LESSON 2

READ (SELECT) - BASIC QUERIES, WHERE CLAUSE, SORTING

Ihor Liutak

Content

1. Introduction to SELECT Statement
2. Filtering Data with WHERE Clause
3. Sorting Data with ORDER BY
4. Combining WHERE and ORDER BY
5. Updating Records with UPDATE
6. Deleting Records with DELETE
7. Reviewing INSERT Operations
8. Combining Commands for Data Integrity

Short description

Teach your students how to retrieve data using the SELECT statement. Cover basic queries, filtering with the Where clause, and organizing results with Order By.

Teach your students how to safely update and delete records while maintaining data integrity.

Kurzbeschreibung

Bringen Sie Ihren Schülern bei, wie Sie mit Hilfe der SELECT-Anweisung Daten abrufen können. Behandeln Sie grundlegende Abfragen, das Filtern mit der Where-Klausel und das Organisieren von Ergebnissen mit Order By.

Bringen Sie Ihren Schülern bei, wie Sie Datensätze sicher aktualisieren und löschen und dabei gleichzeitig die Datenintegrität gewährleisten können.

Reading Data with SELECT

Introduction to SELECT Statement:

SELECT [columns] FROM [table_name];

Selecting All Columns: `SELECT * FROM [table_name];`

Selecting Specific Columns: `SELECT column1, column2 FROM [table_name];`

Renaming columns for better readability:

```
SELECT column1 AS 'Alias Name' FROM [table_name];
```

Filtering Data with WHERE Clause

Basic Filtering: `SELECT * FROM [table_name] WHERE column = value;`

Operators for Filtering:

- Comparison: `=`, `>`, `<`, `>=`, `<=`, `<>`.
- Logical: `AND`, `OR`, `NOT`.
- Pattern matching: `LIKE`, `NOT LIKE` (e.g., `'%value%'` for partial matches).

Checking for NULL:

```
SELECT * FROM [table_name] WHERE column IS NULL;
```

Checking for NOT NULL:

```
SELECT * FROM [table_name] WHERE column IS NOT NULL;
```

Sorting Data with ORDER BY

Sorting Data with ORDER BY:

```
SELECT * FROM [table_name]
    ORDER BY column [ASC|DESC];
```

Multiple Column Sorting:

```
SELECT * FROM [table_name]
    ORDER BY column1 ASC, column2 DESC;
```

Combining WHERE and ORDER BY:

```
SELECT * FROM [table_name]
    WHERE column = value
    ORDER BY another_column DESC;
```

Updating Records with UPDATE

Basic Syntax:

```
UPDATE [table_name]
    SET column1 = value1, column2 = value2
    WHERE condition;
```

Update a single column:

```
UPDATE students SET name = 'John Doe' WHERE student_id = 1;
```

Update multiple columns:

```
UPDATE students SET email = 'john@example.com',
enrollment_date = '2024-01-01' WHERE student_id = 1;
```

Ensuring Integrity: *Always use a WHERE clause to avoid updating all rows unintentionally.*

Deleting Records with DELETE

Basic Syntax:

```
DELETE FROM [table_name] WHERE condition;
```

Delete a single row:

```
DELETE FROM students WHERE student_id = 1;
```

Delete multiple rows:

```
DELETE FROM students WHERE enrollment_date < '2023-01-01';
```

Cautions: *Always use a WHERE clause to avoid deleting all rows*

Consider using foreign key constraints to handle related data in other tables.

Reviewing INSERT Operations

Basic Syntax:

```
INSERT INTO [table_name] (column1, column2)
    VALUES (value1, value2);
```

Inserting Multiple Rows:

```
INSERT INTO [table_name] (column1, column2) VALUES
    (value1, value2),
    (value3, value4);
```

Handling Constraints:

NOT NULL, UNIQUE, DEFAULT, PRIMARY KEY

Automatically generated values using **AUTO_INCREMENT**

Combining Commands for Data Integrity

Add a New Student:

```
INSERT INTO students (student_id, name, email)
    VALUES (2, 'Jane Smith', 'jane.smith@example.com');
```

Correct Their Email Address:

```
UPDATE students SET email = 'jane.smith@school.edu'
    WHERE student_id = 2
```

Remove Outdated Student Records:

```
DELETE FROM students WHERE student_id = 2
```