

TECHNOLOGY



Coding Bootcamp

TECHNOLOGY



MySQL

Working with Tables and SQL Commands



Learning Objectives

By the end of this lesson, you will be able to:

- 👁️ Execute SQL commands to perform various data operations within the database
- 👁️ Demonstrate the use of DELETE and UPDATE statements to manage and modify data records accurately
- 👁️ Apply SQL keys to enforce uniqueness and establish relationships between tables, ensuring data integrity
- 👁️ Explain different types of JOIN statements with examples to combine data from multiple tables and retrieve comprehensive results



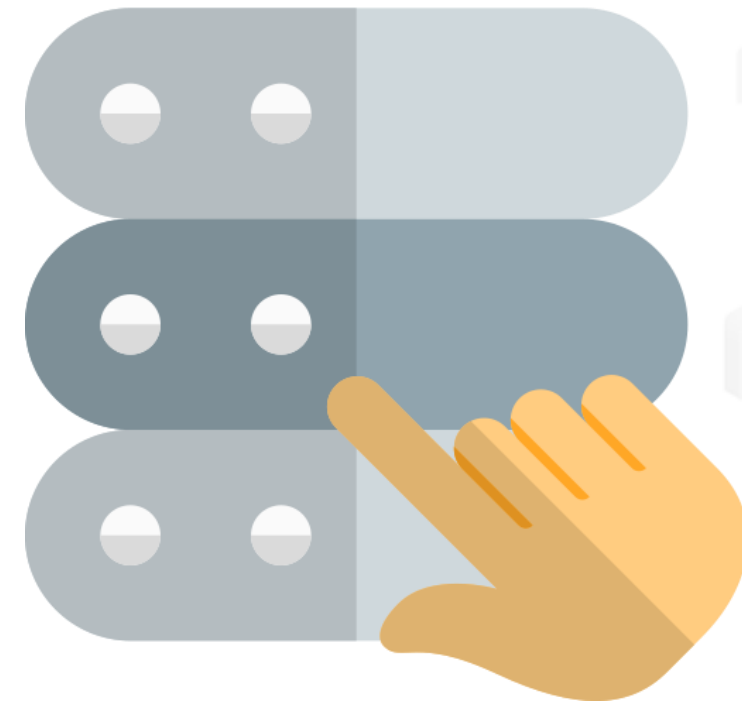
Essential SQL Commands

MySQL Select Statement

It selects data from a database. It allows users to specify the columns and the table from which to retrieve them.

Syntax:

```
SELECT column1, column2, ...  
FROM name_of_table;
```



The returned data is stored in a result table.

MySQL Where Clause

It helps retrieve only the records that meet specific criteria, making data retrieval more precise and efficient.

Syntax:

```
SELECT field1, field2,...fieldN name_of_table1, name_of_table2...  
[WHERE condition1 [AND [OR]] condition2.....
```

It allows for specifying selection criteria to select the required records.



MySQL AND Operator

Below is the syntax for the MySQL AND operator:

Syntax:

```
A AND B
```

	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

The AND and OR operators combine multiple conditions in a WHERE clause. These operators allow complex and specific queries by defining various criteria that must be met for a record to be included in the result set.

MySQL OR Operator

Below is the syntax for the MySQL OR operator:

Syntax:

```
A OR B
```

	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

MySQL ORDER BY Keyword

It sorts the outcome in an ascending or descending order.

Syntax:

```
SELECT column1, column2, ...  
FROM name_of_table  
ORDER BY column1, column2, ... ASC | DESC;
```

Using the ORDER BY operator effectively allows users to control the presentation of the query results, making it easier to analyze and interpret the data.



MySQL GROUP BY Keyword

It collects data from multiple records and groups the output.

This is mainly used in SELECT statements along with SUM, COUNT, and MAX.

Syntax:

```
SELECT expression_a, expression_b, ... expression_n,  
       aggregate_function (expression)  
  
FROM tables  
[WHERE conditions]  
  
GROUP BY expression_a, expression_b, ... expression_n;
```

Using the GROUP BY operator effectively allows for deriving meaningful insights from data by organizing and summarizing it based on specific criteria.



MySQL HAVING Clause

The HAVING clause in MySQL filters records returned by a GROUP BY clause.

Syntax:

```
SELECT expression_a, expression_b, ... expression_n,  
       aggregate_function (expression)  
  
FROM tables  
[WHERE conditions]  
  
GROUP BY expression_a, expression_b, ... expression_n  
HAVING condition;
```

It operates similarly to the WHERE clause, but while WHERE filters rows before the aggregation, HAVING filters rows after the aggregation has been performed.



Implementing Select Statement with Various Clauses



Duration: 10 Min.

Problem Statement:

You have been asked to implement the Select Statement with various clauses.

Outcome:

By completing the task, you will gain the ability to confidently access and explore databases, effectively query and manipulate data, and employ advanced techniques within the SELECT statement to retrieve precise information based on specific criteria.

Note: Refer to the demo document for detailed steps:
[02_Implementing_Select_Statement_with_Various_Clauses](#)

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed are:

1. Access and explore the database
2. Query and manipulate data
3. Query with advanced techniques



MySQL Delete and Update Records

DELETE Keyword

Using the DELETE keyword effectively helps manage and maintain clean and relevant data in a database by removing unnecessary or obsolete records.

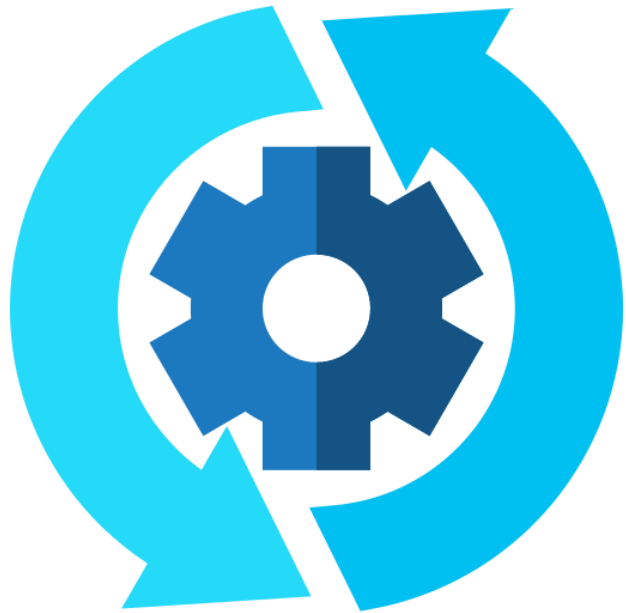
Syntax:

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



UPDATE Keyword

Using the UPDATE keyword effectively helps manage and maintain accurate and current data in a database by allowing precise modifications to existing records.



Syntax:

```
UPDATE [LOW_PRIORITY] [IGNORE] table_name
SET
  column_name1 = expr1,
  column_name2 = expr2,
  ...
[WHERE condition];
```

Modifying and Deleting: SQL Table

UPDATE

The UPDATE statement allows you to modify existing records in a table. Here's how it works:



Syntax:

```
UPDATE name_of_table  
SET column1 = value1, column2 = value2, ...  
WHERE condition;
```



DELETE

The DELETE statement removes existing records based on a specified condition from a table. Here's a detailed look:



Syntax:

```
DELETE FROM name_of_table WHERE condition;
```

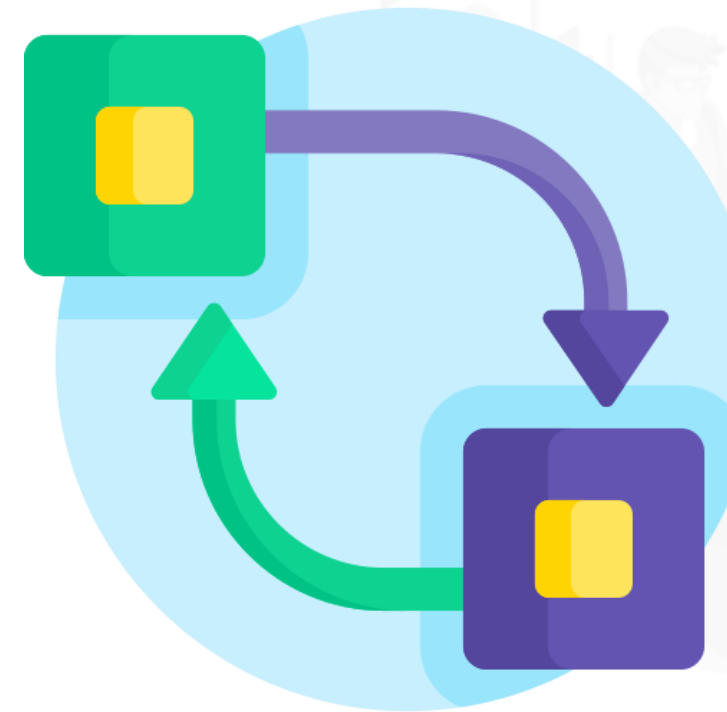


ALTER

The ALTER statement changes the structure of an existing table by adding, deleting, or modifying columns. Here's its syntax and usage:

Syntax:

```
ALTER TABLE name_of_table ADD column_name datatype;
```



DROP TABLE

The DROP TABLE statement permanently deletes a table and all its data from the database. Here's the syntax for this command:



Syntax:

```
DROP TABLE name_of_table
```



Implementing Insert, Update and Delete operations



Duration: 10 Min.

Problem Statement:

You have been asked to implement the Insert, update, and delete operations on a MySQL table.

Outcome:

By completing the tasks, you'll possess the skills to effectively manipulate data within the database by implementing insert, update, and delete operations, and you'll be adept at crafting precise queries using the SELECT statement with various clauses to extract the desired information.

Note: Refer to the demo document for detailed steps:
01_Implementing_Insert_Update_and_Delete_operation

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed are:

1. Insert, update, and delete records from the table



Modifying and Deleting: SQL Database

Modifying an SQL Database

The ALTER statement changes the features of an existing database. This can include modifying the character set, collation, encryption, and read-only status of the database.

Syntax:

```
ALTER {DATABASE | SCHEMA} database_name, alter_option ...

alter_option: {
  [DEFAULT] CHARACTER SET [=] charset_name
  | [DEFAULT] COLLATE [=] collation_name
  | [DEFAULT] ENCRYPTION [=] {'Y' | 'N'}
  | READ ONLY [=] {DEFAULT | 0 | 1}
}
```



Deleting a Database

The DROP DATABASE statement allows you to remove an entire database, including all its tables, data, and associated structures.



Syntax:

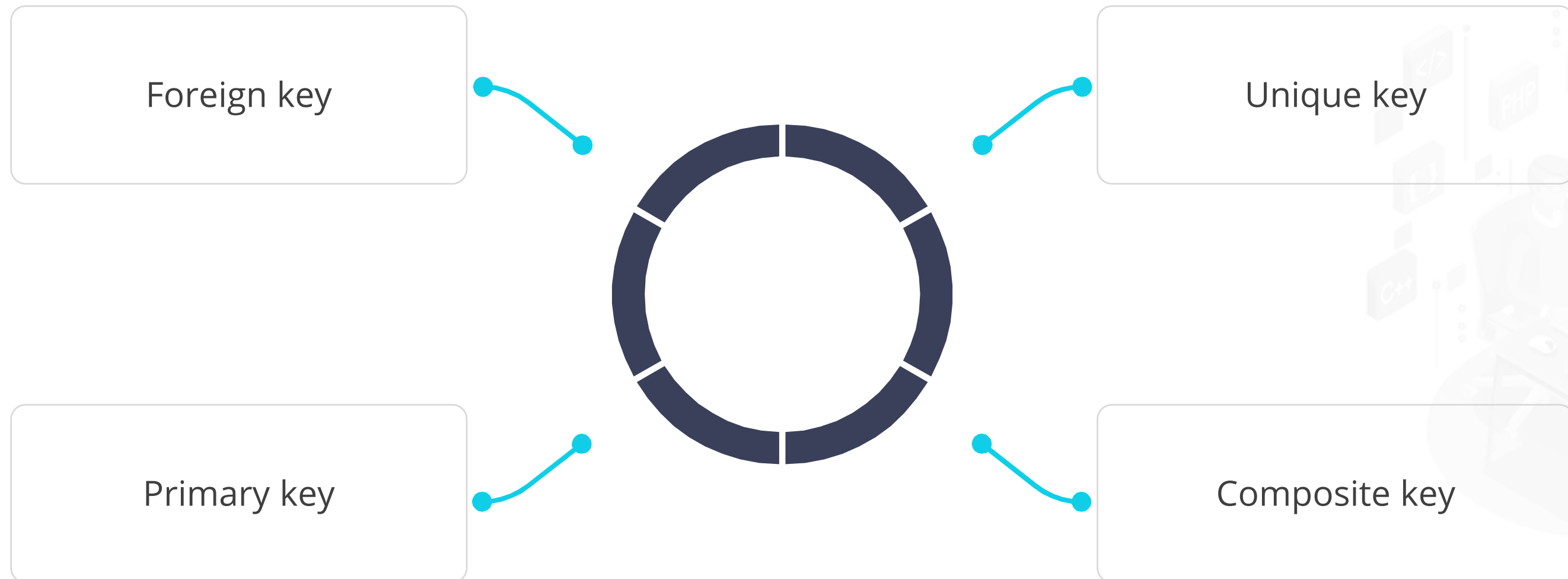
```
DROP DATABASE database_name;
```



MySQL Keys

MySQL Keys

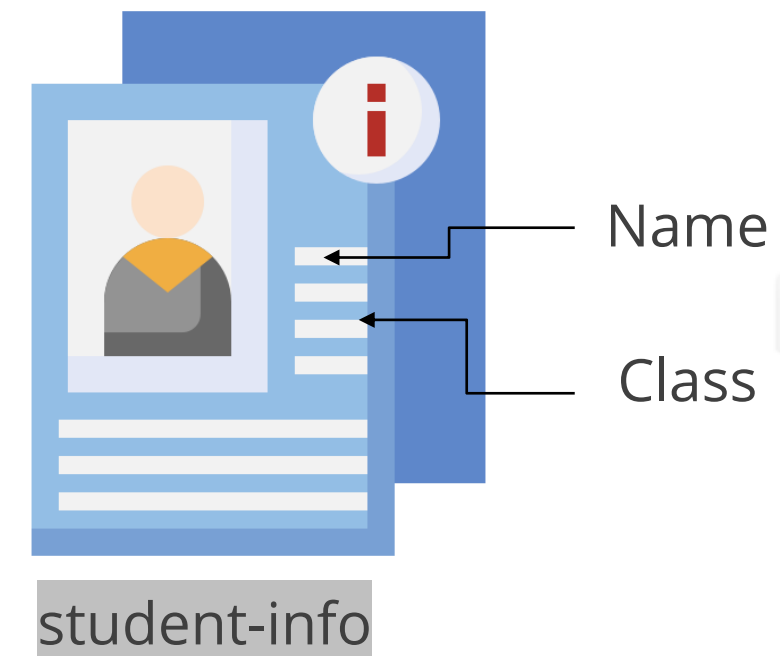
MySQL offers four types of keys essential for maintaining data integrity and relationships within a database. They are:



Unique Key

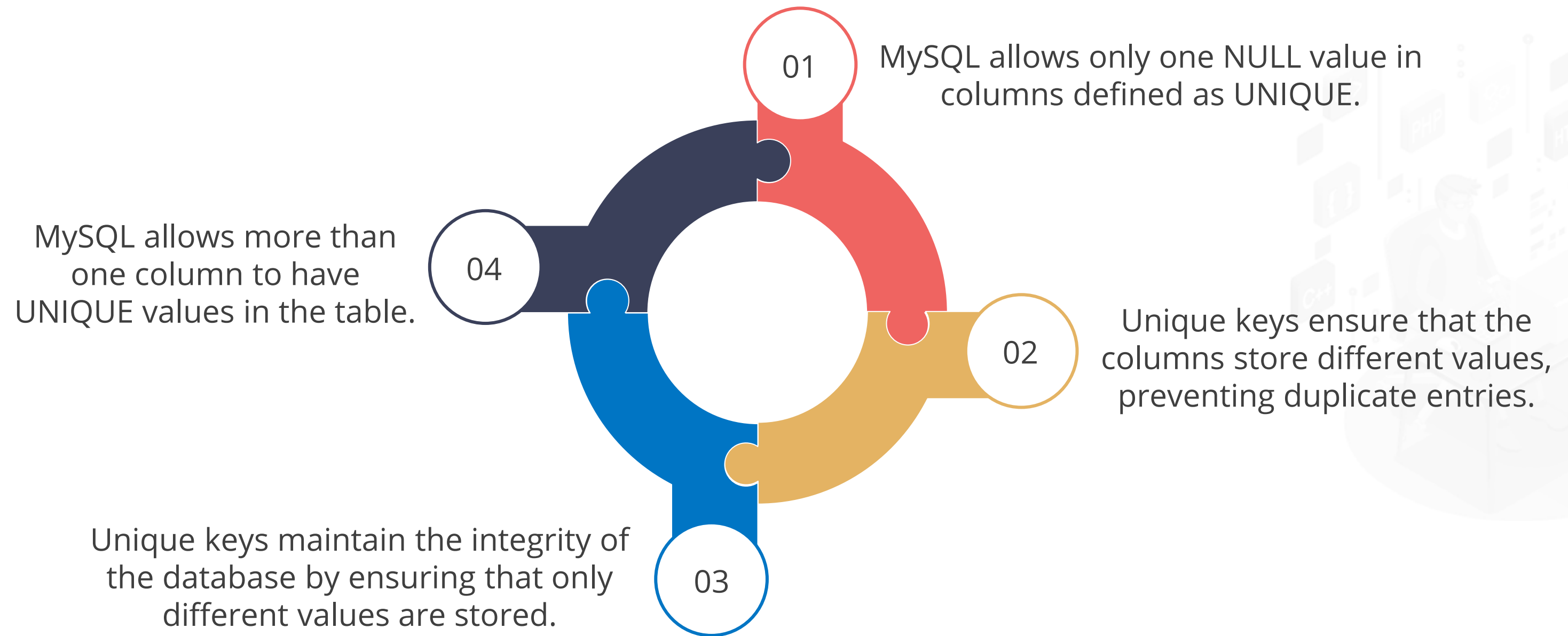
A unique key constraint ensures that no two rows in the table have the same value in the specified column or columns. This is important for maintaining data integrity.

For example, in a student information system, the **student_id** can be set as a unique key to ensure that no two students have the same ID.



Unique Key

The following explains the concept of unique keys in MySQL and their significance in maintaining data integrity:



Unique Key

The syntax to create a unique key in the table is:

```
CREATE TABLE table_name(  
    col1 data_type,  
    col2 data_type UNIQUE,  
    ...  
);
```

The syntax to insert multiple unique keys into a table is:

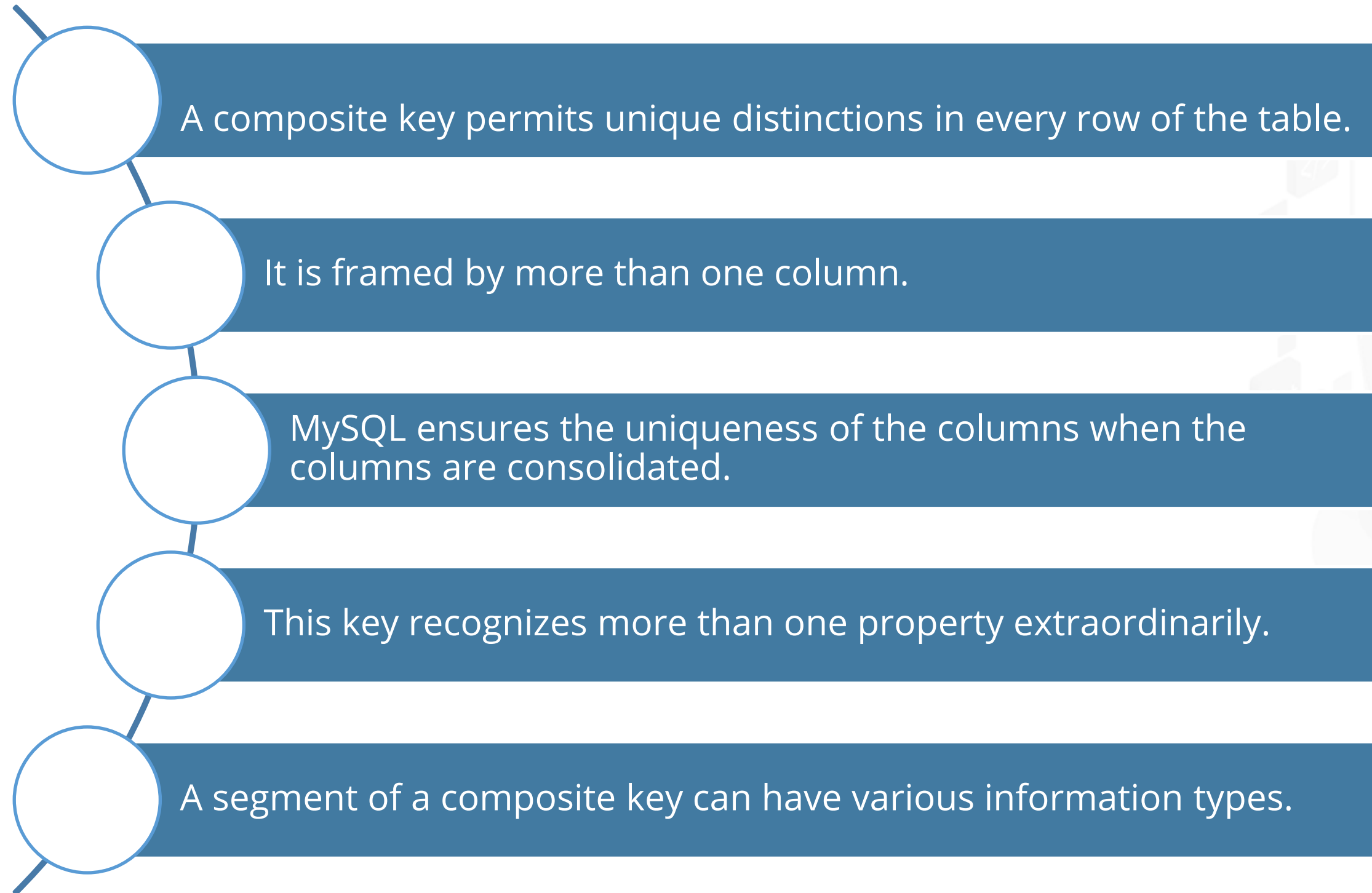
```
CREATE TABLE table_name(  
    col1 column_definition,  
    col2 column_definition,  
    ...  
    [CONSTRAINT constraint_name]  
    UNIQUE(col_name(s))  
);
```

SQL provides a name for that column if a unique constraint is not specified.



Composite Keys

The following explains composite keys in MySQL, which are crucial for ensuring the uniqueness of rows based on multiple columns:



Composite Keys

The composite key can be added in two different ways, using:

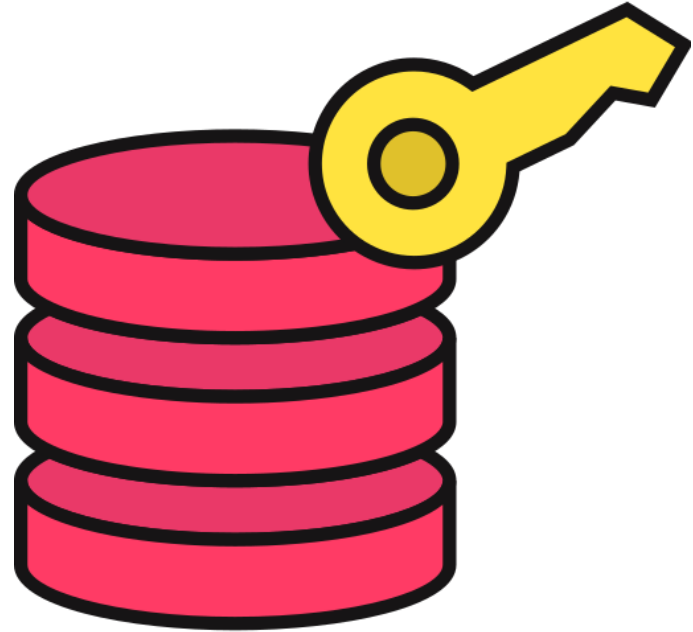


Primary Keys

It recognizes each record in a table uniquely.

It contains unique values and not NULL values.

Syntax:



```
CREATE TABLE Persons (  
  ID int NOT NULL,  
  LastName varchar(255) NOT NULL,  
  FirstName varchar(255),  
  Age int,  
  CONSTRAINT PK_Person PRIMARY KEY  
  (ID,LastName)  
);
```

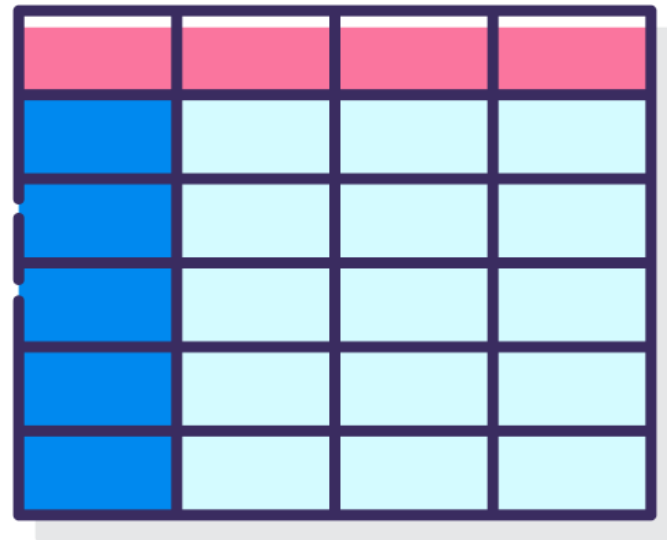


Foreign Keys

It prevents actions that would impact connections between tables.

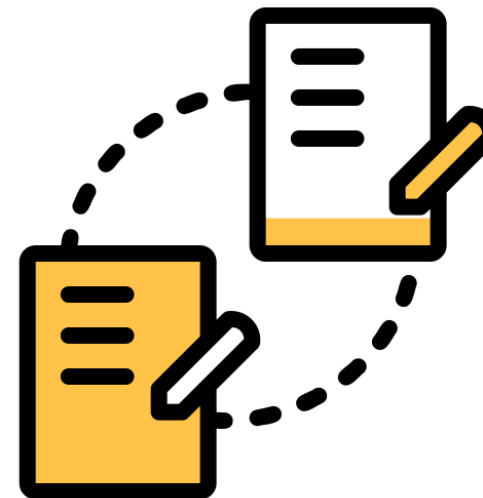
Foreign key

Child table



Parent key

Referenced or
parent table



Foreign Keys

The syntax to create a Foreign Key on the **ProductId** column when the **Orders** table is created is shown below:

```
CREATE TABLE Orders (  
  OrderID int NOT NULL,  
  OrderNumber int NOT NULL,  
  ProductID int,  
    PRIMARY KEY (OrderID),  
    FOREIGN KEY (ProductId) REFERENCES  
  Persons(ProductId)  
);
```



MySQL Distinct Keyword

The SELECT DISTINCT statement returns different values.



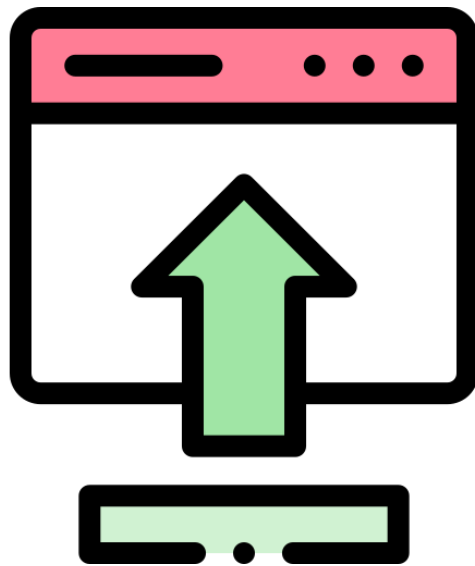
Syntax:

```
SELECT DISTINCT column1, column2, ...  
FROM name_of_table;
```



MySQL Insert Into Keyword

The MySQL Insert statement inserts single or multiple records into a table.



Syntax:

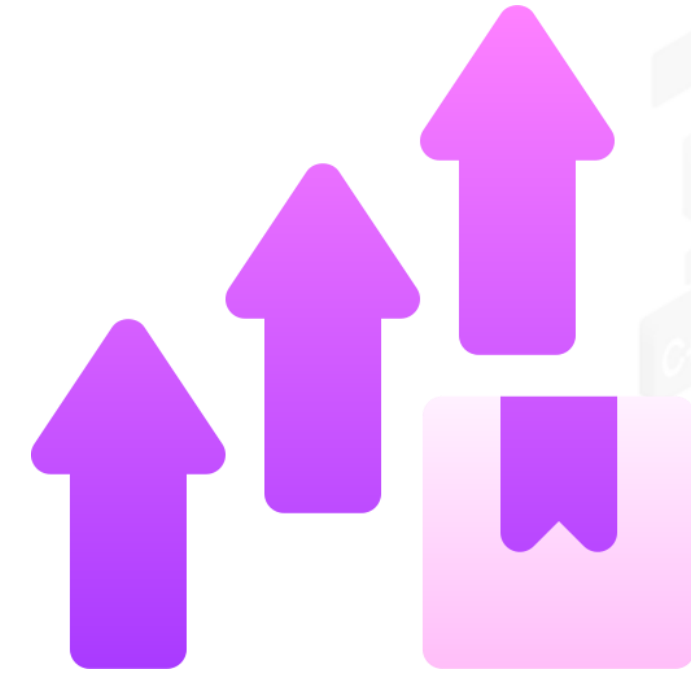
```
INSERT INTO table  
(column1, column2, ... )  
VALUES  
(expression1, expression2, ... ),  
(expression1, expression2, ... ),  
...;
```

MySQL Get Last Inserted ID

The LAST_INSERT_ID() function returns the AUTO_INCREMENT ID of the last row added or updated in a table.

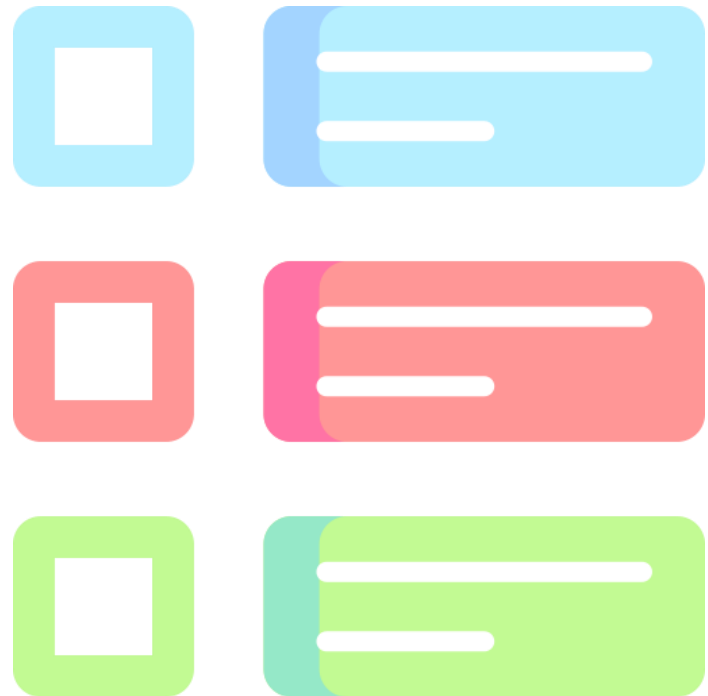
Syntax:

```
LAST_INSERT_ID(expression)
```



MySQL Insert Multiple Records Command

It inserts multiple rows into a table.



Syntax:

```
LAST_INSERT_ID(exINSERT INTO name_of_table  
(column_list)  
VALUES  
(value_list_1),  
(value_list_2),  
...  
(value_list_n);  
pression)
```


Creating and Managing Related Table



Duration: 10 Min.

Problem Statement:

You have been asked to create and manage related tables in MySQL.

Outcome:

By completing the task of creating and managing related tables in MySQL, you establish a robust database structure that enables efficient organization and retrieval of interconnected data, thus facilitating smoother data management and analysis processes.

Note: Refer to the demo document for detailed steps:
[03_Creating_and_Managing_Related_Tables](#)

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed are:

1. Creating and Managing Related Table

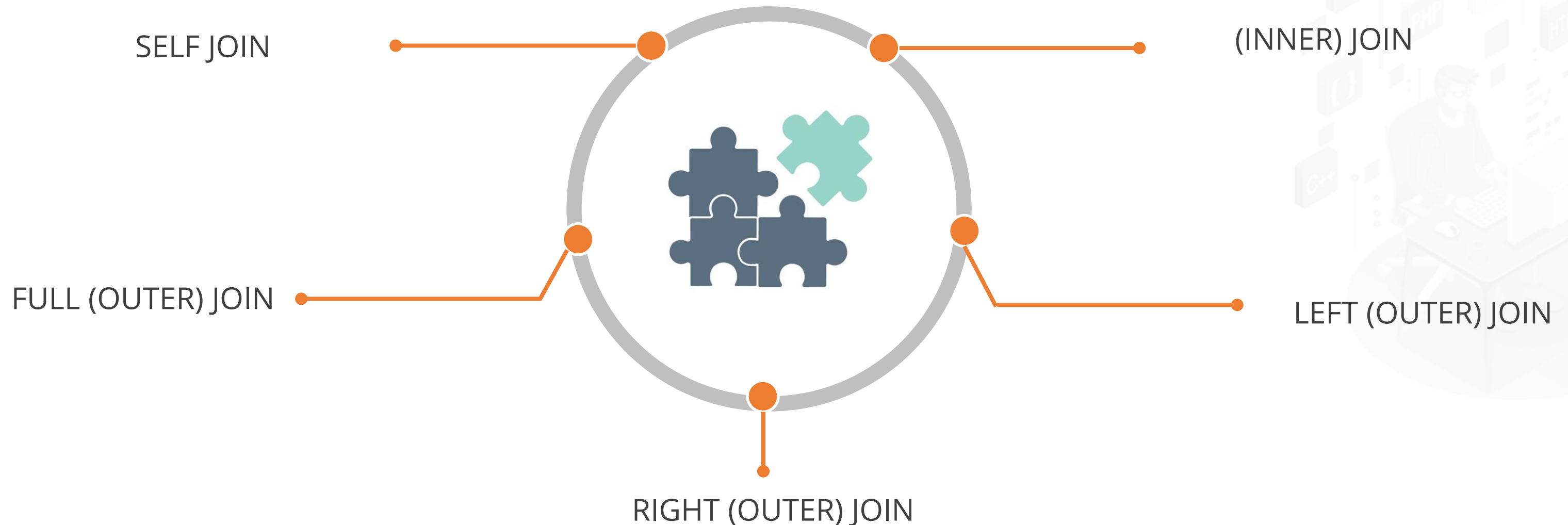


MySQL Joins

MySQL Joins

A JOIN statement merges the rows from two or more tables.

The different types of JOINS are:



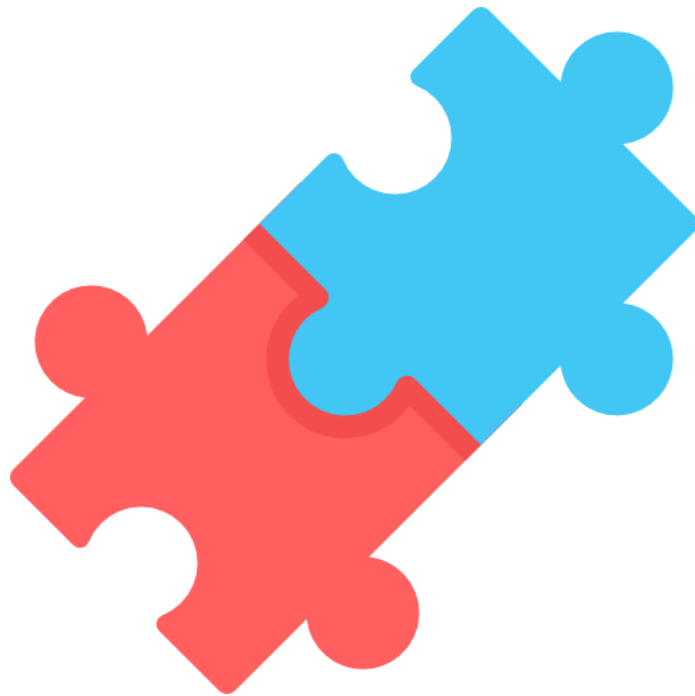
Inner Join

It returns records that have the same values in both tables.

The keyword takes the records with matching values in both tables.

Syntax:

```
SELECT column_name_(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```



Left Join

The LEFT JOIN statement returns all records from the left table and the matched records from the right table. The result is NULL on the right side if no match is found.

Syntax:



```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```


Right Join

The RIGHT JOIN statement returns all records from the right table and the matched records from the left table. If no match is found, the result is NULL on the left side.

Syntax:

```
SELECT column_name_(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```



Full Join

The FULL JOIN statement returns all records when there is a match in the left or right table records. Non-matched rows are filled with NULL values.

Syntax:

```
SELECT column_name_(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name = table2.column_name
WHERE condition;
```



Self Join

It is a regular join where the table is joined by itself. A SELF JOIN statement is helpful when comparing rows within the same table, such as finding pairs of employees working in the same department.

Syntax:

```
SELECT column_name(s)
FROM table1 T1, table1 T2
WHERE condition;
```



Implementing Joins on Tables



Duration: 10 Min.

Problem Statement:

You have been asked to implement various JOIN operations on MySQL tables.

Outcome:

By completing the tasks, you'll master diverse MySQL join operations, enabling sophisticated data merging for advanced analysis and insights.

Note: Refer to the demo document for detailed steps:
04_Implementing_Joins_on_Tables

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed are:

1. Implement different types of table joins



Key Takeaways

- The UPDATE statement modifies the existing records in the table. It allows users to change existing data based on specified conditions.
- ALTER changes the features of a database. This command modifies the structure of an existing table.
- A composite key permits you to distinguish every row of the table. It comprises two or more columns uniquely identifying a row in a table.
- The MySQL INSERT statement inserts single or multiple records into a table. It adds new rows of data to a table, ensuring data integrity and consistency.
- A JOIN statement merges rows from two or more tables. This statement combines rows based on a related column, enabling complex queries and data analysis.
- The DELETE keyword deletes the current records from a table.



TECHNOLOGY

Thank You