10. ANSWERS TO THE TYPICAL MYSQL QUESTIONS

10.1 Database creation using SQL language

At the command prompt of the MySQL server, enter the command to access the MySQL server command line:

```
mysql -u root -p
```

Then you need to enter the password.

At the command prompt of the MySQL server, enter the command to connect to a MySQL server using a specific username and password:

```
mysql -u username -p password
```

Then you need to enter the password.

You can create a database with the following command:

```
create database and database name;
```

You can check the creation of the database with the help of the command:

```
show databases;
```

This command will show what databases exist on your computer.

The table is created using the commands below. And binding tables is done using a foreign key, which is specified when creating a table:

```
create table table_name (
    a list of fields that will be in the table;
    foreign key (field_name) references table_name (field_name)
);
```

Modification the table is done by a command:

```
alter table name_table_;
```

The table is deleted using the command:

```
drop table name table;
```

You can use the command to check created or deleted tables:

```
show tables;
```

10.2 Data manipulation using SQL language: insert, update, and delete

The structure and examples of using the INSERT command:

Syntax:

INSERT INTO table_name (field1, field2 ...) VALUES (value1, value2 ...);

INSERT INTO table_name (field1, field2 ...) VALUES (value1, value2 ...), (value1, value2 ...) ... '';

Example:

INSERT INTO workers SET name = 'Alex', age = 23, salary = 500;

INSERT INTO workers (name, age, salary) VALUES ('Alex', 23, 500), ('John', 30, 1000);

The structure and examples of using the UPDATE command:

Syntax:

UPDATE table_name SET field1 = value1, field2 = value2, field3 = value3;

Example:

UPDATE workers SET age = 30, salary = 1000 WHERE id = 1;

The structure and examples of using the DELETE command:

Syntax:

DELETE FROM table_name WHERE condition;

Example:

DELETE FROM workers WHERE id = 2;

Update all records in the database table:

Syntax:

UPDATE table_name SET field = value;

Delete all records from the database table:

Syntax:

DELETE FROM table_name;

10.3 Data manipulation using SQL language: select queries and their basic features

SQL statement used to select data from one or more tables:

SELECT <* (all fields) or field list>
FROM <the name of the table from which the data is required>

An SQL SELECT query to output all the columns of the table:

```
SELECT * FROM table_name;
```

Select records that meet the search criteria:

```
SELECT * FROM table_name WHERE conditions;
```

Where indicates that the conditions under which you want to filter the result of the select statement will be specified below.

Use the keyword to avoid repetition DISTINCT.

The ORDER BY construct allows you to arrange the selected records in ascending or descending order of the values of any column or combination of columns, regardless of whether these columns are present in the results table or not.

The ORDER BY construct allows you to arrange the selected records in descending order (DESC) of the values of any column.

That is, the following statement will look like:

```
SELECT * FROM table ORDER BY field DESC;
```

The result set could be limited by the LIMIT keyword:

```
SELECT * FROM table LIMIT 4;
```

This example indicates that the limit has been set and the first four entries will be displayed.

With the help of the operator GROUP BY groups of rows having the same value in the specified column are formed.

We can say that a query in which the GROUP BY construct is present is called a grouping query, because it groups the data obtained as a result of the SELECT operation, after which a single summary string is created for each individual group.

Aggregation functions, their purpose and main features:

- COUNT - returns the number of values in the specified column;
- MAX - returns the maximum value in the specified column;
- SUM - returns the sum of the values in the specified column;
- AVG - returns the average value in the specified column;
- MIN - returns the minimum value in the specified column.

All these functions operate on values in a single column of the table and return a single value. The COUNT, MIN, and MAX functions apply to both numeric and non-numeric fields, while the SUM and AVG functions apply only to numeric fields.

You can assign a new table name by querying:

```
ALTER TABLE table RENAME TO new_name;
```

HAVING has the same purpose as WHERE – filtering, but with some differences. HAVING is applied to row groups (after the GROUP BY command).

In SQL there are arithmetic operations that allow you to manipulate numbers:

- adding (+);
- subtraction (-);
- multiplication (*);
- division (/);
- the remainder of the division (%).

Examples of use:

```
SELECT 2 + 5;

SELECT 5 - 2;

SELECT 2 * 5;

SELECT 10/2;

select 11% 2;
```

Standard logical operators in SQL are AND, OR, and NOT:

- AND – takes two values as arguments and results in the truth if they are both true.
- OR – takes two Boolean expressions as arguments and evaluates the result as true, if at least one of them is true.
- NOT – takes a single logical expression as an argument and changes its value from true to false or false to true.

Examples of use:

```
SELECT name FROM Users WHERE name = 'Jack' OR name = 'John';
SELECT name FROM Users WHERE name = 'Alex' AND age> 18;
SELECT name FROM Users WHERE name = 'John' AND NOT age > 18;
```

The following comparison operators are present in SQL:

- = (equal to);
- <=> (equivalence, return the truth when comparing);
- (more than);
- <(less than);
- = (greater than or equal to);
- <= (less than or equal to);
- or! = (not equal to).

Examples of use:

```
SELECT fields FROM table WHERE field_1 = field_2;
SELECT fields FROM table WHERE field_1 <=> field_2;
SELECT fields FROM table WHERE field_1> field_2;
SELECT fields FROM table WHERE field_1 <field_2;
SELECT fields FROM table WHERE field_1> = field_2;
SELECT fields FROM table WHERE field_1 <= field_2;
SELECT fields FROM table WHERE field_1 <> field_2;
SELECT fields FROM table WHERE field_1! = Field_2;
```

The MONTH function extracts the month number from the date:

```
SELECT *, MONTH (date) as month FROM workers;
```

The YEAR function extracts the year number from the date:

```
SELECT *, YEAR (date) as year FROM workers;
```

The IFNULL function checks the value of an expression. If it is NULL, the function returns the value passed as the second parameter:

SELECT FirstName, LastName, IFNULL(Phone, 'not specified') AS Phone FROM Clients;

The CONCAT function is designed to compile rows when sampling from a database. Rows are usually table fields:

```
SELECT *, CONCAT (age, name, salary) as concat FROM workers;
```

The RTRIM function removes all empty characters to the right of the text:

```
SELECT RTRIM ('Removes trailing spaces.');
```

The SUBSTRING function cuts a part of the line from the specified element to the specified:

```
SELECT x = SUBSTRING ('abcdef', 2, 3);
```

The IF function returns one of two values depending on the result of the conditional expression:

```
SELECT ProductName, Manufacturer, IF (ProductCount > 3, 'Many Items', 'Low Items')
FROM Products;
```

The UNION command combines data from several tables into one when sampling:

```
SELECT *FROM table_name1 WHERE condition
UNION
SELECT *FROM table_name2 WHERE condition
```

10.4 Creation and usage of views

Views are virtual tables which contents consist of data from other tables.

Benefits of views include data protection, as well as the ability to save your time by memorizing the most commonly used complex queries.

Disadvantages of views include the ability to open read-only, as well as the inability to use UNION in the query.

The CREATE VIEW statement is used to create views.

You can delete a view using the DROP VIEW statement.

To check for availability of a view in the database use:

```
SELECT * FROM view_name;
```

or

```
SHOW TABLES;
```

Each created view is added to the list of tables.

The query after SELECT specifies the columns to be created in the submitted. You can use aliases to change the name.

A vertical view is a view that results in only selected columns of tables. There is no line restriction.

A horizontal view is a view that results in only selected query strings. The number of columns in the table is saved.

10.5 Creation and usage of stored procedures and triggers

Stored procedures are database objects in which the algorithm is embedded as a set of SQL statements.

Some of the benefits of stored procedures are:

- the procedure request is usually cached, which allows you to reuse it over and over again without having to re-prepare it;
- fewer worries about who can call a stored procedure than control over who can access those tables or table rows.

You can create a stored procedure using:

```
CREATE PROCEDURE procedure_name ()
```

To determine the input or output parameters of a stored procedure use:

```
CREATE PROCEDURE procedure_name (IN input_parameters, OUT output_parameters)
```

The IF operator is designed to control the flow of data.

The BEGIN operator indicates that the body of the procedure or trigger will be described below

The END operator indicates that the body or trigger body description is complete.

Triggers are a special type of stored procedure that is called automatically when performing a certain action on a table or view, in particular, when adding, modifying or deleting data, ie when executing commands INSERT, UPDATE, DELETE.

Some advantages of triggers are:

- automatically apply data constraints to make sure users enter only valid column values;

- automatic notification of database changes using event notification tools.

The trigger communicates with the table using the ON statement:

CREATE TRIGGER trigger_name… ON table_name

A trigger is a procedure that is stored but tied to a table content change event. There are three possible table change events to which you can bind a trigger: change the contents of the table using the insert, delete, and update statements.

The condition when the trigger must be executed is specified when creating the trigger:

CREATE TRIGGER trigger_name AFTER / BEFORE INSERT ON table_name

The prefixes NEW and OLD are used for:

- obtaining valid defaults in some conditions;
- checks and, if necessary, conversion of user input data;
- get keys and values to perform automatic updates in other tables;
- implementation of auto-incremental keys by means of generators.

The SET statement allows you to assign a new value to local variables.

The stored procedure could be removed using the command:

DROPP ROCEDURE procedure_name;

The trigger could be removed using the command:

DROP TRIGGER trigger_name;

10.6 Basics of data integrity control mechanisms

When creating a CREATE TABLE or ALTER TABLE command, the use of the ON DELETE and ON UPDATE constructs is optional.

The ON DELETE statement determines exactly how certain queries will be deleted.

The ON UPDATE statement determines exactly how certain queries will be updated.

After the ON DELETE and ON UPDATE constructs, the following constructs can be given:

NO ACTION

CASCADE

SET NULL

When using the CASCADE relational integrity mechanism, the entries in the table will be deleted or cascaded. That is, if you change or delete values from the primary key of the parent table, the corresponding values in the foreign key of the child table are changed or deleted according to the primary key.

When using the relational integrity mechanism SET NULL, if you change or delete a value from the primary key of the parent table, the corresponding value in the foreign key of the child table is NULL.

When using the relational integrity mechanism NO ACTION , if the value from the primary key of the parent table is changed or deleted, the corresponding value in the foreign key of the child table will remain unchanged.

When using the relational integrity mechanism DEFAULT SET, if you change or delete a value from the parent key's primary key, the corresponding value in the child table's foreign key is set by default.

When using the RESTRICT relational integrity mechanism, the records in the table will be deleted or updated if there are no related records. That is, RESTRICT forbids operations if there are related records and if you change or delete the value from the primary key of the parent table, the action will be prohibited because there is a child table.

You should not use the CASCADE parameter if the table is included in a merge publication that uses logical records. The ON DELETE CASCADE parameter cannot be specified if the table already has an INSTEAD OF trigger for the condition ON DELETE for ON UPDATE.

10.7 Work with transactions

A transaction is a sequence of different SQL queries that run as a whole and is not interrupted by other clients. That is, when transaction requests are executed, no one can access the records.

The transaction is an archive for database queries. It protects your data thanks to the principle of "all or nothing".

MySQL supports seven types of tables. Only three of them are transaction-oriented, extended-type tables:

- InnoDB;
- Berkeley DB;
- Gemini.

MySQL supports seven types of tables. Four of them do not support transactions:

- Heap;
- ISAM;
- Merge;
- MyISAM.

You can disable the automatic transaction completion mode using the system variable AUTOCOMMIT, the value of which is set by the operator SET AUTOCOMMIT = 0.

After disabling the automatic mode transaction completion should be used COMMIT statement to save changes.

After disabling the automatic mode transaction completion should be used ROLLBACK to undo changes made from the moment of the beginning of the transaction.

To enable automatic mode completion of transactions for individual only sequence of operators used START TRANSACTION operator.

SAVEPOINT and ROLLBACK TO SAVEPOINT are operators for working with subtransactions, so these operators can be used with the following types of tables:

- InnoDB;
- Berkeley DB;
- Gemini.

Operator SAVEPOINT is needed to mark the beginning of a subtransaction (the small transaction that makes up the transaction), to mark the rollback point.

ROLLBACK TO SAVEPOINT – this operator is needed to cancel all requests starting from the endpoint.

The main problems that arise in the parallel execution of transactions are divided into the following types:

- Missing changes. This situation can occur if two transactions change the same record in the database at the same time.
- Intermediate data problems. Consider the same problem of simultaneous operation of two operators.
- Problems of inconsistent data. This situation occurs when both transactions change the same row in the database. The state of the database is consistent.

Read Uncommitted (RU) – at this level it is forbidden to change the data by other transactions, if these data are modified before the transaction has ended. However, other transactions are allowed to read unconfirmed data, which is classified as "dirty reading".

Read Committed (RC) – dirty reading is prohibited at this level.

Repeatable Read (RR) – at this level, "dirty reading" and re-reading are prohibited.

Serializable (S) – "phantom reading" is prohibited at this level.

By default (since version 5.5) the type of tables used in MySQL is InnoDB.

InnoDB supports the following levels of transaction isolation:

REPEATABLE READ

READ COMMITED

READ UNCOMMITED

SERIALIZABLE

The default transaction isolation level is REPEATABLE READ.

That is, SELECT does not block anything, reads lines from the snapshot created during the first read in the transaction. Single queries will always return the same result.

And for lock read (SELECT ... FOR UPDATE / LOCK IN SHARE MODE), UPDATE and DELETE lock will depend on the type of condition. If the condition is unique (WHERE id = 42), then only the found index record is blocked. If the condition has a range (WHERE id > 42), the entire range is blocked (gap lock or next-key lock).

10.8 User rights management

Creating a user with this name in the system:

```
CREATE USER user IDENTIFIED BY 'password';
```

The password part is optional, the default password is a blank line. Purpose - to add a password to a specific account.

View all accounts:

```
SELECT host, user, password FROM mysql.user;
```

Create an account:

```
CREATE USER ...
```

Removal account:

```
DROP USER ...
```

Change a username of the account:

```
RENAME USER ...
```

You can use the operator to define certain privileges for the required account GRANT.

Abolition of privileges:

REVOKE ...

There are many privileges, but the main ones are listed in the Table 10.1 below.

Table 10.1

Privilege	**Description**
ALL	A combination of all the privileges.
ALTER	Allows you to edit tables using the ALTER TABLE statement.
ALTER ROUTINE	Allows you to edit or delete the saved procedure.
CREATE	Allows you to create a table using the CREATE TABLE statement.
CREATE ROUTINE	Allows you to create a saved procedure.
CREATE USER	Allows you to work with accounts using operators CREATE USER, DROP USER, RENAME USER and REVOKE ALL PRIVILEGES.
CREATE VIEW	Allows you to create a view using the CREATE VIEW statement.
DELETE	Allows you to delete records using the DELETE statement.
DROP	Allows you to delete tables using the DROP TABLE statement.
EXECUTE	Allows you to perform saved procedures,
INSERT	Allows you to add new records to the table using the operator INSERT,
SELECT	Allows you to fetch tables using the select statement.
UPDATE	Allows you to update the contents of tables using the UPDATE statement.
GRANT OPTION	Allows you to manage the privileges of other users without this privilege the GRANT and REVOKE statements cannot be executed.

There are not so many levels of privilege assignment, so they are all listed in the Table 10.2 below.

Table 10.2

Level	**Description**
ON *. *	Global level – applies to all databases and tables that are part of them, so a user with global authority can access all databases.
ON *	If the current database was not selected with the USE statement, this parameter is equivalent to ON *. *, If the current database is selected, the privileges set by the GRANT statement apply to all tables in the current database.
ON db. *	Database level – this parameter means that privileges are distributed to the db database table.
ON db.tbl	Table level – this parameter means that the privileges extend to the tbl table of the db database.
ON db.tbl	Column level – column level privileges apply to individual columns in the tbl table of the db database. The list of columns is indicated in parentheses through a comma after the keywords SELECT, INSERT, UPDATE.

Check database privileges:

```
SELECT * FROM mysql.db WHERE Db = database name;
```