HOMEWORK 1 – DATABASES OVERVIEW

1. **What database models do you know?**

A **database model** is a type of data model that determines the logical structure of a database and fundamentally determines in which manner data can be stored, organized, and manipulated. The most popular example of a database model is the relational model, which uses a table-based format. Common logical data models for databases include:

- Hierarchical database model
- Network model
- Relational model
- Entity–relationship model
    - Enhanced entity–relationship model
- Object model
- Document model
- Entity–attribute–value model
- Star schema

2. **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)

3. **Define what is "table" in database terms.**

A table is a collection of related data held in a structured format within a database. It consists of fields (columns), and rows.

In relational databases and flat file databases, a table is a set of data elements (values) using a model of vertical columns (which are identified by their name) and horizontal rows, the cell being the unit where a row and column intersect. A table has a specified number of columns, but can have any number of rows. Each row is identified by the values appearing in a particular column subset which has been identified as a unique key index.

Table is another term for relation; although there is the difference in that a table is usually a multiset (bag) of rows where a relation is a set and does not allow duplicates. Besides the actual data rows, tables generally have associated with them some metadata, such as constraints on the table or on the values within particular columns.

The data in a table does not have to be physically stored in the database. Views are also relational tables, but their data are calculated at query time. Another example are nicknames, which represent a pointer to a table in another database.

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

- Primary key is a column of the table that uniquely identifies its rows (usually its is a number)
- Two records (rows) are different if and only if their primary keys are different

- The primary key can be composed by several columns (composite primary key)
- Relationships between tables are based on interconnections: primary key / foreign key
- The foreign key is an identifier of a record located in another table (usually its primary key)
- By using relationships we avoid repeating data in the database
    - In the last example the name of the country is not repeated for each town (its number is used instead)
- Relationships have multiplicity:
    - One-to-many – e.g. country / towns
    - Many-to-many – e.g. student / course
    - One-to-one – e.g. example human / student

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

Relationship one-to-many (or many-to-one)
- A single record in the first table has many corresponding records in the second table
- Used very often

Relationship many-to-many
- Records in the first table have many corresponding records in the second one and vice versa
- Implemented through additional table

Relationship one-to-one
- A single record in a table corresponds to a single record in the other table
- Used to model inheritance between tables

6. **When is a certain database schema normalized? What are the advantages of normalized databases?**

Normalization of the relational schema removes repeating data.
Non-normalized schemas can contain many data repetitions

- 1-st Normal Form
    - Data is stored in tables
    - Fields in the rows are atomic (inseparable) values
    - There are no repetitions within a single row
    - A primary key is defined for each table
- 2-nd Normal Form
    - Retains all requirements of 1-st Normal Form
    - There are no columns that do not depend on part of the primary key (if it consists of several columns)
- 3-rd Normal Form
    - Retains all requirements of 2-nd Normal Form
    - The only dependencies between columns are of type "a column depends on the PK"
- 4-th Normal Form
    - Retains all requirements of 3-rd Normal Form
    - There is one column at most in each table that can have many possible values for a single key (multi-valued attribute)

7. **What are database integrity constraints and when are they used?**

- Integrity constraints ensure data integrity in the database tables
    - 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
- Foreign key constraint
  - Ensures that the value in given column is a key from another table
- Check constraint
  - Ensures that values in a certain column meet some predefined condition

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

- Indices speed up searching of values in a certain column or group of columns
  - Usually implemented as B-trees
- Indices can be built-in the table (clustered) or stored externally (non-clustered)
- Adding and deleting records in indexed tables is slower!
  - Indices should be used for big tables only (e.g. 50 000 rows)

9. **What's the main purpose of the SQL language?**

SQL (/ˈɛs kjuː ˈɛl/, or /ˈsiːkwəl/; **Structured Query Language**) is a special-purpose programming languagedesigned for managing data held in a relational database management system (RDBMS).

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, adeclarative language (4GL), it also includes procedural elements.

SQL was one of the first commercial languages for Edgar F. Codd'srelational model, as described in his influential 1970 paper, "A Relational Model of Data for Large Shared Data Banks." Despite not entirely adhering to the relational model as described by Codd, it became the most widely used database language.

SQL became a standard of the American National Standards Institute(ANSI) in 1986, and of the International Organization for Standardization(ISO) in 1987. Since then, the standard has been enhanced several times with added features. Despite these standards, code is not completely portable among different database systems, which can lead to vendor lock-in. The different makers do not perfectly adhere to the standard, for instance by adding extensions, and the standard itself is sometimes ambiguous.

10. **What are transactions used for? Give an example.**

A **transaction** comprises 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. Transactions in a database environment have two main purposes:

1. 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.
2. 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.

- Example:
  - A bank transfer from one account into another (withdrawal + deposit)
  - If either the withdrawal or the deposit fails the entire operation should be cancelled

11. **What is a NoSQL database?**

A **NoSQL** or **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 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. 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 and FoundationDB, have made them central to their designs.

12. **Explain the classical non-relational data models.**

Definition

A non-relational database is a database that does not incorporate the table/key model that relational database management systems (RDBMS) promote. These kinds of databases require data manipulation techniques and processes designed to provide solutions to big data problems that big companies face. The most popular emerging non-relational database is called NoSQL (Not Only SQL).

Explanation

Most non-relational databases are incorporated into websites such as Google, Yahoo!, Amazon and Facebook. These websites introduce a slew of new applications every single day with millions and millions of users, so they would not be able to handle large traffic spikes with existing RDBMS solutions. Since RDBMS cannot handle the problem, they've switched to a new kind of DBMS that is capable of handling Web scale data in a non-relational way.

An interesting aspect of a non-relational database such as NoSQL is scalability. NoSQL uses the BASE system (basically available, soft-state, eventually consistent). Non-relational databases forgo the table form of rows and columns relational databases use in favor of specialized

frameworks to store data, which can be accessed by special query APIs. Persistence is an important element in these databases. To enable fast throughput of vast amounts of data the best option for performance is "in memory," rather than reading and writing from disks.

Relational databases use the ACID system, which ensures consistency of data in all situations of data management but obviously takes longer to process because of all those relations and its branching nature. However, the BASE system loosened up the requirements on consistency to achieve better availability and partitioning for better scalability.

13. **Give few examples of NoSQL databases and their pros and cons.**

Examples:

- Flat file
  - CSV or other delimited data
  - spreadsheets
  - /etc/passwd
  - mbox mail files
- Hierarchical
  - Windows Registry
  - Subversion using the file system, FSFS, instead of Berkley DB

Advantages:

Plain text files in a filesystem
- Very simple to create and edit
- Easy for users to manipulate with simple tools (i.e. text editors, grep etc)
- Efficient storage of binary documents

XML or JSON files on disk
- As above, but with a bit more ability to validate the structure.

Spreadsheet / CSV file
- Very easy model for business users to understand

Subversion (or similar disk based version control system)
- Very good support for versioning of data

Berkeley DB (Basically, a disk based hashtable)
- Very simple conceptually (just un-typed key/value)
- Quite fast
- No administration overhead
- Supports transactions I believe

Amazon's Simple DB
- Much like Berkeley DB I believe, but hosted

Google's App Engine Datastore
- Hosted and highly scalable
- Per document key-value storage (i.e. flexible data model)

CouchDB

- Document focus
- Simple storage of semi-structured / document based data

Native language collections (stored in memory or serialised on disk)

- Very tight language integration

Custom (hand-written) storage engine

- Potentially very high performance in required uses cases