- **Tuesday December 5th (2 h): Eugene.**
- **Advanced techniques: JOINS. NO SQL Databases.**

SQL syntax rules:

1. Clauses: SQL clauses perform specific tasks, which are used as essential components of queries and commands. Examples include 'FROM', 'WHERE', 'AND', 'OR', 'NOT', 'ORDER BY', etc. These must be in a correct syntax and sequence to avoid errors.

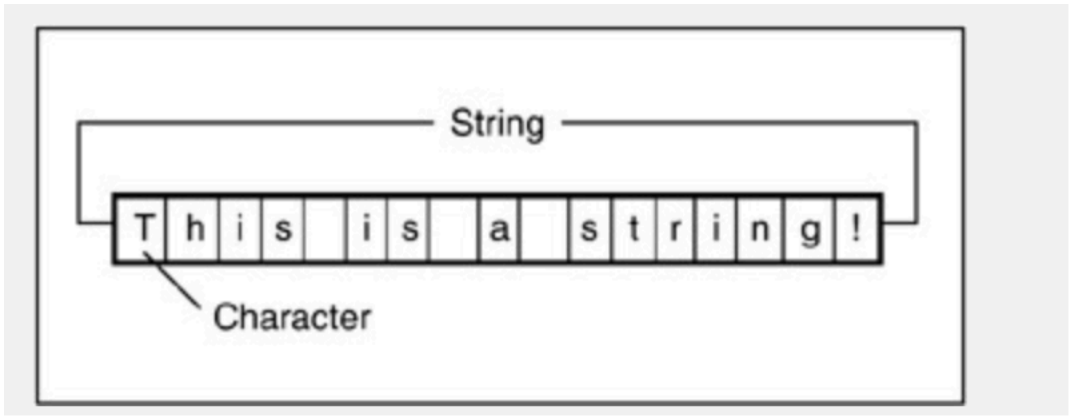
2. Statements: Each SQL command is called a statement. They start with any SQL command keyword like SELECT, INSERT, UPDATE, DELETE, ALTER, DROP etc. and ends with a semicolon (;).

3. Keywords: SQL keywords are not case sensitive, such as select, from, where, etc. However, best practices often dictate using uppercase for better readability.

4. Identifiers: Some database systems require identifiers (such as table names, column names, and aliases) to be in uppercase, and others are case sensitive if quotes are used ('Name' vs 'NAME').

5. Single/Double Quotes: String values must be enclosed within single quotes ('Hello'), and identifiers/aliases can be enclosed within double quotes (“FirstName").

A string value in SQL refers to alphanumeric characters which can consist of letters, numbers, special characters, or spaces. String values are primarily used to store text-based data such as names, addresses, descriptions, etc., in a SQL database.



6. Comments: Single line comments start with -- and comment section starts with /* and ends with */. //

7**. Trimming Spaces:** SQL is not sensitive to white spaces or line breaks. However, proper formatting with whitespace helps improve readability of the query.

9. Order of Execution: Different parts of a query are executed in a specific order: FROM, WHERE, GROUP BY, HAVING, SELECT, ORDER BY. Understanding this order is important to get accurate results.

10. Data Types: Each column in a database table is required to have a name and a data type such as CHAR, VARCHAR, INT, DATE, etc. SQL requires you to define the type of data to be stored in each field.

These are different data types used in SQL:

1. **CHAR:** This is a data type that can store characters. The user needs to define a fixed length (up to 8000 characters) when creating a column of this type. For example, CHAR(5) could store a 5-character word.

2. VARCHAR: This is similar to CHAR. It also stores characters but it's variable-length up to the limit specified when created (up to 8000 in SQL Server and 65535 in MySQL). It uses only as much space as the actual value needs. For example, VARCHAR(100) can store a string of any length up to 100 characters.

3. INT: This stands for integer. It can store a whole number (a number without a decimal place > (,)). The range of values an INT can store depends on the specific SQL database but generally it can store a number between about -2 billion and 2 billion.

4. DATE: This data type is used to store dates. The format in which the dates are stored depends on the specific SQL database. For example, in MySQL, the DATE data type format is 'YYYY-MM-DD'. It only includes the date, not the time.

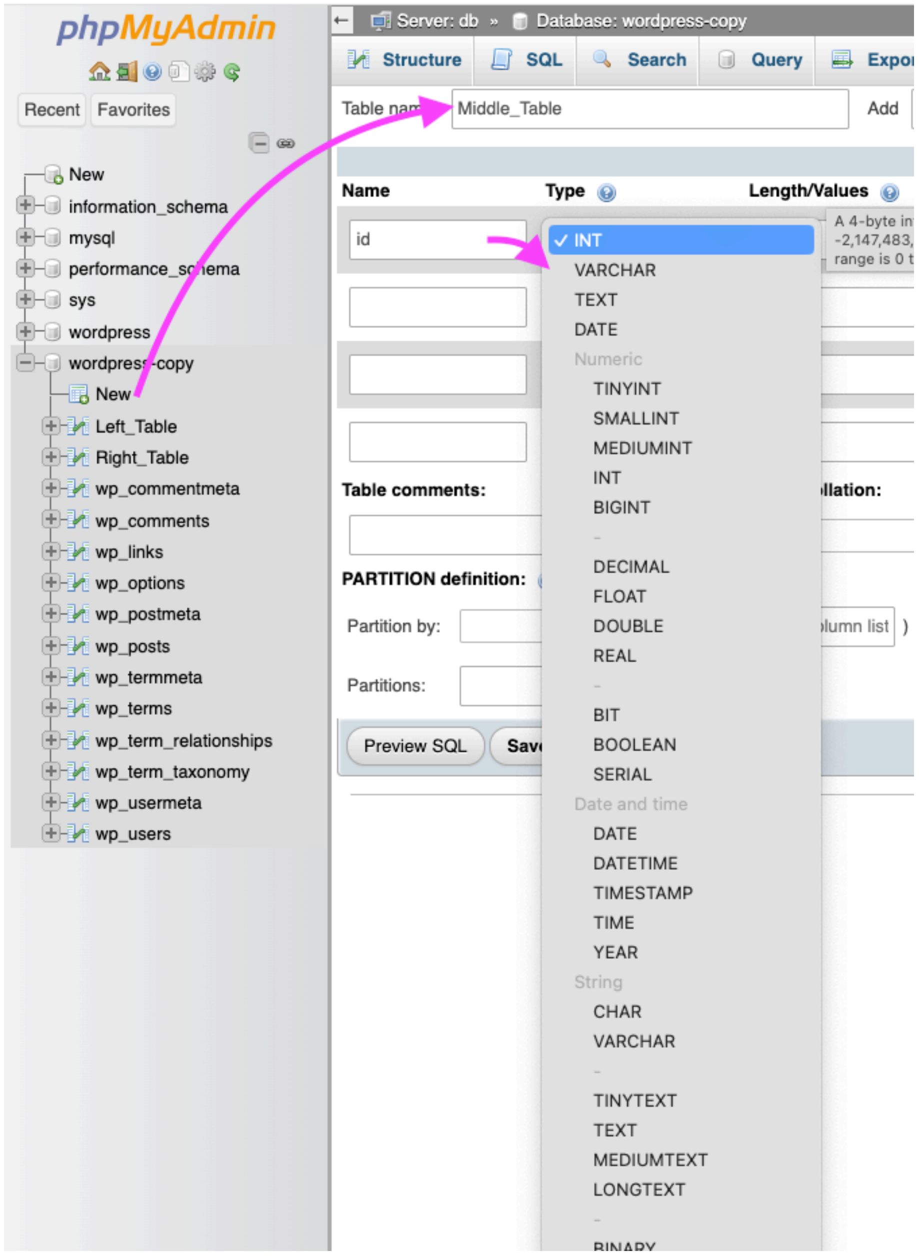


Table nam: Middle_Table

Name	Type	Length/Values
id	INT	

A 4-byte in
-2,147,483,
range is 0 t

Table comments:

llation:

PARTITION definition:

Partition by:

lumn list)

Partitions:

11. Use of Semicolons: In SQL, semicolon is the standard way to separate each SQL statement. While some SQL databases, like MySQL, allow you to omit the semicolon at the end of the statement, others require it.

12. Database-Specific Syntax: Keep in mind that each SQL database may have slightly different syntax rules. For instance, MS SQL Server uses T-SQL, Oracle Database uses PL/SQL, MySQL uses MySQL procedural language, etc.

Keep in mind that the specifics and limitations of these data types can vary slightly depending on the SQL database management system (DBMS) being used.
It's always essential to refer to the documentation specific to your DBMS for any clarifications.

Each database has slightly different syntax and functionalities.
It is important to understand the specifics of the SQL variant being used.

Wildcards
% _ [ ] *

A wildcard character in SQL is used to substitute any other character(s) in a string. It's used with the SQL LIKE operator, and the commonly used wildcards are `%`, `_`, and `[]`.

1. Percent `%`: The percent sign represents zero, one, or multiple characters. For instance, the expression `'a%'`would find any values that start with `a`. %Evgeny%

2. Underscore `_`: The underscore represents a single character. For example, the expression `'a_'` would find values such as `an`, `am`, `at`, where there is exactly one character following `a`.

3. Brackets `[ ]`: In Microsoft's SQL Server, you can also use brackets to represent any single character within the brackets. For example `'a[bc]'` would find values such as `ab` or `ac`.

4. The asterisk (*) is a commonly used wildcard character. In many programming languages, command-line interfaces, and search engines, the asterisk can replace zero or more characters in a string. This is useful when you want to match or select a group of items having a common pattern. For example, in a file directory, typing "*.txt" would select all text files, whereas "doc*" may select all files beginning with "doc".

Example usage with the LIKE operator:

```
SELECT column_name
FROM table_name
WHERE column_name LIKE 'a%';
```

This query would return all values in `column_name` that start with `a`.

Remember that the interpretation of the wildcard can slightly differ based on the SQL variant you are using. Always refer to the SQL reference specific to the dialect you are using for precise information

What do you need to know as a tester:

The login/password/email field, which is usually a username or an email address, should not accept wildcards.

WHY?

Accepting wildcard characters like '%' or '_' could lead to security vulnerabilities.
For instance, if your system had a flaw and was doing a simple SQL match for the login field, a user could potentially enter a '%' in the login field to match any number of arbitrary characters, leading to unauthorized access. **This is a type of attack known as a SQL injection.**

In this case ANY character can be accepted as login, password, email information.

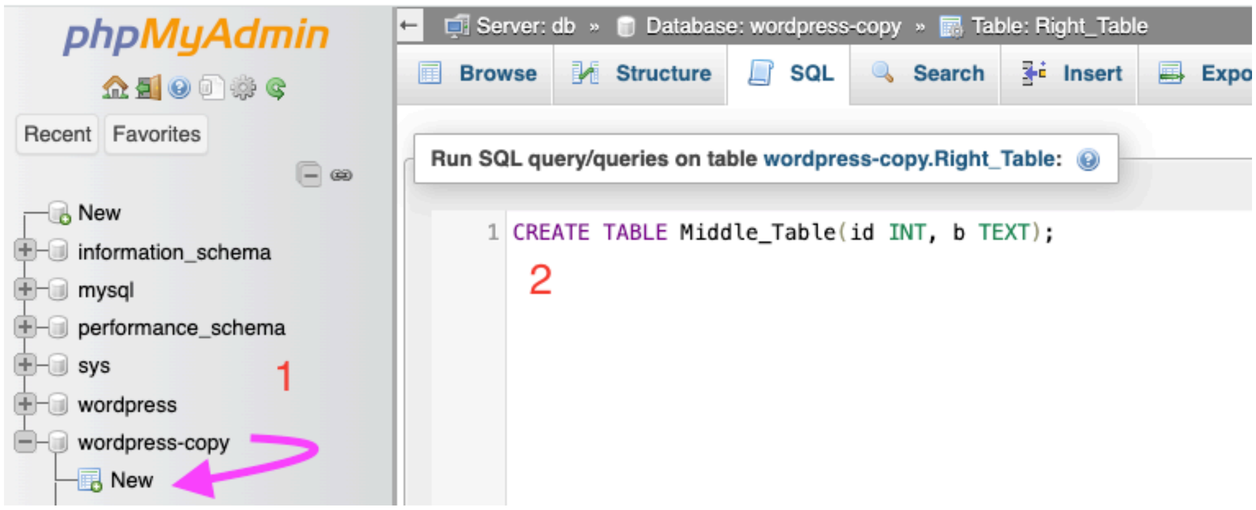
Apart from security concerns, allowing wildcards in login fields could complicate user experience, cause confusion, or lead to duplicate username/email issues.

You need to test all these fields this way.
From a QA tester's perspective, it's crucial to check that wildcards are not accepted in login fields, both to prevent security issues and to maintain the quality and integrity of user data.

Example with Middle table

Can be created manually > Example

OR with query



Run SQL query/queries on table wordpress-copy.Right_Table:

```
CREATE TABLE Middle_Table(id INT, b TEXT);
```

Populate it with data:

INSERT INTO Middle_Table(id,value) VALUES(1,"middle1");

DROP (example with middle table)

DROP TABLE name_of_the_table;

DROP TABLE Middle_Table;

DROP TABLE IF EXISTS name_of_the_table;

```
DROP TABLE IF EXISTS Middle_Table;
```

ASK THREE TIMES BEFORE DO THIS!

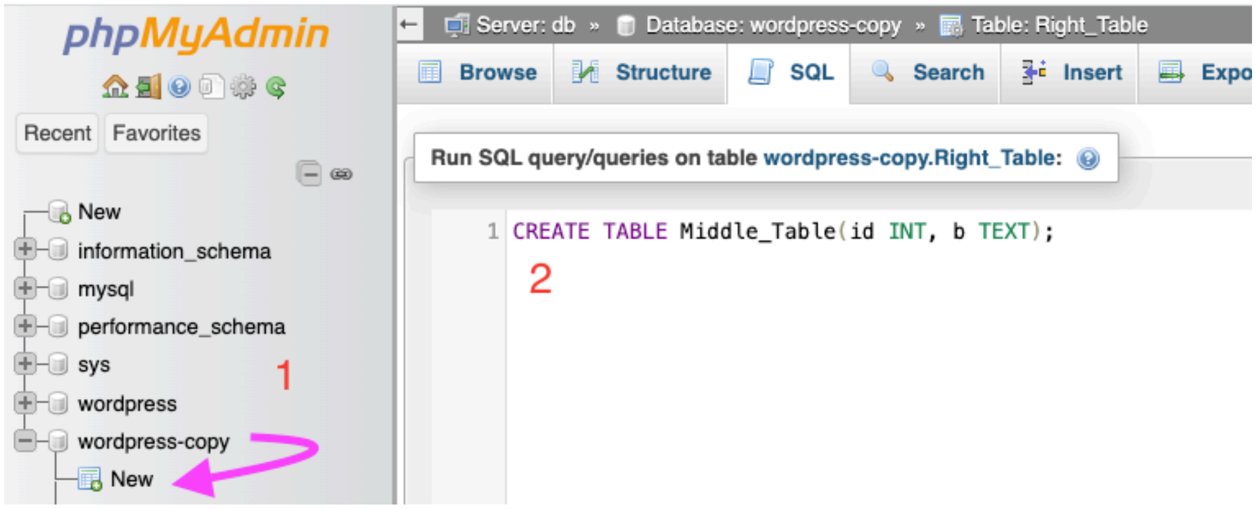
DROP: The DROP command is used to delete an entire database, table, indexes, or view. This is a destructive command – once you drop a table or database, all the information in it is deleted and there's no way to recover it (unless you have a backup). For example:
   ```sql
   DROP TABLE table_name;   ```

   This command removes the complete data along with the table structure or the entire database from the system. DROP operations cannot be rolled back.
   ```

Before the join let create tables:

Wordpress-copy table
https://phpmyadmin.tech-start.io/index.php?route=/database/structure&db=wordpress-copy

Manually
1. New > Create > Provide data



Run SQL query/queries on table wordpress-copy.Right_Table:

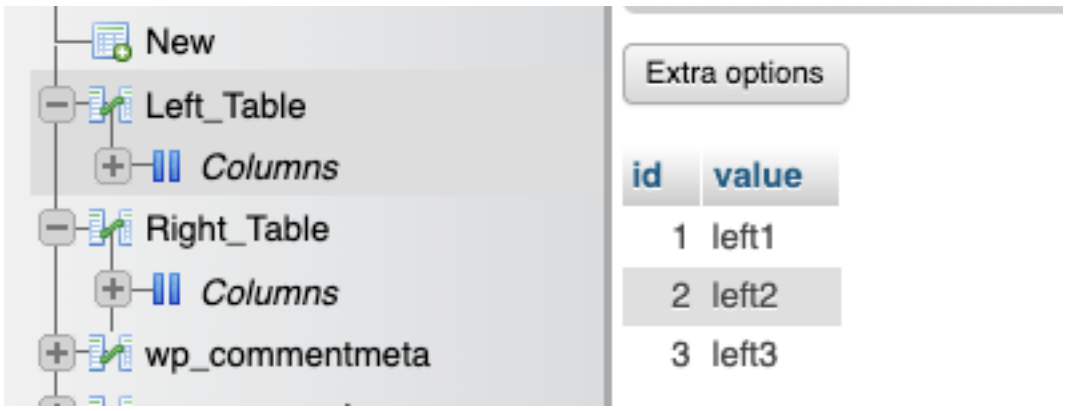
```
CREATE TABLE Middle_Table(id INT, b TEXT);
```

OR > remember logical operator? :)

2. Run CREATE query

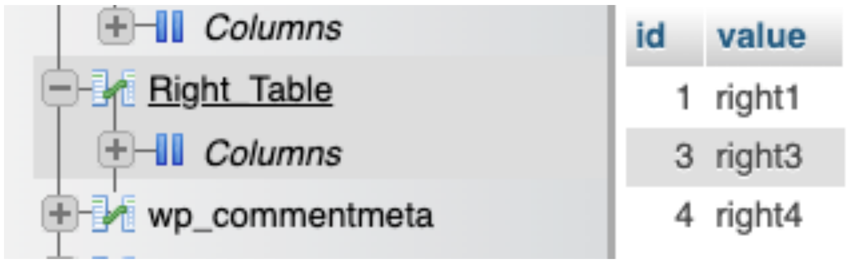
LEFT and RIGHT table

LEFT



id	value
1	left1
2	left2
3	left3

RIGHT
INSERT INTO Right_Table(id, value) VALUES (1, "right1");



id	value
1	right1
3	right3
4	right4

JOIN is a command used in SQL databases to combine rows from two or more tables based on a related column between them. It allows users to query data from multiple tables as if the data resides in a single table.

There are several types of JOIN methods, including:

1. **INNER JOIN:** This returns records that have matching values in both tables.

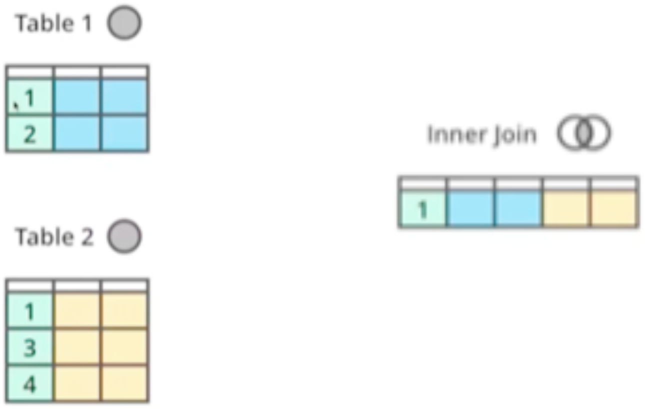
2. **LEFT (OUTER) JOIN:** This returns all records from the left table, and the matched records from the right table. If there is no match, the result is NULL on the right side.

3. **RIGHT (OUTER) JOIN:** This returns all records from the right table, and the matched records from the left table. If there is no match, the result is NULL on the left side.
4. **FULL (OUTER) JOIN:** This returns all records when there is a match in either the left or the right table.

The JOIN operation is a crucial command for combining and extracting useful insights from multiple database tables.

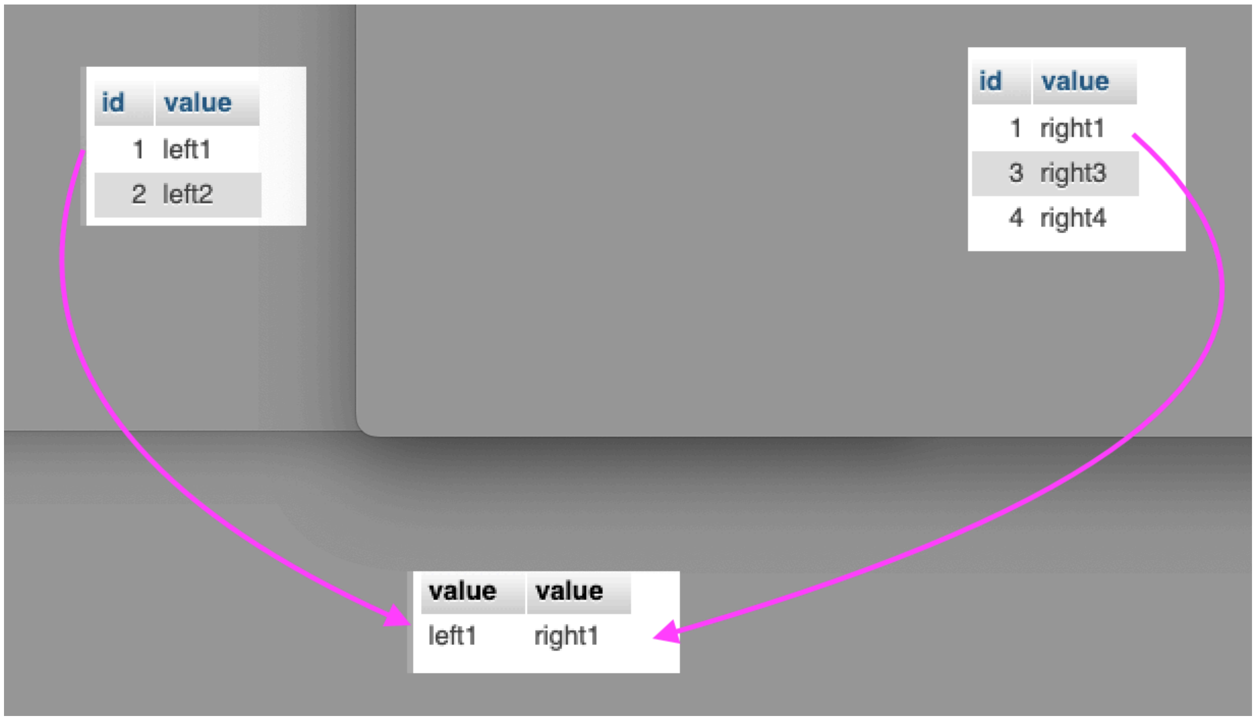
If you understand "JOIN" you will pass any SQL related interview.

1. **INNER JOIN: This returns records that have matching values <u>in both tables.</u>**



SELECT Left_Table.value, Right_Table.value FROM Left_Table JOIN Right_Table ON Left_Table.id=Right_Table.id;

SELECT Left_Table.value, Right_Table.value FROM Left_Table JOIN Right_Table ON Left_Table.id=Right_Table.id;



id	value
1	left1
2	left2

id	value
1	right1
3	right3
4	right4

value	value
left1	right1

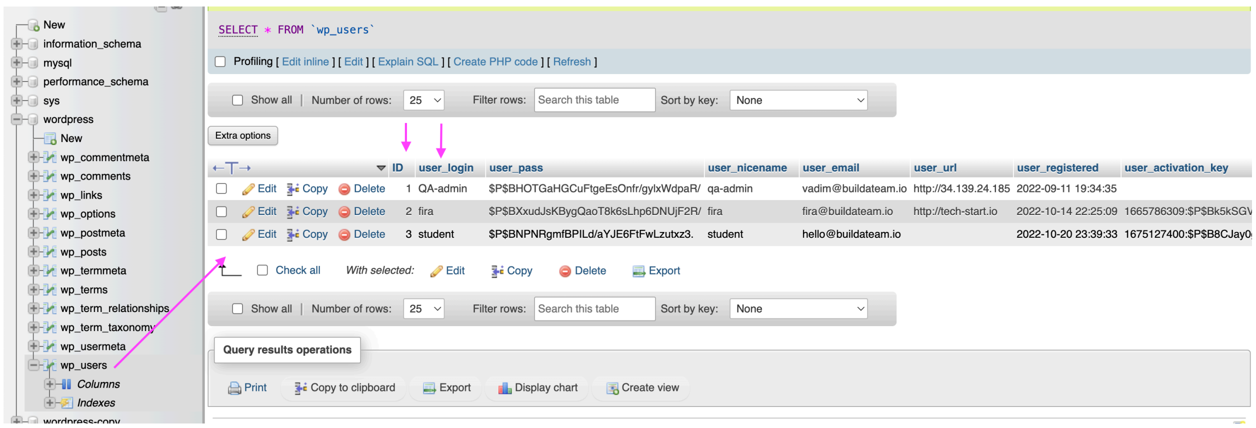
With aliases

```
SELECT L.value, R.value FROM Left_Table AS L JOIN Right_Table AS R ON
L.id=R.id;
```

SELECT wp_users.ID, wp_users.user_login,
wp_comments.comment_author,wp_comments.comment_content
FROM
wp_comments INNER JOIN wp_users ON
wp_comments.user_id=wp_users.ID;

line by line:

SELECT wp_users.ID, wp_users.user_login,



New
information_schema
mysql
performance_schema
sys
wordpress
New
wp_commentmeta
wp_comments
wp_links
wp_options
wp_postmeta
wp_posts
wp_termmeta
wp_terms
wp_term_relationships
wp_term_taxonomy
wp_usermeta
wp_users
Columns
Indexes

```
SELECT * FROM `wp_users`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	ID	user_login	user_pass	user_nicename	user_email	user_url	user_registered	user_activation_key
Edit Copy Delete	1	QA-admin	PBHOTGaHGCuFtgeEsOnfr/gylxWdpaR/	qa-admin	vadim@buildateam.io	http://34.139.24.185	2022-09-11 19:34:35	
Edit Copy Delete	2	fira	PBXxudJsKBygQaoT8k6sLhp6DNUjF2R/	fira	fira@buildateam.io	http://tech-start.io	2022-10-14 22:25:09	1665786309:PBk5kSGV
Edit Copy Delete	3	student	PBNPNRgmfBPILd/aYJE6FtFwLzutxz3.	student	hello@buildateam.io		2022-10-20 23:39:33	1675127400:PB8CJay0

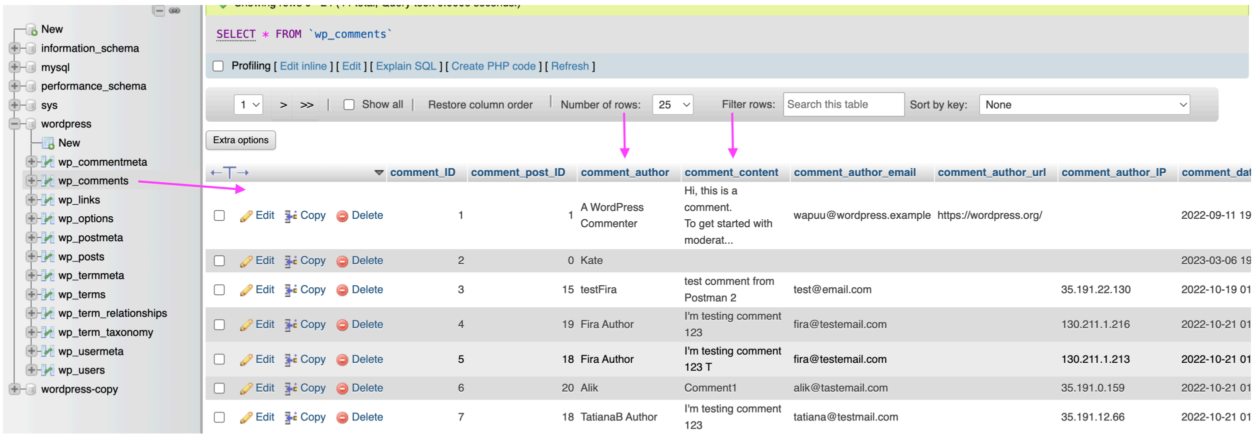
Check all With selected: Edit Copy Delete Export

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Query results operations

Print Copy to clipboard Export Display chart Create view

wp_comments.comment_author,wp_comments.comment_content



New
information_schema
mysql
performance_schema
sys
wordpress
New
wp_commentmeta
wp_comments
wp_links
wp_options
wp_postmeta
wp_posts
wp_termmeta
wp_terms
wp_term_relationships
wp_term_taxonomy
wp_usermeta
wp_users
wordpress-copy

```
SELECT * FROM `wp_comments`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

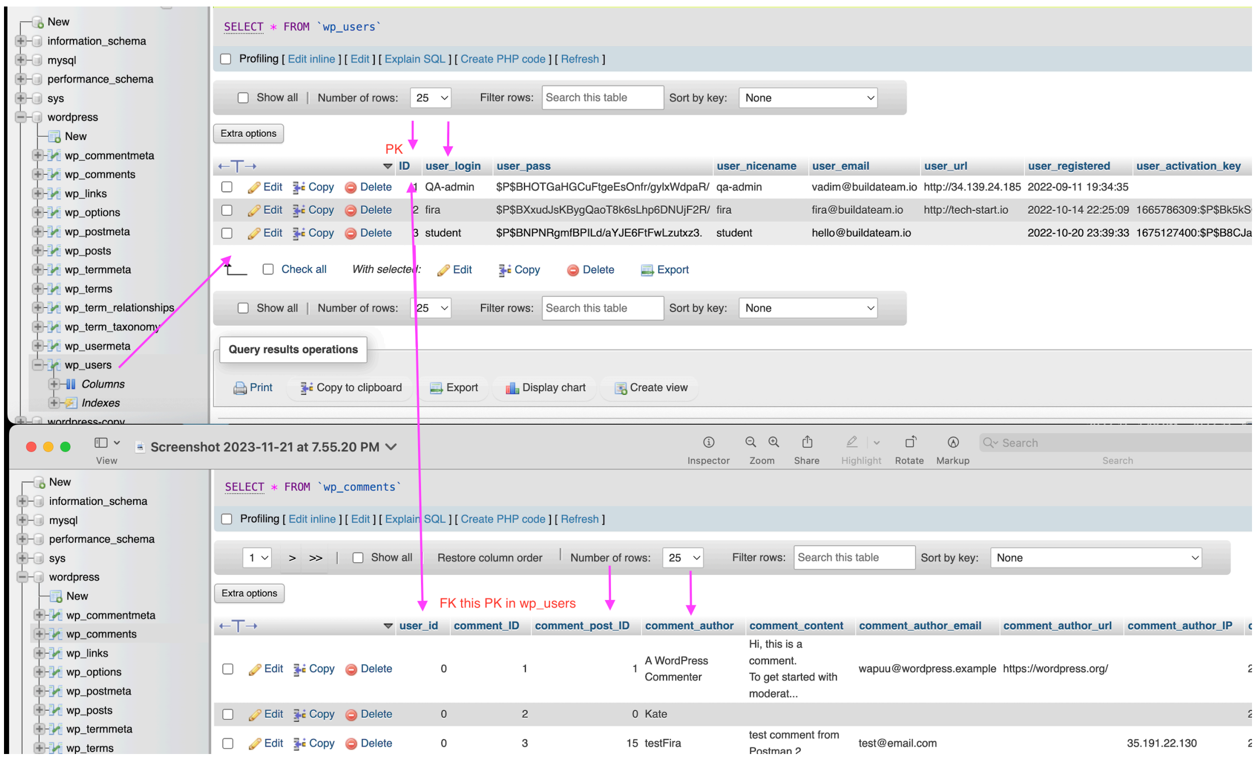
1 > >> | Show all | Restore column order | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	comment_ID	comment_post_ID	comment_author	comment_content	comment_author_email	comment_author_url	comment_author_IP	comment_dat
Edit Copy Delete	1	1	A WordPress Commenter	Hi, this is a comment. To get started with moderat...	wapuu@wordpress.example	https://wordpress.org/		2022-09-11 19
Edit Copy Delete	2	0	Kate					2023-03-06 19
Edit Copy Delete	3	15	testFira	test comment from Postman 2	test@email.com		35.191.22.130	2022-10-19 01
Edit Copy Delete	4	19	Fira Author	I'm testing comment 123	fira@testemail.com		130.211.1.216	2022-10-21 01
Edit Copy Delete	5	18	Fira Author	I'm testing comment 123 T	fira@testemail.com		130.211.1.213	2022-10-21 01
Edit Copy Delete	6	20	Alik	Comment1	alik@tastemail.com		35.191.0.159	2022-10-21 01
Edit Copy Delete	7	18	TatianaB Author	I'm testing comment 123	tatiana@testmail.com		35.191.12.66	2022-10-21 01

FROM
(connecting these two tables)
wp_comments INNER JOIN wp_users

*(connecting them on the related column **id PK and FK in these tables**)*
ON
wp_comments.user_id=wp_users.ID;



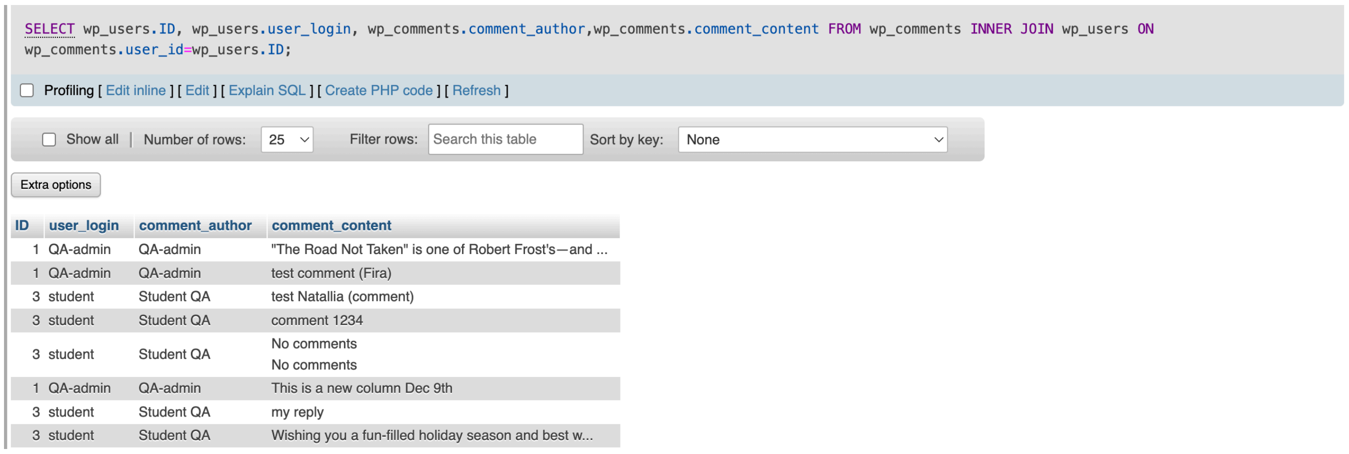
```
SELECT * FROM `wp_users`
```

	ID	user_login	user_pass	user_nicename	user_email	user_url	user_registered	user_activation_key
Edit Copy Delete	1	QA-admin	PBHOTGaHGCuFtgeEsOnfr/gylxWdpaR/	qa-admin	vadim@buildateam.io	http://34.139.24.185	2022-09-11 19:34:35	
Edit Copy Delete	2	fira	PBXxudJsKBygQaoT8k6sLhp6DNUjF2R/	fira	fira@buildateam.io	http://tech-start.io	2022-10-14 22:25:09	1665786309:PBk5kS
Edit Copy Delete	3	student	PBNPNRgmfBPILd/aYJE6FtFwLzutxz3.	student	hello@buildateam.io		2022-10-20 23:39:33	1675127400:PB8CJa

```
SELECT * FROM `wp_comments`
```

	user_id	comment_ID	comment_post_ID	comment_author	comment_content	comment_author_email	comment_author_url	comment_author_IP	c
Edit Copy Delete	0	1	1	A WordPress Commenter	Hi, this is a comment. To get started with moderat...	wapuu@wordpress.example	https://wordpress.org/		2
Edit Copy Delete	0	2	0	Kate					2
Edit Copy Delete	0	3	15	testFira	test comment from Postman 2	test@email.com		35.191.22.130	2

Final result



```
SELECT wp_users.ID, wp_users.user_login, wp_comments.comment_author,wp_comments.comment_content FROM wp_comments INNER JOIN wp_users ON
wp_comments.user_id=wp_users.ID;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

ID	user_login	comment_author	comment_content
1	QA-admin	QA-admin	"The Road Not Taken" is one of Robert Frost's—and ...
1	QA-admin	QA-admin	test comment (Fira)
3	student	Student QA	test Natallia (comment)
3	student	Student QA	comment 1234
3	student	Student QA	No comments No comments
1	QA-admin	QA-admin	This is a new column Dec 9th
3	student	Student QA	my reply
3	student	Student QA	Wishing you a fun-filled holiday season and best w...

With aliases. Aliases are used in SQL queries for several reasons:

1. Simplification: In a SQL query with join operations, we're often dealing with multiple tables that might have complex or long names. By using aliases, we can simplify the query, making it easier to read and write.

2. Avoiding ambiguity: If two or more tables share some column names, the SQL engine will not know to which table the common column names refer when they are used in the where clause or join condition. By using aliases, we can give a unique identifier to each table's column, avoiding ambiguity.

3. Efficiency: By using aliases, we can save time and reduce errors in our queries. Once an alias is defined, we can use it instead of the full table name or column name, making our code neater and shorter.

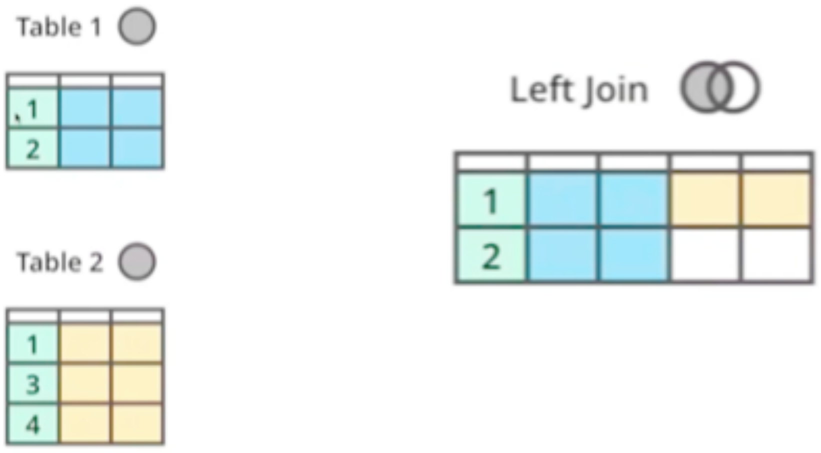
4. Readability: When performing multiple joins or working with subqueries, aliases can be used to help identify which table you are referencing, making the SQL query more readable and maintainable. In overall it looks less scary

```
SELECT U.ID, U.user_login,C.comment_author,C.comment_content FROM
wp_comments AS C INNER JOIN wp_users AS U ON C.user_id=U.ID;
```

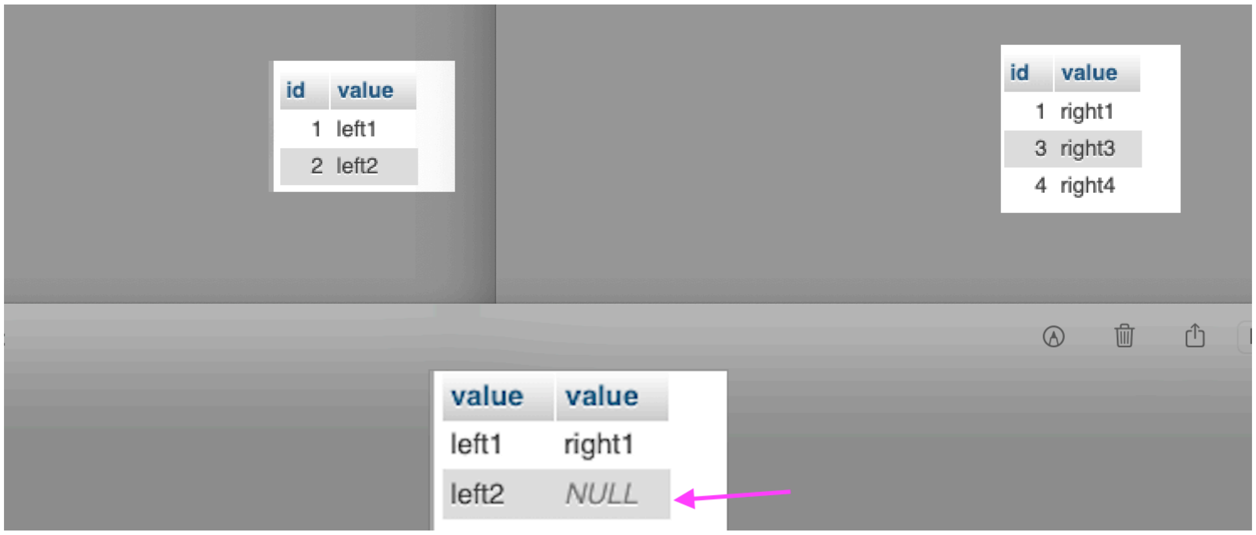
U = wp_users database

C = wp_comments database

LEFT (OUTER) JOIN: This returns **<u>all records from the left table</u>**, and the matched records from the right table. If there is no match, the result is NULL on the right side.



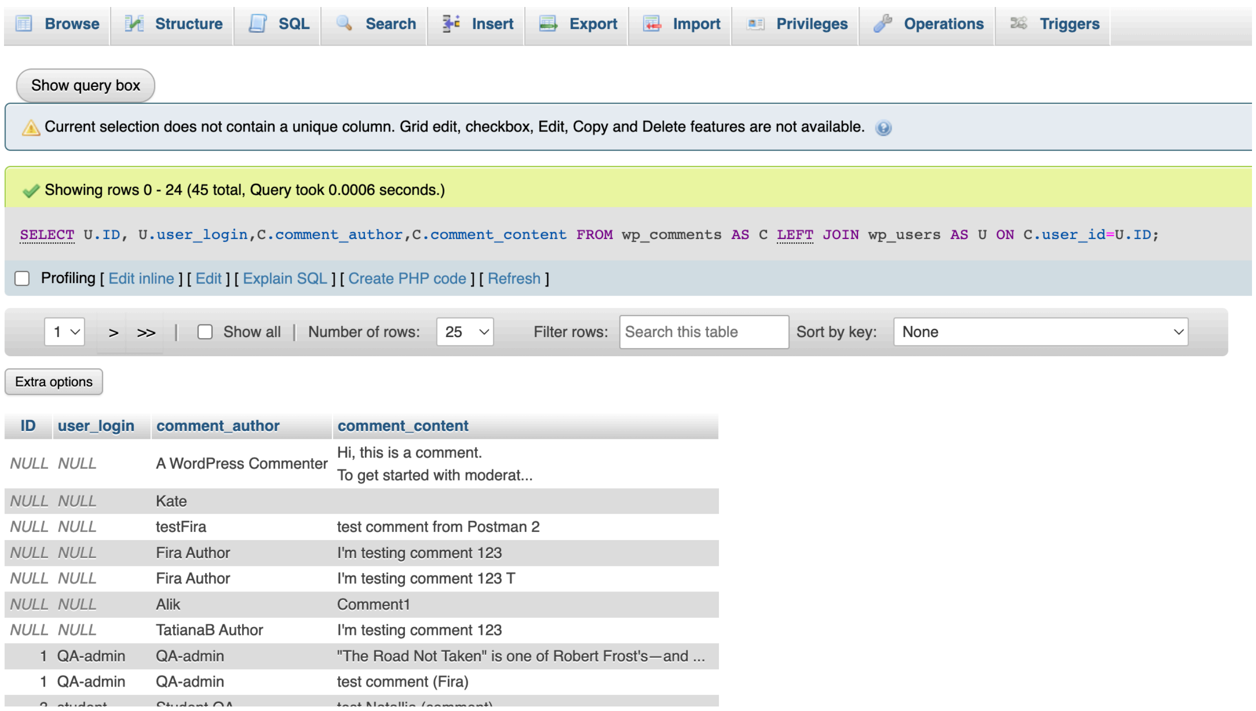
SELECT Left_Table.value, Right_Table.value FROM Left_Table LEFT JOIN Right_Table ON Left_Table.id=Right_Table.id;



id	value
1	left1
2	left2

id	value
1	right1
3	right3
4	right4

value	value
left1	right1
left2	*NULL*


Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Operations
Triggers
Show query box

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

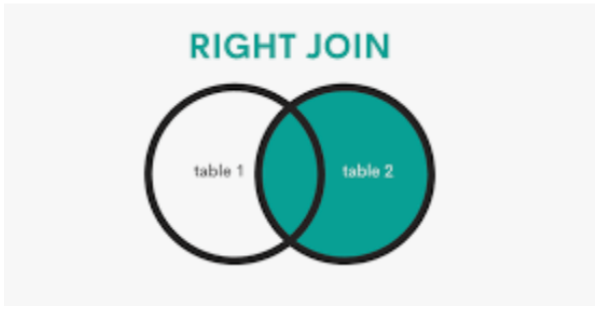
Showing rows 0 - 24 (45 total, Query took 0.0006 seconds.)

```
SELECT U.ID, U.user_login,C.comment_author,C.comment_content FROM wp_comments AS C LEFT JOIN wp_users AS U ON C.user_id=U.ID;
```

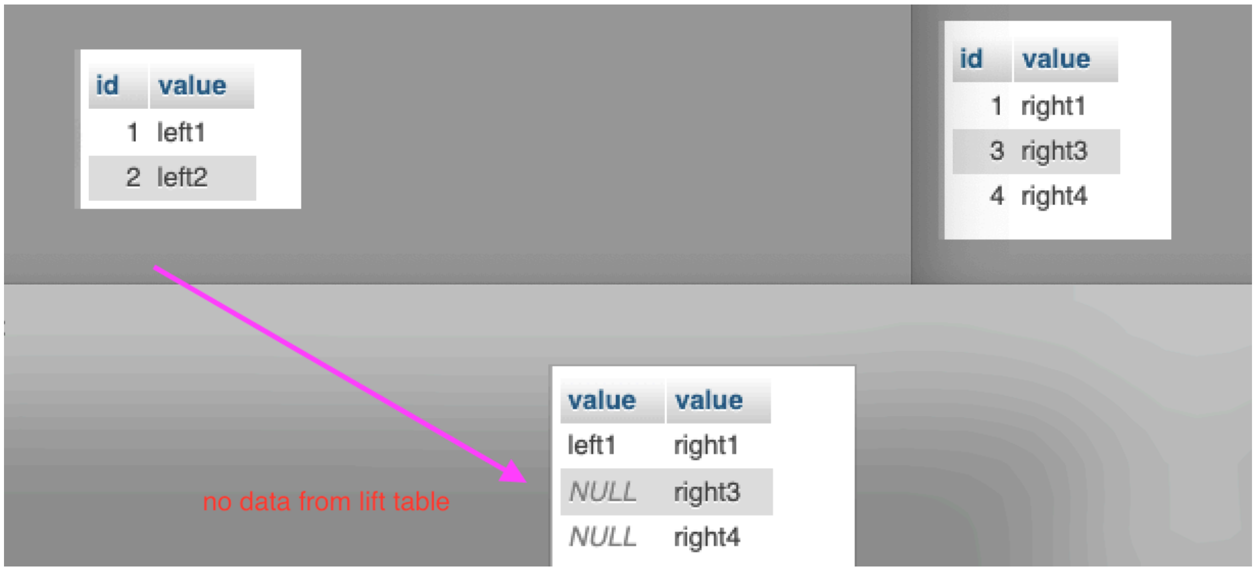
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
1 > >> | Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
Extra options

ID	user_login	comment_author	comment_content
NULL	NULL	A WordPress Commenter	Hi, this is a comment. To get started with moderat...
NULL	NULL	Kate	
NULL	NULL	testFira	test comment from Postman 2
NULL	NULL	Fira Author	I'm testing comment 123
NULL	NULL	Fira Author	I'm testing comment 123 T
NULL	NULL	Alik	Comment1
NULL	NULL	TatianaB Author	I'm testing comment 123
1	QA-admin	QA-admin	"The Road Not Taken" is one of Robert Frost's—and ...
1	QA-admin	QA-admin	test comment (Fira)
2	student	Student QA	test Natallia (comment)

RIGHT (OUTER) JOIN: This returns **all records from the right table**, and the matched records from the left table. If there is no match, the result is NULL on the left side.



```
SELECT Left_Table.value, Right_Table.value FROM Left_Table
RIGHT JOIN Right_Table ON Left_Table.id=Right_Table.id;
```



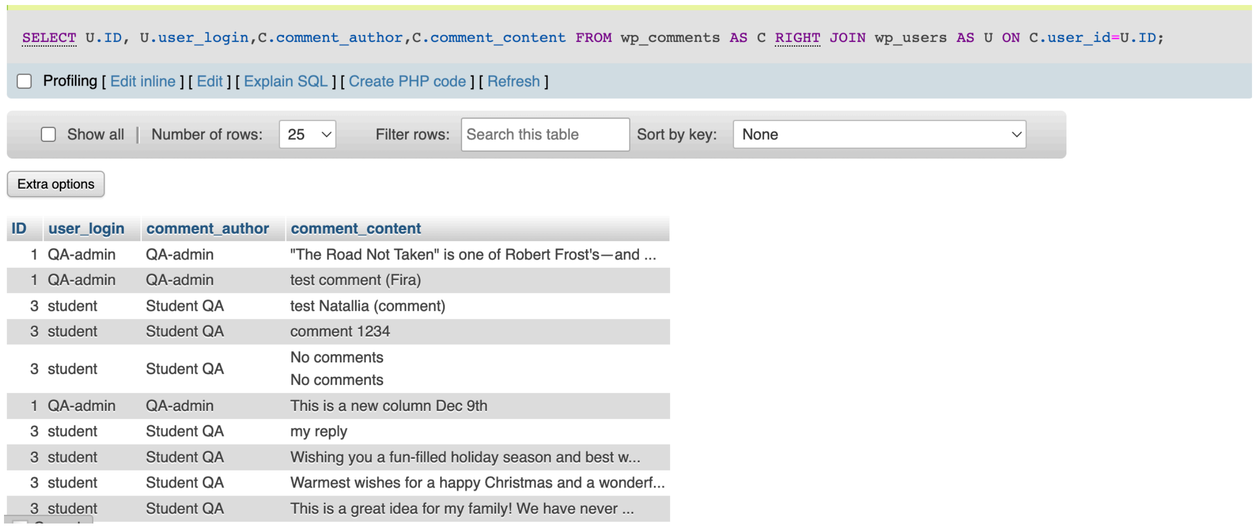
id	value
1	left1
2	left2

id	value
1	right1
3	right3
4	right4

no data from lift table

value	value
left1	right1
NULL	right3
NULL	right4

SELECT U.ID, U.user_login,C.comment_author,C.comment_content FROM wp_comments AS C RIGHT JOIN wp_users AS U ON C.user_id=U.ID;

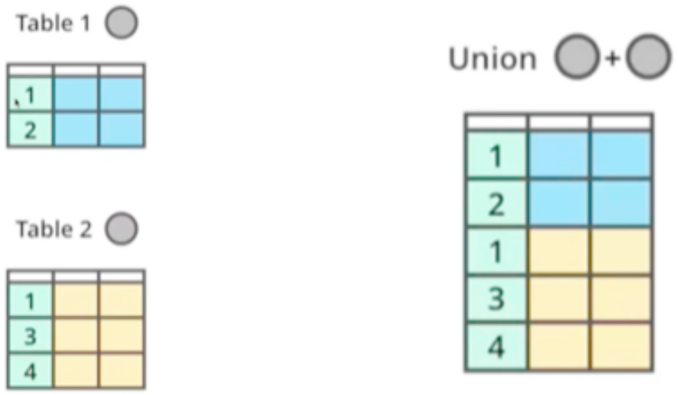


```
SELECT U.ID, U.user_login,C.comment_author,C.comment_content FROM wp_comments AS C RIGHT JOIN wp_users AS U ON C.user_id=U.ID;
```

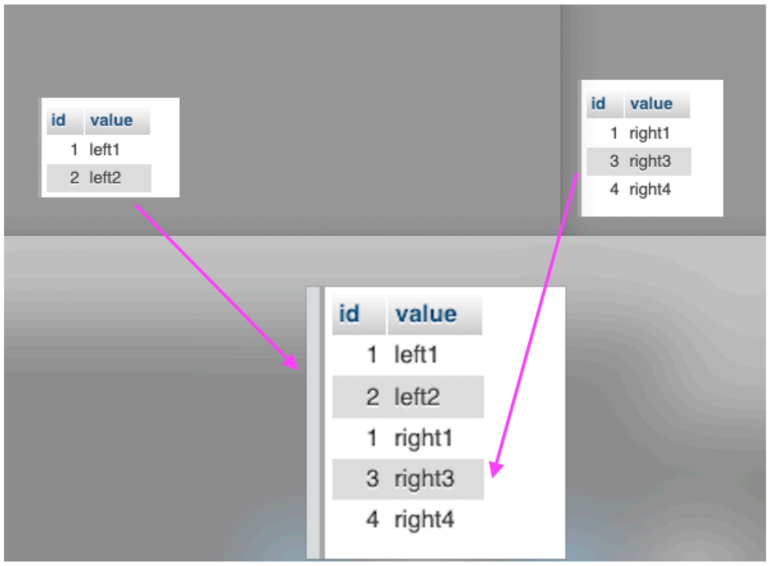
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
Extra options

ID	user_login	comment_author	comment_content
1	QA-admin	QA-admin	"The Road Not Taken" is one of Robert Frost's—and ...
1	QA-admin	QA-admin	test comment (Fira)
3	student	Student QA	test Natallia (comment)
3	student	Student QA	comment 1234
3	student	Student QA	No comments No comments
1	QA-admin	QA-admin	This is a new column Dec 9th
3	student	Student QA	my reply
3	student	Student QA	Wishing you a fun-filled holiday season and best w...
3	student	Student QA	Warmest wishes for a happy Christmas and a wonderf...
3	student	Student QA	This is a great idea for my family! We have never ...

The UNION operator in SQL is used to combine the result set of two or more SELECT statements. However, it removes any duplicate rows from the combined result set. Also, the SELECT statements within the UNION must have the same number of columns and those columns must have similar data types.



SELECT Left_Table.id, Left_Table.value FROM Left_Table UNION SELECT Right_Table.id, Right_Table.value FROM Right_Table;

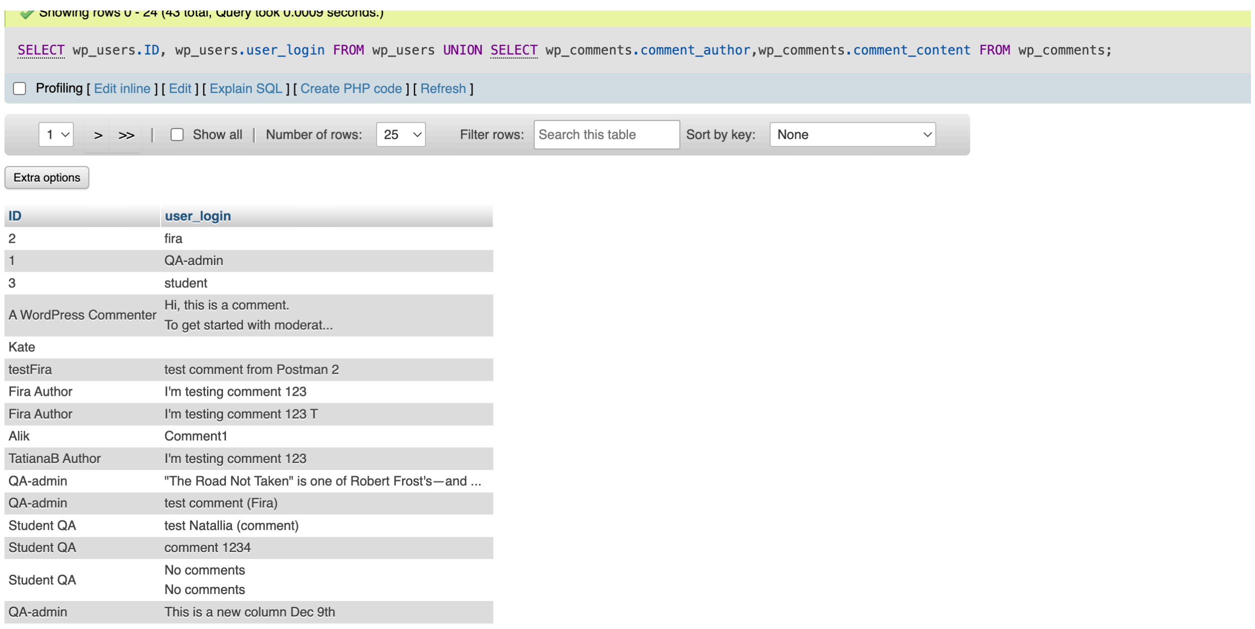


id	value
1	left1
2	left2

id	value
1	right1
3	right3
4	right4

id	value
1	left1
2	left2
1	right1
3	right3
4	right4

SELECT wp_users.ID, wp_users.user_login FROM wp_users
UNION
SELECT wp_comments.comment_author,wp_comments.comment_content

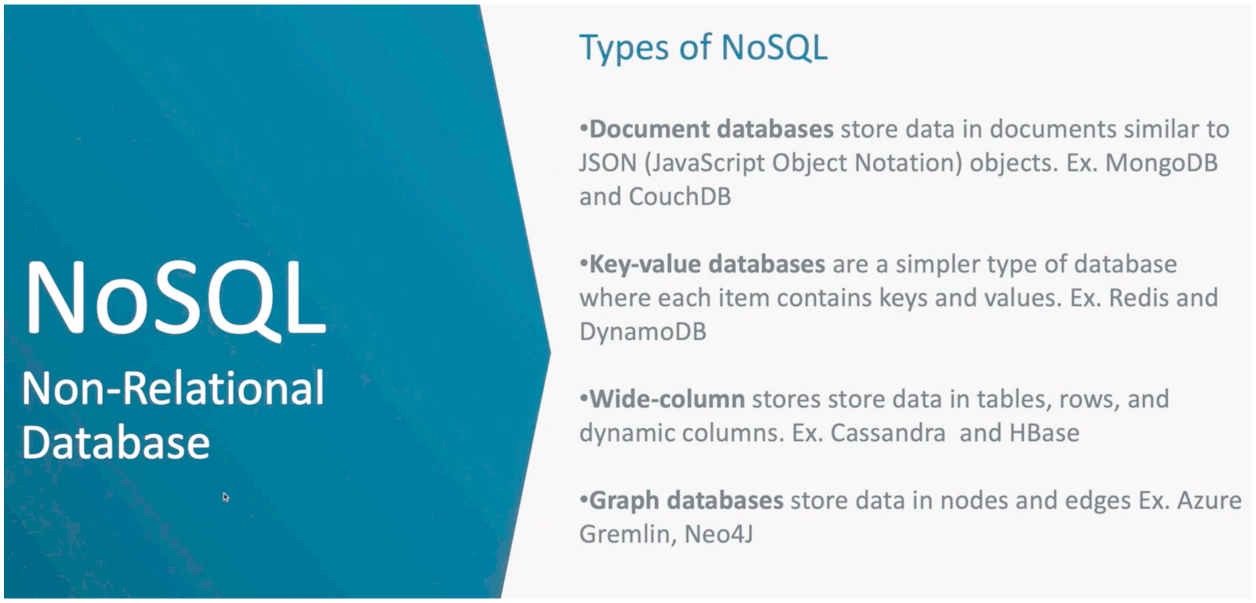


Showing rows 0 - 24 (43 total, Query took 0.0009 seconds.)

```
SELECT wp_users.ID, wp_users.user_login FROM wp_users UNION SELECT wp_comments.comment_author,wp_comments.comment_content FROM wp_comments;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
1 > >> | Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
Extra options

ID	user_login
2	fira
1	QA-admin
3	student
A WordPress Commenter	Hi, this is a comment. To get started with moderat...
Kate	
testFira	test comment from Postman 2
Fira Author	I'm testing comment 123
Fira Author	I'm testing comment 123 T
Alik	Comment1
TatianaB Author	I'm testing comment 123
QA-admin	"The Road Not Taken" is one of Robert Frost's—and ...
QA-admin	test comment (Fira)
Student QA	test Natallia (comment)
Student QA	comment 1234
Student QA	No comments No comments
QA-admin	This is a new column Dec 9th

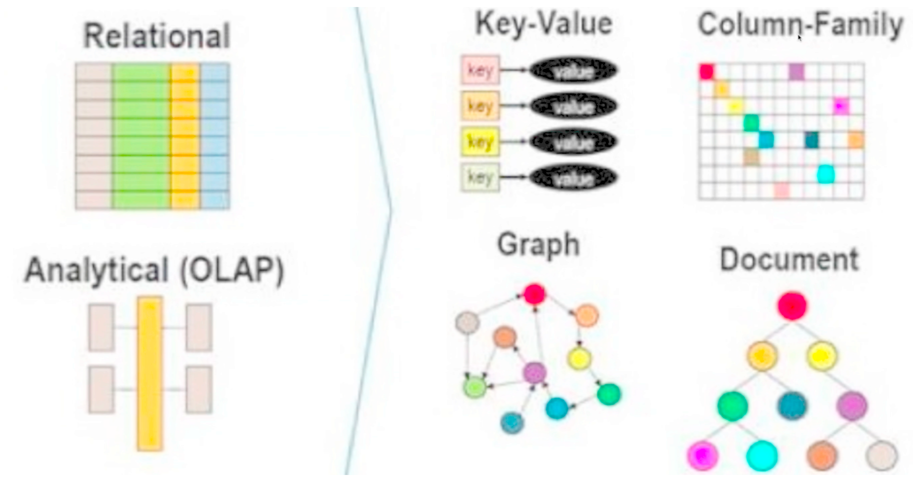


NoSQL

Non-Relational Database

Types of NoSQL

- **Document databases** store data in documents similar to JSON (JavaScript Object Notation) objects. Ex. MongoDB and CouchDB
- **Key-value databases** are a simpler type of database where each item contains keys and values. Ex. Redis and DynamoDB
- **Wide-column** stores store data in tables, rows, and dynamic columns. Ex. Cassandra and HBase
- **Graph databases** store data in nodes and edges Ex. Azure Gremlin, Neo4J



SQL (Structured Query Language) and NoSQL (Not Only SQL) databases are both ways to store and retrieve data, but they provide different mechanisms for storage and retrieval, which makes them useful for different kinds of applications.

Here's a breakdown of some of the key differences:

1. **Structure:** SQL databases are structured, meaning they store data in tables with rows and columns, much like a spreadsheet. This kind of structure is also known as a relational database.
2. 1.1 NoSQL databases, on the other hand, can store data in various ways, including key-value pairs, wide-column stores, graph databases, or document-based stores. This flexibility can make NoSQL databases a good fit for data that doesn't fit neatly into a table.

2. Schema: SQL databases have a predefined schema, or structure, that their data must adhere to. Before you can store data, you need to define what types of data will be stored where.

NoSQL databases are schema-less, meaning you can store any kind of data in any structure you want at any time, which provides flexibility when dealing with unstructured or semi-structured data.

3. Scaling: SQL databases are typically scaled up by increasing the horsepower of the machine they're hosted on (CPU, RAM, SSD). This is called

Vertical Scaling. Conversely, NoSQL databases are typically scaled out by adding more machines to the database. This is called Horizontal Scaling. Horizontal scaling can make NoSQL databases a good fit for handling large amounts of data and high traffic loads.

4. Transactions: SQL databases use ACID transactions (Atomic, Consistent, Isolated, Durable) which assure that any transaction will complete or fail as a whole. This is crucial for applications where consistency of data is important, like banking systems. NoSQL databases typically do not provide ACID transactions, but some do offer eventual consistency.

5. Language: SQL databases use SQL (structured query language) for defining and manipulating the data. It's highly powerful and versatile, allowing you to do things like query specific data, join separate data sets together, and perform complex operations on the data. NoSQL databases have a variety of query languages and APIs to work with. Some have their own languages (like MongoDB's MQL), while others use APIs or even SQL-like languages.

6. Complexity: In general, SQL databases may require more upfront planning and may be more complex to design due to the need for a predefined schema, but they offer robust data integrity through ACID transactions.

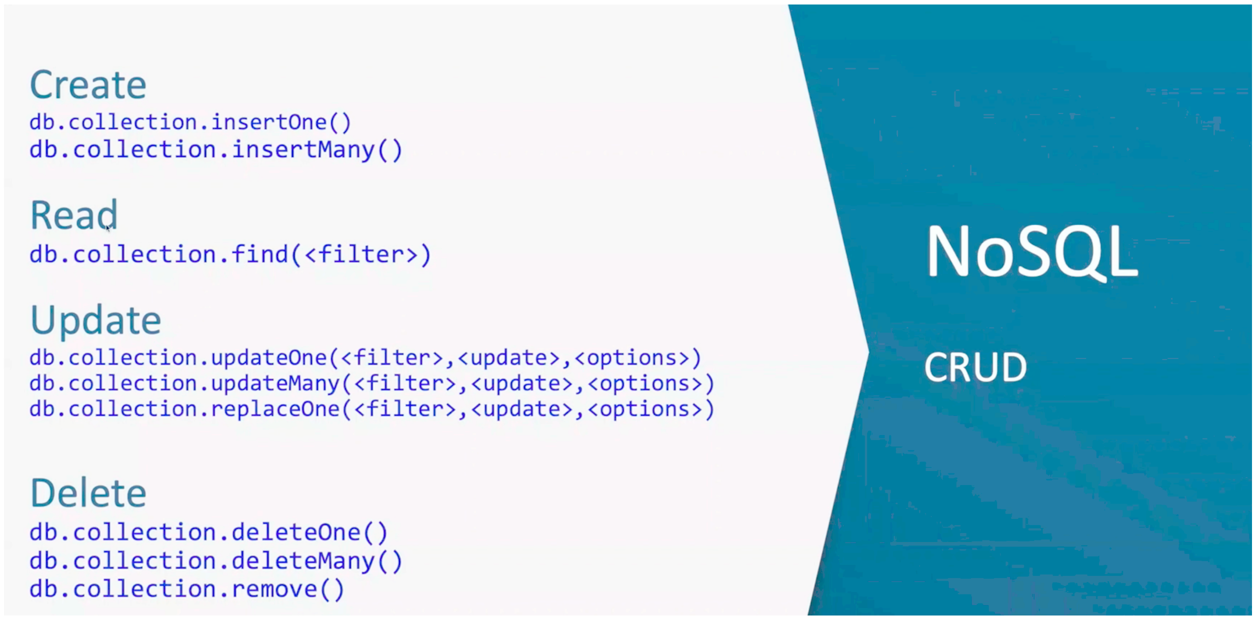
In conclusion, the choice between SQL and NoSQL
will largely depend on the specific requirements and the nature of your project.

SQL is often the choice for projects that involve complex queries and require heavy transactional operations, where data **integrity and consistency over time are key.** Examples of applications that might benefit from an SQL database include accounting systems or systems that manage transactions.

On the other hand, **NoSQL** databases are generally better suited for storing **large volumes of diverse, fast-changing, or real-time data** and for horizontal scaling. They are often used in big data and real-time web applications. Examples include real-time analytics, content management systems, or any scenario where you need to quickly store and retrieve data across large, often unpredictable user bases.

It's worth noting that many organizations today are opting for polyglot persistence, where they use a combination of SQL and NoSQL databases, each purpose-built for different needs within the total application.

Different Syntax



Create

```
db.collection.insertOne()
db.collection.insertMany()
```

Read

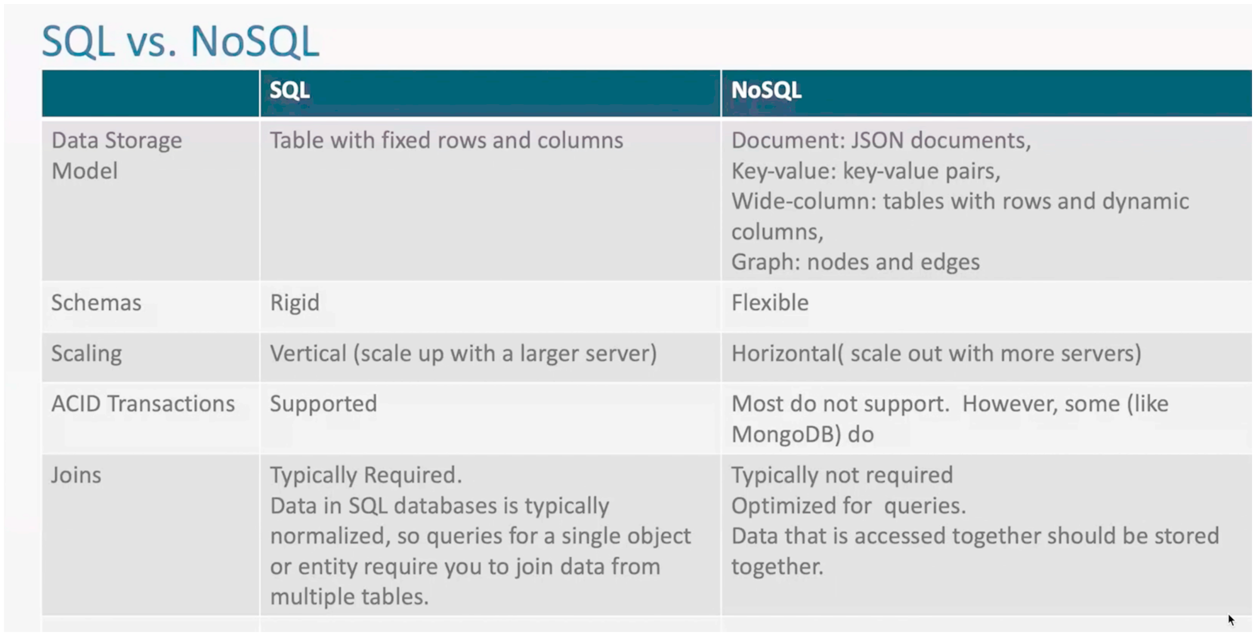
```
db.collection.find(<filter>)
```

Update

```
db.collection.updateOne(<filter>,<update>,<options>)
db.collection.updateMany(<filter>,<update>,<options>)
db.collection.replaceOne(<filter>,<update>,<options>)
```

Delete

```
db.collection.deleteOne()
db.collection.deleteMany()
db.collection.remove()
```



SQL vs. NoSQL

	SQL	NoSQL
Data Storage Model	Table with fixed rows and columns	Document: JSON documents, Key-value: key-value pairs, Wide-column: tables with rows and dynamic columns, Graph: nodes and edges
Schemas	Rigid	Flexible
Scaling	Vertical (scale up with a larger server)	Horizontal(scale out with more servers)
ACID Transactions	Supported	Most do not support. However, some (like MongoDB) do
Joins	Typically Required. Data in SQL databases is typically normalized, so queries for a single object or entity require you to join data from multiple tables.	Typically not required Optimized for queries. Data that is accessed together should be stored together.

Advanced Stuff just FYI:
ACID is an acronym that stands for Atomicity, Consistency, Isolation, and Durability.
These are a set of properties that guarantee reliable processing of data in a database system - particularly for database transactions.

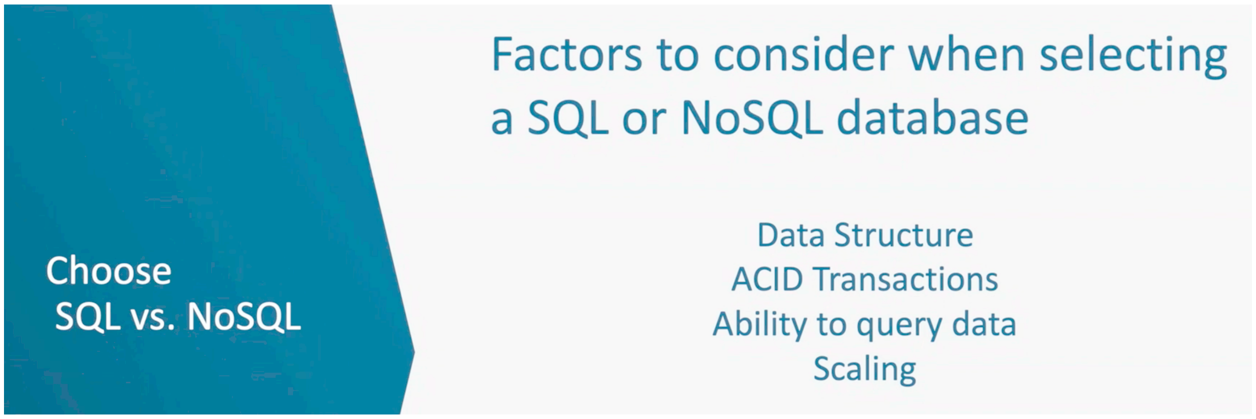
1. **Atomicity**: Refers to the "all or nothing" nature of transactions. This means if a transaction has multiple operations, either all of them are completed successfully or none of them is performed. If any operation fails, the entire transaction is rolled back so that a partial transaction does not occur.

2. **Consistency**: Ensures that all data in a database must meet specified rules (constraints, cascades, triggers, etc.). When a transaction is completed, it must leave the database in a consistent state, meaning the overall integrity of data is maintained.

3. **Isolation**: This property ensures that multiple transactions occurring at the same time won't impact each other's execution. Each transaction should occur in a "transactional bubble" isolated from other simultaneous transactions, maintaining data integrity.

4. **Durability**: Guarantees that once a transaction is committed, it will remain so, even in the event of power loss, crashes, or errors. This property ensures powered-down data persistence in the database.

In summary, ACID properties play a critical role in any database system as they maintain the reliability, functionality, and integrity of data, especially in situations like system failures, concurrent access, database crashes, and error recovery.



Choose SQL vs. NoSQL

Factors to consider when selecting a SQL or NoSQL database

Data Structure
ACID Transactions
Ability to query data
Scaling

SQL Database Example:

A multinational bank uses a SQL database to manage its operations. Customer information, account details, transaction histories, bank branch details, etc. are all stored in a structured format, which is facilitated excellently by a relational SQL database. SQL's efficient querying capabilities are utilized to quickly retrieve specific information - such as transaction details for a particular customer, account status, customers of a particular branch, etc.

Non-SQL Database Example:

A large e-commerce company, such as Amazon, uses a NoSQL database to handle its big data requirements. Product details, customer behaviors, website clicks, and purchase histories are all stored in a non-structured format and analyzed to improve customer experience, provide personalized product recommendations, and predict trends. Because of its capability to store a large volume of diverse data and its scalability, a NoSQL database is ideal for this purpose.

Here are some aspects testers need to know:

1. **SQL Joins:** Understanding INNER JOIN, LEFT JOIN, RIGHT JOIN and FULL JOIN is crucial for manipulating and fetching data from multiple tables.

2. Basic SQL Syntax: Understanding the basics of SQL, such as how to create, read, update, and delete data, as well as how to use WHERE, AND, OR conditions, allows testers to set up, modify, and tear down test data.

3. Aggregation Functions: Knowledge of aggregation functions like COUNT, SUM, AVG, MAX, MIN could be useful in validation of the data.

4. Understanding Relationships: Testers should understand how tables are related via foreign keys, primary keys, and indices. This will allow them to understand the database schema and how changes to one table might affect others.

Advanced security > SQL Injection: (wildcards etc)
It's important to understand SQL injection vulnerabilities and how to prevent them to ensure the application is secured.

SQL Injection is a code injection technique that attackers use to insert malicious SQL code into a query.
The malicious data then produces unanticipated and potentially harmful actions when the database is queried.

Attackers use SQL Injection attacks to manipulate a site's database, usually to extract valuable information such as usernames, passwords, emails, credit card numbers, etc.

For example, suppose there is a login form on a website that queries the database like this:

```sql
SELECT * FROM Users WHERE Username='' AND Password=''
```

An attacker could submit a username of **`admin' --`** and
a password of **`******`**. This modifies the query to:

```sql
SELECT * FROM Users WHERE Username='admin' --' AND
Password='anyvalue'
```

The double dash `--` in SQL represents a comment indicator, and anything following it is ignored. As a result, this query effectively checks if there is a user `admin`, without considering the password.

To prevent SQL Injection attacks, you should:
- Always use Parameterized queries or Prepared Statements.
- Use a Web Application Firewall.
- Regularly update and patch your systems.
- Limit the privileges of database accounts used by web applications.
- Validate and sanitize all user inputs.

Database testing involves checking the integrity, accuracy, and consistency of data in a database-driven application. It's a critical part of a software testing process as it ensures that the application provides accurate information.

The main goal of database testing is to find bugs and errors related to database operations, such as any data leakage, deadlocks, data corruption, performance issues, etc.

Database testing usually involves the following activities:

1. **Schema Testing**: Verifying the database schema to ensure it matches the **expected** schema.

2. **Data Consistency Testing**: Checking the data's correctness and consistency.

3. **Data Integrity Testing**: Checking the integrity of data, especially after operations such as update, delete, insert, and migration.

4. **Data Accuracy Testing**: Validating the data inside the database to ensure accuracy.

5. **Performance Testing**: Assessing the database's performance and optimizing the queries for better speed.

6. **Security Testing**: Ensuring that only authorized personnel have access to the database and checking the database's vulnerability to attacks like SQL Injection.

7. **ACID Properties Testing**: Verifying Atomicity, Consistency, Isolation, and Durability properties.

8. **Functional Flow Testing**: Checking for any functional errors during user flow, such as creating a new record

A Database Management Client is a computer program that provides a way to interact with the database. These can be command-line tools or graphical interfaces that let you connect to the database server, run SQL commands, browse and edit data, manage database objects (such as tables, indexes, keys), export and import data, debug, profile and more.

Here are some common Database Management Clients for different databases:

1. **MySQL Workbench**: It's a comprehensive tool for MySQL database management. It provides data modeling, SQL development, and easy administration.

2. **phpMyAdmin**: A web interface that is widely used for managing MySQL databases.

3. **SSMS (SQL Server Management Studio)**: It's a software application used for configuring, managing, and administering all components within Microsoft SQL Server.

4. **Oracle SQL Developer**: An integrated development environment that simplifies the development and management of Oracle Database.

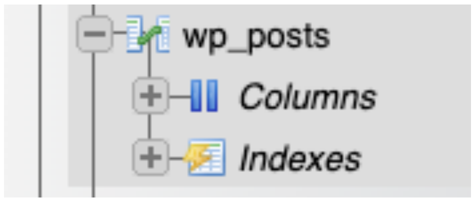
5. **pgAdmin**: A popular open-source and feature-rich administration and management tool for PostgreSQL.

6. **MongoDB Compass**: The GUI for MongoDB which allows you to visually explore your data, run ad-hoc queries, and more.

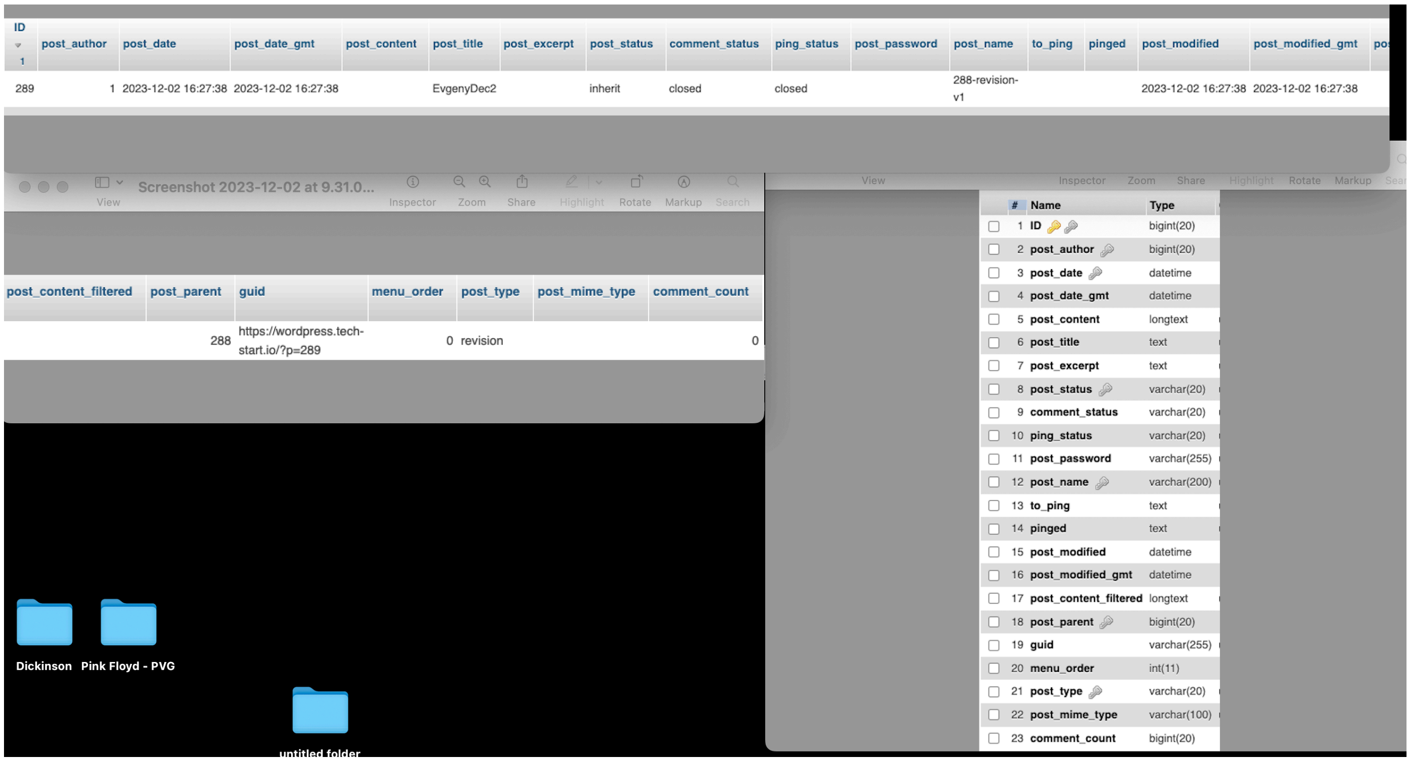
The choice of a client often depends on what database you're using, your specific needs, and personal preferences.

Homework

Create a post from backend.



How to understand data in columns



ID	post_author	post_date	post_date_gmt	post_content	post_title	post_excerpt	post_status	comment_status	ping_status	post_password	post_name	to_ping	pinged	post_modified	post_modified_gmt	po
289	1	2023-12-02 16:27:38	2023-12-02 16:27:38		EvgenyDec2		inherit	closed	closed		288-revision-v1			2023-12-02 16:27:38	2023-12-02 16:27:38	

post_content_filtered	post_parent	guid	menu_order	post_type	post_mime_type	comment_count
	288	https://wordpress.tech-start.io/?p=289	0	revision		0

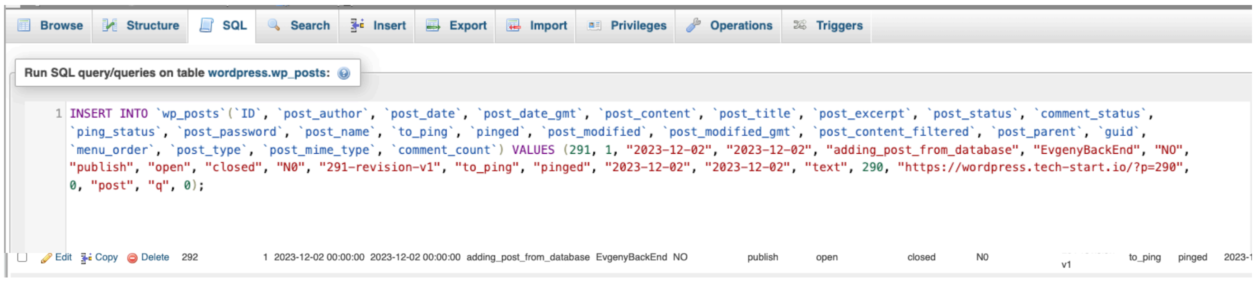
#	Name	Type
1	ID	bigint(20)
2	post_author	bigint(20)
3	post_date	datetime
4	post_date_gmt	datetime
5	post_content	longtext
6	post_title	text
7	post_excerpt	text
8	post_status	varchar(20)
9	comment_status	varchar(20)
10	ping_status	varchar(20)
11	post_password	varchar(255)
12	post_name	varchar(200)
13	to_ping	text
14	pinged	text
15	post_modified	datetime
16	post_modified_gmt	datetime
17	post_content_filtered	longtext
18	post_parent	bigint(20)
19	guid	varchar(255)
20	menu_order	int(11)
21	post_type	varchar(20)
22	post_mime_type	varchar(100)
23	comment_count	bigint(20)

Query

```
INSERT INTO `wp_posts`(`ID`, `post_author`, `post_date`, `post_date_gmt`,
`post_content`, `post_title`, `post_excerpt`, `post_status`, `comment_status`,
`ping_status`, `post_password`, `post_name`, `to_ping`, `pinged`,
`post_modified`, `post_modified_gmt`, `post_content_filtered`, `post_parent`,
`guid`, `menu_order`, `post_type`, `post_mime_type`, `comment_count`)
VALUES
(290,
 1,
 2023-12-02,
 2023-12-02,
"adding_post_from_database",
 "EvgenyBackEnd",
 "NO",
 "inherit",
"closed",
"closed",
 "N0",
 "290-revision-v1",
 "to_ping",
 "pinged",
 "2023-12-02",
 "2023-12-02",
 "text",
 289,
 "https://wordpress.tech-start.io/?p=290",
 0,
 "revision",
 "q",
 0);
```

OR

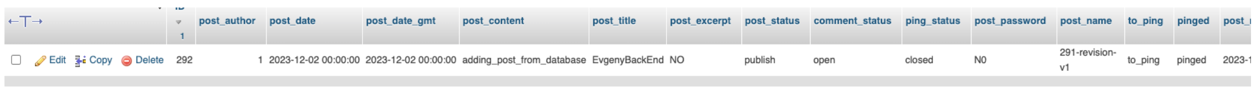
INSERT INTO `wp_posts`(`ID`, `post_author`, `post_date`, `post_date_gmt`, `post_content`, `post_title`, `post_excerpt`, `post_status`, `comment_status`, `ping_status`, `post_password`, `post_name`, `to_ping`, `pinged`, `post_modified`, `post_modified_gmt`, `post_content_filtered`, `post_parent`, `guid`, `menu_order`, `post_type`, `post_mime_type`, `comment_count`) VALUES (**291**, 1, "2023-12-02", "2023-12-02", "adding_post_from_database", "EvgenyBackEnd", "NO", "publish", "open", "closed", "N0", "291-revision-v1", "to_ping", "pinged", "2023-12-02", "2023-12-02", "text", 290, "https://wordpress.tech-start.io/?p=290", 0, "post", "q", 0);



```
INSERT INTO `wp_posts`(`ID`, `post_author`, `post_date`, `post_date_gmt`, `post_content`, `post_title`, `post_excerpt`, `post_status`, `comment_status`,
`ping_status`, `post_password`, `post_name`, `to_ping`, `pinged`, `post_modified`, `post_modified_gmt`, `post_content_filtered`, `post_parent`, `guid`,
`menu_order`, `post_type`, `post_mime_type`, `comment_count`) VALUES (291, 1, "2023-12-02", "2023-12-02", "adding_post_from_database", "EvgenyBackEnd", "NO",
"publish", "open", "closed", "N0", "291-revision-v1", "to_ping", "pinged", "2023-12-02", "2023-12-02", "text", 290, "https://wordpress.tech-start.io/?p=290",
0, "post", "q", 0);
```

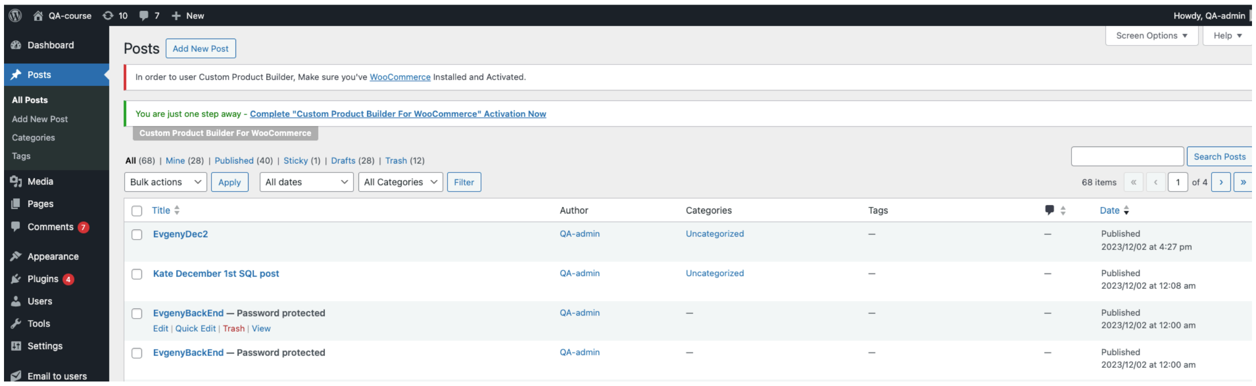
```
Edit  Copy  Delete  292   1  2023-12-02 00:00:00  2023-12-02 00:00:00  adding_post_from_database  EvgenyBackEnd  NO   publish   open   closed   N0   v1   to_ping   pinged   2023-1
```

Create a query and find you post



←T→	ID 1	post_author	post_date	post_date_gmt	post_content	post_title	post_excerpt	post_status	comment_status	ping_status	post_password	post_name	to_ping	pinged	post_
Edit Copy Delete	292	1	2023-12-02 00:00:00	2023-12-02 00:00:00	adding_post_from_database	EvgenyBackEnd	NO	publish	open	closed	N0	291-revision-v1	to_ping	pinged	2023-1

Validate it from frontend



Posts Add New Post

In order to user Custom Product Builder, Make sure you've WooCommerce Installed and Activated.

You are just one step away - Complete "Custom Product Builder For WooCommerce" Activation Now

Custom Product Builder For WooCommerce

All (68) | Mine (28) | Published (40) | Sticky (1) | Drafts (28) | Trash (12)

Bulk actions Apply All dates All Categories Filter Search Posts

68 items « ‹ 1 of 4 › »

Title	Author	Categories	Tags	💬	Date
EvgenyDec2	QA-admin	Uncategorized	—	—	Published 2023/12/02 at 4:27 pm
Kate December 1st SQL post	QA-admin	Uncategorized	—	—	Published 2023/12/02 at 12:08 am
EvgenyBackEnd — Password protected Edit | Quick Edit | Trash | View	QA-admin	—	—	—	Published 2023/12/02 at 12:00 am
EvgenyBackEnd — Password protected	QA-admin	—	—	—	Published 2023/12/02 at 12:00 am