Relational Databases and SQL

Mashenkov Timofei
mashfeii

March 21, 2025

1. Relational Databases

Relational database is structured table-based data storage. It consists of several entities:

- **Table** - a collection of data organized in rows and columns with its own name.
- **Column** - a set of data values of a particular type.
- **Row** - a single record in a table.
- **Data** - actual values stored in a table in `column x row` place.
- **Primary Key** - a unique identifier for each row in a table.

Name	Surname	Age
Timofei	Mashenkov	20
John	Doe	30
Jane	Doe	25

Table 1: Possible database view

1.1. Normalization Forms

Normalization is a process of organizing data in a database. It is done to reduce redundancy and improve data integrity. There are several normalization forms:

- **First Normal Form (1NF)** - each column should contain atomic (non-splitting) values, and there should be no repeating groups (attributes).
- **Second Normal Form (2NF)** - the table should be in 1NF and all non-key attributes should be fully functional dependent on the primary key.
- **Third Normal Form (3NF)** - the table should be in 2NF and all non-key attributes should be non-transitively dependent on the primary key (depend only on key attributes).
- **Boyce-Codd Normal Form (BCNF)** - the table should be in 3NF and for every functional dependency, the left-hand side should be a super key.
- **Fourth Normal Form (4NF)** - the table should be in BCNF and there should be no multivalued dependencies.
- **And more...**

1.2. Tables Relationships

- **One-to-One** - each record in the first table is related to one record in the second table.
- **One-to-Many** - each record in the first table is related to one or more records in the second table.
- **Many-to-Many** - each record in the first table is related to one or more records in the second table, and each record in the second table is related to one or more records in the first table.

```sql
CREATE TABLE grades
(
  grade_id INT PRIMARY KEY, -- Unique identifier
  student_id INT NOT NULL, -- Foreign key
  course_id INT NOT NULL, -- Foreign key
  grade DECIMAL(3, 1) NOT NULL CHECK (grade >= 0 AND grade <= 5) -- Grade value from 0 to 5
  grade_date DATE NOT NULL DEFAULT CURRENT_DATE -- Date of the grade

  FOREIGN KEY (student_id) REFERENCES students(student_id) ON DELETE CASCADE,
  -- Connection with student

  FOREIGN KEY (course_id) REFERENCES courses(course_id) ON DELETE CASCADE
  -- Connection with course
);
```

Listing 1: Creating tables in SQL

1.3. Joins

- `LEFT JOIN` - returns all records from the left table and the matched records from the right table.
- `RIGHT JOIN` - returns all records from the right table and the matched records from the left table.
- `INNER JOIN` - returns only the records that have matching values in both tables.
- `OUTER JOIN` - returns all records when there is a match in either left or right table.
- `FULL JOIN` - returns all records when there is a match in either left or right table.

2. Transactions

Transaction is a sequence of operations performed as a single logical unit of work. It is a set of SQL statements that are executed as a single unit. Transactions are used to ensure data integrity and consistency.

2.1. ACID

A set of properties that guarantee that database transactions are processed reliably.

- **Atomicity** - all operations in a transaction are completed successfully or none of them are.
- **Consistency** - the database remains in a consistent state before and after the transaction.
- **Isolation** - transactions are executed independently and do not interfere with each other.
- **Durability** - changes made by a transaction are permanent and cannot be lost.

2.2. Isolation Levels

A property that defines how transactions interact with each other.

- **Read Uncommitted** - concurrent transactions can read uncommitted data.
- **Read Committed** - concurrent transactions can read only committed data.
- **Repeatable Read** - Read Committed + the same `SELECT` inside single transaction guarantees the same result.
- **Serializable** - dependent transactions are executed sequentially.

Dirty Read:
Reading of yet uncommitted data. First transaction modifies data, but the second transaction reads it before the first transaction commits.
Lost Update:
Update for the same data within both concurrent transactions. First transaction modifies data, and the second transaction modifies it again. As a result one transaction is rewritten.
Non-Repeatable Read:
Can be observed when we commit the first transaction that updated the data, and select values within the second one. Based on updated data the second transaction could be rolled back since transaction needs to be executed again with new values.
Phantom Read:
Can be observed when we commit the first transaction that inserted new data, and select values within the second one. Based on new data the second transaction could be rolled back since transaction needs to be executed again with new values.
Serialization Anomaly:
Unpredictable result after concurrent transactions' execution, which can be changed based on transactions' execution order.