

MySQL Indexes

Indexes in MySQL are used to **speed up data retrieval**. They work like the index of a book — helping the database engine find rows faster, especially for **searches, filters, and joins**.

Viewing Indexes on a Table

To see the indexes on a table, use:

```
SHOW INDEXES FROM users;
```

This shows all the indexes currently defined on the `users` table, including the automatically created **primary key** index.

Creating a Single-Column Index

Suppose you're frequently searching users by their `email`. You can speed this up by indexing the `email` column.

```
CREATE INDEX idx_email ON users(email);
```

What this does:

- Creates an index named `idx_email`
- Improves performance of queries like:

```
SELECT * FROM users WHERE email = 'example@example.com';
```

Important Notes

- Indexes **consume extra disk space**
 - Indexes **slow down** `INSERT` , `UPDATE` , and `DELETE` operations slightly (because the index must be updated)
 - Use indexes **only when needed** (i.e., for columns used in `WHERE` , `JOIN` , `ORDER BY`)
-

Creating a Multi-Column Index

If you often query users using **both** `gender` and `salary` , a multi-column index is more efficient than separate indexes.

```
CREATE INDEX idx_gender_salary ON users(gender, salary);
```

Usage Example:

```
SELECT * FROM users  
WHERE gender = 'Female' AND salary > 70000;
```

This query can take advantage of the combined index on `gender` and `salary` .

Index Order Matters

For a multi-column index on `(gender, salary)` :

- This works efficiently:

```
WHERE gender = 'Female' AND salary > 70000
```

- But this **may not** use the index effectively:

```
WHERE salary > 70000
```

Because the first column in the index (`gender`) is missing in the filter.

Dropping an Index

To delete an index:

```
DROP INDEX idx_email ON users;
```

Summary

Feature	Description
<code>SHOW INDEXES</code>	View current indexes on a table
<code>CREATE INDEX</code>	Create single or multi-column indexes
<code>DROP INDEX</code>	Remove an index
Use when	Query performance on large tables is a concern
Avoid on	Columns that are rarely queried or always unique