CREATE INDEX 语句

> CREATE INDEX Statement：https://dev.mysql.com/doc/refman/8.0/en/create-index.html

CREATE INDEX 语句可以向已有的表添加索引。

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

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

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

index_type:
    USING {BTREE | HASH}

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

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

根据索引的结构，可以把索引分为：

- 部分索引 (partial)：使用了列值的一部分，通常是前缀 (指定 length)
- 单列索引 (column)：只使用了一列
- 复合索引 (composite)：使用了多列

除了普通的索引，还有一些特殊类型的索引：

- 唯一索引 (unique)：被索引的列添加一个唯一约束，不能有重复的值
- 全文索引 (fulltext)：用于全文检索操作
- 空间索引 (spatial)：用于地理位置相关的数据类型

唯一索引

> unique index：https://dev.mysql.com/doc/refman/8.0/en/glossary.html#glos_unique_index

对于不能有重复值的列可以添加唯一索引。唯一索引不仅是一种约束，能够防止插入或更新造成的重复值，而且由于每个值都是唯一的，部分查询操作和 count 操作能够以更高效的方式完成。

添加了唯一索引的列不能有重复的值，如果索引只包含了列值的一部分 (指定了 length)，那么在前缀范围内不能有重复的值。如果该列允许有 NULL 值，则可以有多个 NULL 值。

```
create unique index id_uni_idx
    on test (id);
```

SHOW INDEX 语句

> SHOW INDEX Statement：https://dev.mysql.com/doc/refman/8.0/en/show-index.html

```
SHOW [EXTENDED] {INDEX | INDEXES | KEYS}
    {FROM | IN} tbl_name
    [{FROM | IN} db_name]
    [WHERE expr]
```

EXPLAIN 语句

> EXPLAIN Statement：https://dev.mysql.com/doc/refman/8.0/en/explain.html
>
> EXPLAIN Output Format：https://dev.mysql.com/doc/refman/8.0/en/explain-output.html
>
> Optimizing Queries with EXPLAIN：https://dev.mysql.com/doc/refman/8.0/en/using-explain.html

EXPLAIN 语句用于查看 MySQL 对语句的执行计划。

```
{EXPLAIN | DESCRIBE | DESC}
    [explain_type]
    {explainable_stmt | FOR CONNECTION connection_id}

explain_type: {
    FORMAT = format_name
}

format_name: {
    TRADITIONAL
```

```
    | JSON
    | TREE
}

explainable_stmt: {
    SELECT statement
  | TABLE statement
  | DELETE statement
  | INSERT statement
  | REPLACE statement
  | UPDATE statement
}
```

EXPLAIN 语句的输出结果中，有 3 个字段可以帮助了解索引的使用情况：

列名	含义	备注
possible_keys	可能使用的索引	where 子句中涉及到的列上的索引都会被列出，但不一定被使用
key	真正使用的索引	在 possible_keys 中选择一个最合适的索引或可以用 index scan 的索引
key_len	使用索引的长度	常用于多列索引

> 当 possible_keys 中的索引都不适合真正使用，但是有另一个索引中的列包含了 select 子句中的所有列 (也就是覆盖索引，covering index)，虽然这个索引不能帮助快速筛选满足条件的行，但是返回索引中的值要比返回行中的值快，所以查找到满足条件的行后直接返回索引中的列值，也就是 index scan 而不是 table scan。
>
> 这时 key 中的索引没有在 possible_keys 中列出。

CREATE TRIGGER 语句

> CREATE TRIGGER Statement：https://dev.mysql.com/doc/refman/8.0/en/create-trigger.html

```
CREATE
    [DEFINER = user]
    TRIGGER trigger_name
    trigger_time trigger_event
    ON tbl_name FOR EACH ROW
    [trigger_order]
    trigger_body

trigger_time: { BEFORE | AFTER }

trigger_event: { INSERT | UPDATE | DELETE }

trigger_order: { FOLLOWS | PRECEDES } other_trigger_name
```

- 触发器名称 trigger_name 位于 schema 命名空间，同一数据库、同一用户下的所有触发器不能重名。
- 当一个表上有多个 trigger_time 和 trigger_event 都相同的触发器，默认以创建顺序触发，也可以通过定义 trigger_order 来改变默认顺序：other_trigger_name 是当前表上另一个相同 trigger_time 和 trigger_event 的触发器，FOLLOWS 表示在其之后，PRECEDES 表示在其之前。
- trigger_body 是触发器触发时执行的语句，若要使用多条语句，需要用 BEGIN...END。在 trigger_body 内可以引用表内数据，使用 `OLD.col_name` 来引用**更新前**或**删除前**的数据，使用 `NEW.col_name` 来引用**插入后**或**更新后**的数据。