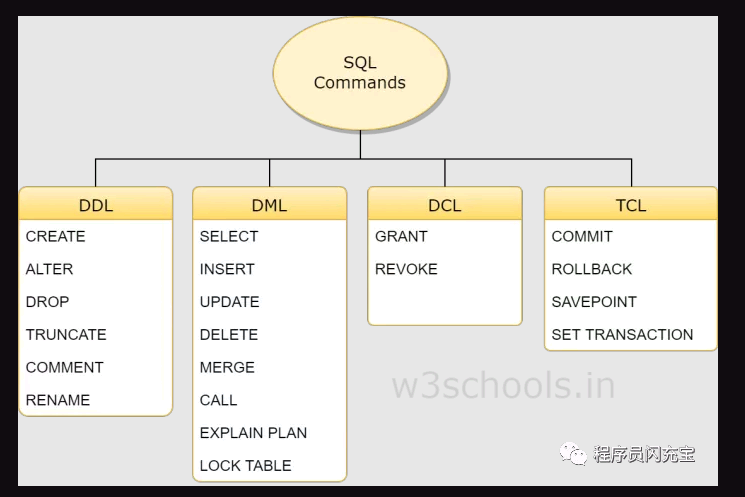
# Sql命令

Sql命令分为四组：ddl,dml,dcl和tcl



## DDL

DDL是数据定义语言（Data Definition Language）的简称，它处理数据库schemas和描述数据应如何驻留在数据库中。

CREATE：创建数据库及其对象（如表，索引，视图，存储过程，函数和触发器） ALTER：改变现有数据库的结构

DROP：从数据库中删除对象

TRUNCATE：从表中删除所有记录，包括为记录分配的所有空间都将被删除 COMMENT：添加注释

RENAME：重命名对象

### 常用命令

#### 建表

CREATE TABLE sicimike (

id int(4) primary key auto\_increment COMMENT '主键ID',

name varchar(10) unique,

age int(3) default 0,

identity\_card varchar(18)

# PRIMARY KEY (id) // 也可以通过这种方式设置主键

# UNIQUE KEY (name) // 也可以通过这种方式设置唯一键

# key/index (identity\_card, col1...) // 也可以通过这种方式创建索引

) ENGINE = InnoDB;

设置主键

Alter table sicimike add primary key(id);

删除主键

Alter table sicimike drop primary key;

设置唯一主键

Alter table sicimike drop unique key(column\_name)

删除唯一主键

Alter table sicimike drop index column\_name;

创建索引

alter table sicimike add [unique/fulltext/spatial] index/key index\_name (identity\_card[(len)] [asc/desc])[using btree/hash]

create [unique/fulltext/spatial] index index\_name on sicimike(identity\_card[(len)] [asc/desc])[using btree/hash]

example：alter table sicimike add index idx\_na(name, age);

删除索引

Alter table sicimike drop key/index identity\_card;

Drop inde index\_name on sicimike;

查看索引

show index from sicimike;

# 查看列

desc sicimike;

# 新增列

alter table sicimike add column column\_name varchar(30);

# 删除列

alter table sicimike drop column column\_name;

# 修改列名

alter table sicimike change column\_name new\_name varchar(30);

# 修改列属性

alter table sicimike modify column\_name varchar(22);

# 查看建表信息

show create table sicimike;

# 添加表注释

alter table sicimike comment '表注释';

# 添加字段注释

alter table sicimike modify column column\_name varchar(10) comment '姓名';

## DML

DML是**数据操纵语言**（Data Manipulation Language）的简称，包括最常见的SQL语句，例如SELECT，INSERT，UPDATE，DELETE等，它用于**存储**，**修改**，**检索**和**删除**数据库中的数据。

分页

-- 查询从第11条数据开始的连续5条数据

select \* from sicimike limit 10, 5

group by

默认情况下，MySQL中的分组（group by）语句，不要求select返回的列，必须是分组的列或者是一个聚合函数。如果select查询的列不是分组的列，也不是聚合函数，则会返回该分组中第一条记录的数据。对比下面两条SQL语句，第二条SQL语句中，cname既不是分组的列，也不是以聚合函数的形式出现。所以在liming这个分组中，cname取的是第一条数据。

having

having关键字用于对分组后的数据进行筛选，功能相当于分组之前的where，不过要求更严格。过滤条件要么是一个聚合函数( ... having count(x) > 1)，要么是出现在select后面的列(select col1, col2 ... group by x having col1 > 1)

多表更新

- update tableA a inner join tableB b on a.xxx = b.xxx set a.col1 = xxx, b.col1 = xxx where ...

多表删除

- delete a, b from tableA a inner join tableB b on a.xxx = b.xxx where a.col1 = xxx and b.col1 = xxx

## DCL

DCL是数据控制语言（Data Control Language）的简称，它包含诸如GRANT之类的命令，并且主要涉及数据库系统的权限，权限和其他控件。

GRANT ：允许用户访问数据库的权限

REVOKE：撤消用户使用GRANT命令赋予的访问权限

# 范式

数据库规范化，又称正规化、标准化，是数据库设计的一系列原理和技术，以减少数据库中数据冗余，增进数据的一致性。

现在数据库设计**最多满足3NF**，普遍认为范式过高，虽然具有对数据关系更好的约束性，但也导致数据关系表增加而令数据库IO更易繁忙，原来交由数据库处理的关系约束现更多在数据库使用程序中完成。

## 第一范式

定义：数据库中的所有**字段**（列）都是单一属性，**不可再分**的。这个单一属性由基本的数据类型所构成，如整型、浮点型、字符串等。第一范式是为了保证列的原子性。

## 第二范式

定义：数据库中的表不存在非关键字段对任一关键字字段的**部分函数**依赖**部分函数依赖**是指存在着**组合关键字**中的某一关键字决定非关键字的情况 第二范式在满足了第一范式的基础上，消除非主键列对**联合主键**的部分依赖

## 第三范式

定义：所有非主键属性都只和候选键有相关性，也就是说非主键属性之间应该是独立无关的。第三范式是在满足了第二范式的基础上，消除列与列之间的**传递依赖**。

数据库设计时，遵循范式和反范式一直以来是一个颇受争议的问题。遵循范式对数据关系更好的约束性，并且减少数据冗余，可以更好地保证数据一致性。而反范式则是为了获得更好地性能。所以范式还是反范式并没有明确的标准，适合自己业务场景的才是最好的。

# 横表纵表

# 横表

CREATE TABLE `table\_h2z` (

`name` varchar(32) DEFAULT NULL,

`chinese` int(11) DEFAULT NULL,

`math` int(11) DEFAULT NULL,

`english` int(11) DEFAULT NULL

) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8;

/\*Data for the table `table\_h2z` \*/

insert into `table\_h2z`(`name`,`chinese`,`math`,`english`) values

('mike',45,43,87),

('lily',53,64,88),

('lucy',57,75,75);

# 纵表

CREATE TABLE `table\_z2h` (

`name` varchar(32) DEFAULT NULL,

`subject` varchar(8) NOT NULL DEFAULT '',

`score` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

/\*Data for the table `table\_z2h` \*/

insert into `table\_z2h`(`name`,`subject`,`score`) values

('mike','chinese',45),

('lily','chinese',53),

('lucy','chinese',57),

('mike','math',43),

('lily','math',64),

('lucy','math',75),

('mike','english',87),

('lily','english',88),

('lucy','english',75);

## 横表转纵表

SELECT NAME, 'chinese' AS `subject`, chinese AS `score` FROM table\_h2z

UNION ALL

SELECT NAME, 'math' AS `subject`, math AS `score` FROM table\_h2z

UNION ALL

SELECT NAME, 'english' AS `subject`, english AS `score` FROM table\_h2z

执行结果

+------+---------+-------+

| name | subject | score |

+------+---------+-------+

| mike | chinese | 45 |

| lily | chinese | 53 |

| lucy | chinese | 57 |

| mike | math | 43 |

| lily | math | 64 |

| lucy | math | 75 |

| mike | english | 87 |

| lily | english | 88 |

| lucy | english | 75 |

+------+---------+-------+

9 rows in set (0.00 sec)

## 纵表转横表

SELECT NAME,

SUM(CASE `subject` WHEN 'chinese' THEN score ELSE 0 END) AS chinese,

SUM(CASE `subject` WHEN 'math' THEN score ELSE 0 END) AS math,

SUM(CASE `subject` WHEN 'english' THEN score ELSE 0 END) AS english

FROM table\_z2h

GROUP BY NAME

执行结果

+------+---------+------+---------+

| name | chinese | math | english |

+------+---------+------+---------+

| lily | 53 | 64 | 88 |

| lucy | 57 | 75 | 75 |

| mike | 45 | 43 | 87 |

+------+---------+------+---------+

3 rows in set (0.00 sec)