Laboratory Exercise №3

Creation and Maintenance of Database Structure

Introduction to the problem

Creating and maintaining data structures for a database involves a set of operations essential to ensure the efficiency, integrity, and accessibility of the data. Here's a breakdown of the main operations related to this process:

- Database Schema Design
- Table Creation and Modification
- Indexing
- Views and Stored Procedures
- Constraints and Data Integrity
- Data Import and Export
- Data Partitioning
- Performance Tuning and Optimization
- Monitoring and Maintenance
- Security and Access Control

Each of these operations is crucial for effective database management, with the goal of supporting efficient data access, ensuring data integrity, and enabling scalability as the database grows in size and complexity. For the purposes of the current laboratory exercise we will concentrate our efforts over database schema design, table creation and modification, and data integrity and constrains.

Schema design involves defining tables and their relationships within the database. This includes identifying entities (tables) and defining primary and foreign keys to ensure data integrity. Data types are specified for each attribute, optimizing storage and performance. Normalization rules are followed to organize data and reduce redundancy. In cases where performance is prioritized, some databases use denormalization, duplicating certain data to simplify queries.

Tables are created according to the schema, with each column specified by its data type, constraints, and indexes. As data requirements evolve, tables might need to be modified, which involves adding, removing, or changing columns through ALTER TABLE commands. If a table is no longer needed, it can be removed, freeing up storage and maintaining an organized schema.

Constraints like primary keys, foreign keys, and unique constraints enforce data integrity within the database. These constraints might need to be added, modified, or removed to align with changing data needs or business rules. For instance, foreign key constraints can be adjusted to enforce updated relationships or validations.

In addition, understanding and carefully selecting data types for each table and attribute is essential to ensure a database is efficient, accurate, and scalable. **Proper data type selection optimizes storage, enhances performance, and simplifies maintenance, resulting in a more reliable and effective database system**. Understanding data types when designing tables and attributes in a database is crucial for the following reasons:

- **Efficient Storage Management.** Choosing appropriate data types helps use storage resources efficiently. For example, using an INT for a numeric ID instead of a BIGINT (which requires more

storage) reduces the database size. In large databases, even small savings per row can add up to significant storage savings.

- **Improved Query Performance.** Data types influence how the database engine processes and retrieves data. Optimized data types reduce computational overhead and improve query performance. For instance, using an integer type for numbers rather than text allows faster calculations and comparisons.
- **Data Integrity and Accuracy.** Data types help maintain data accuracy and integrity by enforcing constraints. For example, using a DATE type ensures only valid dates are entered, reducing the risk of errors like "January 40th." Choosing a DECIMAL type for currency values preserves exact values without rounding errors that may occur with floating-point types.
- **Simplified Data Validation.** Data types allow the database to automatically validate data as it's entered. For instance, a VARCHAR(50) field ensures values don’t exceed 50 characters, reducing the need for extra code or checks. This built-in validation reduces errors and simplifies application logic.
- **Consistent Data Interpretation.** Each data type represents data in a specific way, ensuring consistency. For example, an INT field will consistently represent whole numbers, while VARCHAR represents text. This consistency avoids ambiguity in data processing and interpretation across the database.
- **Optimized Indexing and Searching.** Data types impact indexing efficiency. Numeric indexes, for instance, are faster to search and require less storage than text-based indexes. Understanding data types helps you choose those that optimize indexing and search capabilities, particularly in large or heavily queried tables.
- **Compatibility with Application Logic.** Database applications often have specific data handling requirements, and compatible data types make interaction with the database smoother. For instance, if an application uses a particular date format, choosing compatible data types reduces the need for data transformation, simplifying code and improving performance.
- **Easier Maintenance and Scalability.** Carefully chosen data types make it easier to scale the database as requirements change. For example, using VARCHAR for a name field (rather than a fixed CHAR length) allows names of different lengths to be stored without waste. This adaptability simplifies future changes or expansions to the schema.
- **Error Prevention and Reduced Data Anomalies.** Choosing the right data types prevents errors from incorrect data entries. For example, a BOOLEAN data type restricts values to TRUE or FALSE, avoiding unintended values. This helps prevent anomalies and reduces the need for extensive data cleaning.
- **Data Type Constraints and Database Efficiency.** Some databases use data types to manage constraints like default values, uniqueness, and nullability, which impact database performance. For instance, specifying data as NOT NULL ensures fields are never empty, which is crucial for primary keys. These constraints help enforce data quality and make operations more efficient by preventing empty or null values where they shouldn't exist.

SQL constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table. The following constraints are commonly used in SQL:

- `NOT NULL` - Ensures that a column cannot have a NULL value
- `UNIQUE` - Ensures that all values in a column are different

- `PRIMARY KEY` - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- `FOREIGN KEY` - Prevents actions that would destroy links between tables
- `CHECK` - Ensures that the values in a column satisfies a specific condition
- `DEFAULT` - Sets a default value for a column if no value is specified
- `CREATE INDEX` - Used to create and retrieve data from the database very quickly

You may find helpful the following online sources of information about SQL syntax related to data types in MySQL, :

- MySQL Create DB, https://www.w3schools.com/mysql/mysql_create_db.asp
- MySQL Create Table, https://www.w3schools.com/mysql/mysql_create_table.asp
- MySQL Constraints, https://www.w3schools.com/mysql/mysql_constraints.asp
- MySQL Data Types (Version 8.0), https://www.w3schools.com/mysql/mysql_datatypes.asp
- MySQL Insert, https://www.w3schools.com/mysql/mysql_insert.asp
- MySQL Update, https://www.w3schools.com/mysql/mysql_update.asp
- MySQL Delete, https://www.w3schools.com/mysql/mysql_delete.asp

Requirements for documentation of the work done by students

Each student must prepare and send a text document (in DOCX or PDF format) with designed E-R Diagram (created with draw.io online tool!) in simplified and complex form for each case study at the laboratory exercise, XML document containing E-R Diagram, SQL sentences for each question from tasks of the current laboratory exercise. **Prepared document must be sent on the e-mail of your teaching assistant no later than a day before your next laboratory exercise.**

Case Study „SOFIA Airport ”

Introduction. You are an employee of an IT company that won a public procurement for creation of database for the purposes of management and maintenance of flight information related to Sofia Airport. The Sofia Airport is the main international airport of Bulgaria, located 10 km east of the center of the capital Sofia. In 2019 the airport surpassed 7 million passengers for the first time. The airport serves as the home base for BH Air, Bulgaria Air, European Air Charter and GullivAir, and as a base for both Ryanair and Wizz Air. that could manage operates in the vicinity of the city of Boston and, as such, you oversee the delivery of packages, documents and mail across the city.

As a part of the IT development team, you must examine the preliminary information received by employees of Sofia airport and to create a database from scratch by writing a set of SQL statements. **The implementation details are up to you, though you should minimally ensure your database meets the airport’s requirements and that it can represent the given sample data.** Also, you must create a simplified and complex E-R Diagram to describe a business requirement in proper way and to communicate easier with company staff. You also have access to publicly available information sources at the Internet, including:

- General information about Sofia Airport, https://en.wikipedia.org/wiki/Sofia_Airport
- Departures SOF Connect, https://sofia-airport.eu/en/flights/departures/

The preliminary information about each of discussed entities is as follow:

Passengers. When it comes to, SOF airport just need to have the essentials in line: the first name, middle name, last name, and age.

Check-Ins. When passengers arrive at SOF Airport, they'll often "check in" to their flights. That's them telling us they're here and all set to board. Airport would like to keep a tidy log of such moments, including:

- The exact date and time at which our passenger checked in
- The flight they are checking in for, of course. Can't lose track of where they're headed, now can we?

Airlines. SOF Airport is a hub for many domestic and international airlines. Company wants to track:

- The name of the airline
- Airline's website and contact phone/phones
- Airline' IATA and ICAO codes (See also: https://en.wikipedia.org/wiki/List_of_airline_codes)
- Terminals on the airport where the airline operates. We have 2 terminals (T1 and T2)

Flights. SOF serves as many as 80 flights daily. To ensure that their passengers are never left wondering, we need to give them all the critical details about their flight. Here's what we'd like to store:

- The flight number. For example, "LH1703". Just know that sometimes they re-use flight numbers.
- The airline operating the flight. You can keep it simple and assume one flight is operated by one airline.
- Terminal
- Desks for check-in luggage
- Gate
- The code of the airport they're departing from. For example, "SOF".
- The code of the airport they're heading to
- The name of the city they're heading to
- The expected departure date and time (to the minute, of course 😉)
- The expected arrival date and time, to the very same accuracy

Task 1. Create simplified and complex E-R Diagram using **`draw.io`** online tool. Draw a relationship between entities considering cardinality and modality of each relation. For the complex representation try to choose proper names for each attribute of entity taking into account business requirement and domain. Choose a proper data types to store attribute information for each entity.

Task 2. Create a database schema under "sof_airport" name. Use UTF8 encoding.

Task 3. Create tables for each entity with all attributes that arise from business requirements. Try do consider a proper constrains for each attribute, answering questions about their function, connection with attributes from other entities and also uniqueness, repetition, and etc. **Are there any candidates for primary and foreign keys or you must add additional field?**

Task 4. Add some sample data to your tables:

- A passenger, Georgi Georgiev Ivanov, who is 23 years old
- An airline, DEUTSCHE LUFTHANSA AG, which operates out T1 and T2 terminals

- A flight, LH1427, which is expected to depart from SOF on November 3rd, 2024 at 14:15 PM, terminal T2, registration desks 18-24, gate B5 and arrive at FRA FRANKFURT on November 3rd, 2024 at 15:55 PM
- A check-in for Georgi Georgiev Ivanov, for LH1427, on November 3rd, 2024 at 12:03 PM
- A passenger, Ivanka Dimitrova Stoyanova, who is 44 years old
- An airline, BULGARIA AIR, which operates out T1 and T2 terminals
- A flight, FB471, which is expected to depart from SOF on November 3rd, 2024 at 14:55 PM, terminal T2, registration desks 1-7, gate B4 and arrive at MAD MADRID on November 3rd, 2024 at 17:40 PM
- A check-in for Ivanka Dimitrova Stoyanova, for FB471, on November 3rd, 2024 at 12:17 PM
- A passenger, Stefan Borislavov Todorov, who is 61 years old
- An airline, WIZZ AIR, which operates out T1 and T2 terminals
- A flight, W64377, which is expected to depart from SOF on November 3rd, 2024 at 17:55 PM, terminal T1, registration desks 9-14, gate 3 and arrive at CPH COPENHAGEN on November 3rd, 2024 at 19:50 PM
- A check-in for Stefan Borislavov Todorov, for W64377, on November 3rd, 2024 at 15:11 PM