INDEX, VIEW, TRIGGER

How MySQL stores data (by default)

- A MySQL server can store several databases
- Databases are stored as directories
  - Default is at /usr/local/mysql/var/
- Tables are stored as files inside each database (directory)
- For each table, it has three files:
  - table.FRM file containing information about the table structure
  - table.MYD file containing the row data
  - table.MYI containing any indexes belonging with this table, as well as some statistics about the table.

How to Create Index?

https://dev.mysql.com/doc/refman/8.0/en/create-index.html

```
CREATE [UNIQUE | FULLTEXT | SPATIAL] INDEX index_name
    [index_type]
    ON tbl_name (key_part,...)
    [index_option]
    [algorithm_option | lock_option] ...

key_part: {col_name [(length)] | (expr)} [ASC | DESC]

index_option:
    KEY_BLOCK_SIZE [=] value
  | index_type
  | WITH PARSER parser_name
  | COMMENT 'string'
  | {VISIBLE | INVISIBLE}

index_type:
    USING {BTREE | HASH}

algorithm_option:
    ALGORITHM [=] {DEFAULT | INPLACE | COPY}

lock_option:
    LOCK [=] {DEFAULT | NONE | SHARED | EXCLUSIVE}
```

```
CREATE INDEX index_name
ON table_name (column1, column2, ...);
```

```
CREATE UNIQUE INDEX index_name
ON table_name (column1, column2, ...);
```

```
CREATE TABLE test (blob_col BLOB, INDEX(blob_col(10)));
```

ALTER TABLE tbl_name ADD UNIQUE index_name (column_list)

EXPLAIN Command

https://dev.mysql.com/doc/refman/8.0/en/execution-plan-information.html

A tool to explain the output of EXPLAIN command
http://explain.plosquare.com/

What is EXPLAIN Command?
A tool to understand how database execute the query (Query Execution Plan)

```
mysql> EXPLAIN SELECT * FROM breakup WHERE x_id = 2

id: 1
select_type: SIMPLE
Table: breakup
type: ALL
possible_keys: NULL
key: NULL
key_len: NULL
ref: NULL
rows: 6301
Extra: Using where
```

```
mysql> EXPLAIN SELECT * FROM transaction WHERE x_id = 2

id: 1
select_type: SIMPLE
table: breakup
type: ref
possible_keys: x_id
key: x_id
key_len: 4
ref: const
rows: 2
Extra: Using index condition; Using filesort
```

EXPLAIN Command

https://dev.mysql.com/doc/refman/8.0/en/execution-plan-information.html

```
mysql> EXPLAIN SELECT transaction.id, transaction.status, payment.y_id FROM transaction
INNER JOIN `payment` ON transaction.payment_type = payment.type;

id: 1
select_type: SIMPLE
Table: transaction
type: ALL
possible_keys: NULL
key: NULL
key_len: NULL
ref: NULL
rows: 4542
Extra: null

id: 1
select_type: SIMPLE
Table: payment
type: ALL
possible_keys: NULL
key: NULL
key_len: NULL
ref: NULL
rows: 7748
Extra: Using where; Using join buffer (Block Nested Loop)
```

```
id: 1
select_type: SIMPLE
Table: transaction
type: index
possible_keys: payment_type
key: payment_type
key_len: 203
ref: NULL
rows: 4542
Extra: Using where; Using index

id: 1
select_type: SIMPLE
Table: payment
type: ref
possible_keys: type
key: type
key_len: 82
ref: transaction.payment_type
rows: 553
Extra: Using where; Using index
```

VIEW

https://dev.mysql.com/doc/refman/8.0/en/views.html

Advantages

- allows you to simplify complex queries
- helps limit data access to specific users
- view provides extra security layer (Read Only)
- enables computed columns

Disadvantages

- Higher latency
- Table dependency

```
mysql> CREATE TABLE t (qty INT, price INT);
mysql> INSERT INTO t VALUES(3, 50), (5, 60);
mysql> CREATE VIEW v AS SELECT qty, price, qty*price AS value FROM t;
mysql> SELECT * FROM v;
+------+-------+-------+
| qty  | price | value |
+------+-------+-------+
|    3 |    50 |   150 |
|    5 |    60 |   300 |
+------+-------+-------+
mysql> SELECT * FROM v WHERE qty = 5;
+------+-------+-------+
| qty  | price | value |
+------+-------+-------+
|    5 |    60 |   300 |
+------+-------+-------+
```

Updatable VIEW (INSERT, DELETE, UPDATE)

https://dev.mysql.com/doc/refman/8.0/en/view-updatability.html

A view is not updatable if it contains any of the following:

- Aggregate functions or window functions (SUM(), MIN(), MAX(), COUNT(), and so forth)
- DISTINCT, GROUP BY, HAVING, UNION, UNION ALL
- Dependent subquery
- Reference to nonupdatable view in the FROM clause
- Subquery in the WHERE clause that refers to a table in the FROM clause
- Refers only to literal values
- ALGORITHM = TEMPTABLE
- Multiple references to any column of a base table (okay for UPDATE, DELETE)

Updatable VIEW (INSERT, DELETE, UPDATE)

https://dev.mysql.com/doc/refman/8.0/en/view-updatability.html

```
CREATE TABLE t1 (x INTEGER);
CREATE TABLE t2 (c INTEGER);
CREATE VIEW vmat AS SELECT SUM(x) AS s FROM t1;
CREATE VIEW vup AS SELECT * FROM t2;
CREATE VIEW vjoin AS SELECT * FROM vmat JOIN vup ON vmat.s=vup.c;
```

```
INSERT INTO vjoin (c) VALUES (1);
```

This statement is invalid because one component of the join view is nonupdatable

```
INSERT INTO vup (c) VALUES (1);
```

```
UPDATE vjoin SET c=c+1;
```

```
UPDATE vjoin SET x=x+1;
```

```
UPDATE vup JOIN (SELECT SUM(x) AS s FROM t1) AS dt ON ...
SET c=c+1;
```

```
UPDATE vup JOIN (SELECT SUM(x) AS s FROM t1) AS dt ON ...
SET s=s+1;
```

Updatable VIEW (INSERT, DELETE, UPDATE)

https://dev.mysql.com/doc/refman/8.0/en/view-updatability.html

```
CREATE TABLE t1 (x INTEGER);
CREATE TABLE t2 (c INTEGER);
CREATE VIEW vmat AS SELECT SUM(x) AS s FROM t1;
CREATE VIEW vup AS SELECT * FROM t2;
CREATE VIEW vjoin AS SELECT * FROM vmat JOIN vup ON vmat.s=vup.c;
```

```
DELETE vjoin WHERE ...;
```

```
DELETE vup WHERE ...;
```

```
DELETE vup FROM vup JOIN (SELECT SUM(x) AS s FROM t1) AS dt ON ...;
```

TRIGGER

https://dev.mysql.com/doc/refman/5.5/en/trigger-syntax.html

```
CREATE
    [DEFINER = { user | CURRENT_USER }]
    TRIGGER trigger_name
    trigger_time trigger_event
    ON tbl_name FOR EACH ROW
    trigger_body

trigger_time: { BEFORE | AFTER }

trigger_event: { INSERT | UPDATE | DELETE }
```

```
mysql> CREATE TABLE account (acct_num INT, amount DECIMAL(10,2));
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TRIGGER ins_sum BEFORE INSERT ON account
       FOR EACH ROW SET @sum = @sum + NEW.amount;
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> SET @sum = 0;
mysql> INSERT INTO account VALUES(137,14.98),(141,1937.50),(97,-100.00);
mysql> SELECT @sum AS 'Total amount inserted';
+-----------------------+
| Total amount inserted |
+-----------------------+
|               1852.48 |
+-----------------------+
```

TRIGGER

https://dev.mysql.com/doc/refman/5.5/en/trigger-syntax.html

```
mysql> Create table Student_age(age INT, Name Varchar(35));
```

```
mysql> DELIMITER //
mysql> Create Trigger before_inser_studentage BEFORE INSERT ON student_age FOR EACH ROW
BEGIN
IF NEW.age < 0 THEN SET NEW.age = 0;
END IF;
END //
```

```
mysql> INSERT INTO Student_age(age, Name) values(30, 'Rahul');
mysql> INSERT INTO Student_age(age, Name) values(-10, 'Harshit');
mysql> Select * from Student_age;
+------+---------+
| age  | Name    |
+------+---------+
| 30   | Rahul   |
| 0    | Harshit |
```

Compound Statement: https://dev.mysql.com/doc/refman/5.7/en/sql-syntax-compound-statements.html