of the foreign keys from both table A and table B.

One-to-One Relationships

In a one-to-one relationship, a row in table A can have no more than one matching row in table B, and vice versa. A one-to-one relationship is created if both of the related columns are primary keys or have unique constraints.

Problem 6. What is a certain database schema normalized?

Perform a research (e.g. Google or Wikipedia) and provide an information about database normalization. There is a lot of information about database normalization in web.

Database normalization is the process of organizing the attributes and tables of a relational database to minimize data redundancy. Normalization involves decomposing a table into less redundant (and smaller) tables but without losing information; defining foreign keys in the old table referencing the primary keys of the new ones. The objective is to isolate data so that additions, deletions, and modifications of an attribute can be made in just one table and then propagated through the rest of the database using the defined foreign keys. A typical example of normalization is that an entity's unique ID is stored everywhere in the system but its name is held in only one table. The name can be updated more easily in one row of one table.

Problem 7. What are the advantages of normalized databases?

Perform a research (e.g. Google or Wikipedia) and provide an information of what we gain by normalization of databases.

The benefits of normalization include:

- Searching, sorting, and creating indexes is faster, since tables are narrower, and more rows fit on a data page.
- You usually have more tables.
- You can have more clustered indexes (one per table), so you get more flexibility in tuning queries.
- Index searching is often faster, since indexes tend to be narrower and shorter.
- More tables allow better use of segments to control physical placement of data.
- You usually have fewer indexes per table, so data modification commands are faster.
- Fewer null values and less redundant data, making your database more compact.
- Triggers execute more quickly if you are not maintaining redundant data.

- Data modification anomalies are reduced.
- Normalization is conceptually cleaner and easier to maintain and change as your needs change.

Problem 8. What are database integrity constraints and when are they used?

Explain the database integrity constrains. What are the constraint types? When are they used? Provide definition for all of them.

Database integrity constraints are used to specify rules for the data in a table. If there is any violation between the constraint and the data action, the action is aborted by the constraint. Constraints can be specified when the table is created (inside the CREATE TABLE statement) or after the table is created (inside the ALTER TABLE statement).

We have the following constraints:

- NOT NULL - Indicates that a column cannot store NULL value
- UNIQUE - Ensures that each row for a column must have a unique value
- PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Ensures that a column (or combination of two or more columns) have an unique identity which helps to find a particular record in a table more easily and quickly
- FOREIGN KEY - Ensure the referential integrity of the data in one table to match values in another table
- CHECK - Ensures that the value in a column meets a specific condition
- DEFAULT - Specifies a default value when specified none for this column

Problem 9. Point out the pros and cons of using indexes in a database?

Perform a research (e.g. Google or Wikipedia) and provide an information about indexes. What is index? What indexing methods exist? How indexes work and what are the pros and cons of using them?

An index is a copy of select columns of data from a table that can be searched very efficiently that also includes a low-level disk block address or direct link to the complete row of data it was copied from. Some databases extend the power of indexing by letting developers create indices on functions or expressions.

The advantages of indexes are as follows:

- Their use in queries usually results in much better performance.
- They make it possible to quickly retrieve (fetch) data.
- They can be used for sorting. A post-fetch-sort operation can be eliminated.
- Unique indexes guarantee uniquely identifiable records in the database.

The disadvantages of indexes are as follows:

- They decrease performance on inserts, updates, and deletes.

- They take up space (this increases with the number of fields used and the length of the fields).
- Some databases will monocase values in fields that are indexed.

You should only create indexes when they are actually needed.

Problem 10. What's the main purpose of the SQL language?

Perform a research (e.g. Google or Wikipedia) and provide an information about SQL language. What type of language is SQL? What is used for? Provide an example with simple SQL query.

SQL is a special-purpose programming language designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS).

Originally based upon relational algebra and tuple relational calculus, SQL consists of a data definition language and a data manipulation language. The scope of SQL includes data insert, query, update and delete, schema creation and modification, and data access control. Although SQL is often described as, and to a great extent is, a declarative language, it also includes procedural elements.

```
SELECT *
FROM  Book
WHERE price > 100.00
ORDER BY title;
```

Problem 11. What are transactions user for? Give an example.

Perform a research (e.g. Google or Wikipedia) and provide a basic information about transactions and what are they used for?

A transaction symbolizes a unit of work performed within a database management system (or similar system) against a database, and treated in a coherent and reliable way independent of other transactions. A transaction generally represents any change in database. 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 are possibly erroneous.

A database transaction, by definition, must be atomic, consistent, isolated and durable. Database practitioners often refer to these properties of database transactions using the acronym ACID.

Transactions provide an "all-or-nothing" proposition, stating that each work-unit performed in a database must either complete in its entirety or have no effect whatsoever. Further, the system must isolate each transaction from other transactions, results must conform to existing constraints in the database, and transactions that complete successfully must get written to durable storage.

Problem 12. What is a NoSQL database?

Perform a research (e.g. Google or Wikipedia) and provide an information about non-relational databases? Give a few examples of NoSQL databases.

A NoSQL (often interpreted as Not only SQL) database provides a mechanism for storage and retrieval of data that is modeled 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 structures used by NoSQL databases (e.g. key-value, graph, or document) differ from those used in relational databases, making some operations faster in NoSQL and others faster in relational databases. The particular suitability of a given NoSQL database depends on the problem it must solve.

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. Many NoSQL stores compromise consistency (in the sense of the CAP theorem) in favor of availability and partition tolerance. Barriers to the greater adoption of NoSQL stores include the use of low-level query languages, the lack of standardized interfaces, and huge investments in existing SQL. Most NoSQL stores lack true ACID transactions, although a few recent systems, such as FairCom c-treeACE, Google Spanner (though technically a NewSQL database), FoundationDB and OrientDB have made them central to their designs. (See ACID and JOIN Support.)

NoSQL databases examples:

Term	Matching Database
Data-Structures server	Redis
Document Store	Lotus Notes,Clusterpoint, Couchbase, CouchDB, MarkLogic, MongoDB, Qizx, XML-databases
Key-Value Cache	Coherence, eXtreme Scale, GigaSpaces, GemFire, Hazelcast, Infinispan, JBoss Cache, Memcached, Repcached, Terracotta, Velocity
Key-Value Store	Flare, Keyspace, RAMCloud, SchemaFree, Hyperdex, Aerospike
Key-Value Store (Eventually-Consistent)	DovetailDB, Dynamo, Riak, Dynomite, MotionDb, Voldemort, SubRecord

Term	Matching Database
Key-Value Store (Ordered)	Actord, FoundationDB, Lightcloud, Luxio, MemcacheDB, NMDB, Scalaris, TokyoTyrant
Object Database	DB4O, Objectivity/DB, Perst, Shoal, ZopeDB,
Tuple Store	Apache River, Coord, GigaSpaces
Wide Columnar Store	BigTable, Cassandra, Druid, HBase, Hypertable, KAI, KDI, OpenNeptune, Qbase