

# SQL学习笔记

## • 目标

- 熟悉数据库的增删改查：select数据提取，SQL书写规则，where语句的使用（like, between, in/or、逻辑判断），表的内联外联，分组，子查询，排序和去重，聚合函数

## • 什么是资料库

- SQL=structured query language
- 与关联式资料库管理系统做沟通的语言  
eg.youtube --- (sql) -----关联式资料库管理系统 (mysql) -----关联式资料库
- 软件：**MYSQL**、oracle、postgreSQL、SQL Serve

## • tables and keys

- primary key（主键）：设为主键的属性可以唯一的表示一条记录。  
eg.1号同学（作为主键）名叫小白，主修历史；主键不可抽重复
  - 属性一般为表格的列
  - 可以设定多个：表示多个属性合起来可以唯一的表示一条记录
- foreign key外键  
在一个表格中插入的另一个表格的主键

## • 创建资料库

•



## • 创建表格

- 数据类型：
  - **INT** 整数
  - **DECIMAL(m,n)** 有小数点的数  
m表示整数位和小数位总共的位数，n表示小数的位数。eg. 2.33对应 (3,2)
  - **VARCHAR(n)** 字串（纯文字）  
n表示存放的字符数量
  - **BLOB**（binary large object）图片、视频和档案
  - **DATE** 日'YYYY-MM-DD'  
eg.2022-08-08
  - **TIMESTAMP** 记录时间'YYY-MM-DD HH:MM:SS'
- 举例

```
Query 1 x
Limit to 1000 rows

1 • CREATE DATABASE `sql_tutorial`;
2 • SHOW DATABASES;
3 • USE `sql_tutorial`;
4 • CREATE TABLE `student` (
5   `student_id` INT PRIMARY KEY,
6   `name` VARCHAR(20),
7   `major` VARCHAR(20)
8 );
9
10 • describe `student`;
11 • DROP TABLE `student`;
12
13 • ALTER TABLE `student` ADD `gpa` DECIMAL(3,2);
14 • ALTER TABLE `student` DROP COLUMN `gpa`;
```

创建表格

主键的另一种写法，列举属性后，在括号新增primary key(`student\_id`)

展示表格用describe，展示数据库用show

## • 导入数据

### • 举例

```
• SELECT * FROM `student`;
• INSERT INTO `student` VALUES(1,'小白','历史');
• INSERT INTO `student` VALUES(2,'小黑','生物');
• INSERT INTO `student` VALUES(3,'小绿',NULL);
• INSERT INTO `student` (`name`,`major`,`student_id`) VALUES('小蓝','英语',4);
• INSERT INTO `student` (`major`,`student_id`) VALUES('英语',5);
```

注意反引号和单双引号的使用区别

非数值型数据需要用单引号/双引号，如这里的'小白'

## • 限制、约束 (constraint)

### • 常见限制语句

- **DEFAULT'xx'** :xx表示预设值（没有输入的数据的情况下的默认值）
- **AUTO\_INCREMENT**:序号自动加1（只适用于整数型数据列）
- **NOT NULL**:非空值
- **UNIQUE**: 值唯一

### • 举例

```
• CREATE TABLE `student` (
  `student_id` INT PRIMARY KEY,
  `name` VARCHAR(20) NOT NULL,
  `major` VARCHAR(20) UNIQUE
);
```

## • 修改和删除数据 (update & delete)

- 修改语句: **UPDATE** `表格名` **SET** '属性名称'='设定值' **WHERE** 属性名称='已有值'

```

42 • SET SQL_SAFE_UPDATES = 0;
43
44 • UPDATE `student`
45   SET `major`='英语文学'
46   WHERE `major`='英语';
47   -- 将student表格里major是英语的都替换为英语文学
48
49 • UPDATE `student`
50   SET `major`='生物'
51   WHERE `student_id`=3;
52   -- 将第三列的专业替换为生物, 如果没有WHERE则表格所有major均变为物理
53
54 • UPDATE `student`
55   SET `major`='生化'
56   WHERE `major`='生物' OR '化学' ;
57
58 • UPDATE `student`
59   SET `name`='小灰', `major`='物理'
60   WHERE `student_id`=1;
61   -- 可同时修改一条数据中的多个值, 用逗号隔开

```

56行有误, 应为WHERE `major`='生物'OR `major`='化学'

- 注意区分 **ALTER TABLE** '表格名' **ADD**
- 删除语句: **DELETE FROM** `表格名` **WHERE** '限定条件'

```

• DELETE FROM `student`
  WHERE `name`='小灰' AND `major`='物理';

```

	student_id	name	major	score
*	NULL	NULL	NULL	NULL

Tips: ①<>表示不等于; ②不加限定会把表格中的数据均删除 (如图)

## • 查询数据 **select**

- **SELECT DISTINCT** '属性' **FROM** `表格名` **WHERE** **ORDER BY** **LIMIT**

- 属性可以是多个, 用逗号链接

```

• SELECT `name`,`major` FROM `student`;

```

- **ORDER BY** 默认升序 (ASC), 如需降序加上**DESC**; 可以根据多个条件排序, 用逗号隔开; 如每个条件均需降序排序, 则每个条件后均需加上DESC

```

SELECT *
FROM `student`
ORDER BY `score` DESC;

```

```

• SELECT *
  FROM `student`
  ORDER BY `score`,`student_id`;

```

- **LIMIT**后面跟数字，表示只取前几条数据，可以与ORDER BY混用，LIMIT必须在WHERE之后。
  - LIMIT 1,3表示跳过1条，取3条
  - limit 与 offset连用：limit后面只能有一个参数，表示要取的的数量,offset表示要跳过的数量
- **WHERE, IN**的用法：WHERE `major`='英语'OR`major`='化学' OR `major`='历史' 等价于 WHERE `major` IN('英语','化学','历史')  
 不等于的表示：<> ,!=      区间表示 between and
- **DISTINCT** 去重,覆盖所有被查询项
- **select 子句顺序**：select...from ..where..group by...having...order by...

## • 分组group by

- 如果select语句中使用表达式，则group by子句中也必须使用表达式，不能使用别名
- **having**和**where**的区别：having过滤组，where在分组前过滤行；having后可以用聚合函数，where不可以
- select 后查询多个对象且包含聚合函数的情况下，需要使用group by 进行分组

## • 函数

### • 聚合函数 aggregate function

- **COUNT**计数、**AVERAGE**平均、**SUM**总和、**MAX/MIN**最大/最小

```
-- 取所有出生于1970后的女性员工人数
SELECT COUNT(*)
FROM `employee`
WHERE `sex`='F' and `birthdate`>'1970-01-01'
```

```
-- 所有员工的平均薪水
SELECT AVERAGE(`salary`)
FROM `employee`
```

```
-- 所有员工的薪水总和
SELECT SUM(`salary`)
FROM `employee`;
```

```
-- 最高薪水MAX, 最低MIN
SELECT MAX(`salary`)
FROM `employee`;
```

### • 其他函数

- **round(A, x)**将A保留x位小数
- 条件函数：**case when**

题目：现在运营想要将用户划分为25岁以下和25岁及以上两个年龄段，分别查看这两个年龄段用户数量

```
1 SELECT CASE WHEN age < 25 OR age IS NULL THEN '25岁以下'
2           WHEN age >= 25 THEN '25岁及以上'
3           END age_cut, COUNT(*) number
4 FROM user_profile
5 GROUP BY age_cut
```

```
1 CASE 测试表达式
2 WHEN 简单表达式1 THEN 结果表达式1
3 WHEN 简单表达式2 THEN 结果表达式2 ...
4 WHEN 简单表达式n THEN 结果表达式n
5 [ ELSE 结果表达式n+1 ]
6 END
```

是一种多分支的函数，可以根据条件列表的值返回多个可能的结果表达式中的一个。可用在任何允许使用表达式的地方，但不能单独作为一个语句执行。

- 日期函数

- **day(), month(), year(), date\_add(date, interval数字), 日期变量date的后x (数字) 天**

- 文本函数

- 字符串截取函数：**substring\_index(变量名称, '分隔符', 数字)**

[【MySQL】字符串截取之substring\\_index](#)

**substring\_index(str, delim, count)**

**str:**要处理的字符串

**delim:**分隔符

**count:**计数

例子：str=www.wikibt.com

substring\_index(str, '.', 1)

结果是：www

substring\_index(str, '.', 2)

结果是：www.wikibt

也就是说，如果count是正数，那么就是从左往右数，第N个分隔符的左边的全部内容相反，如果是负数，那么就是从右边开始数，第N个分隔符右边的所有内容，如：

substring\_index(str, '.', -2)

结果为：wikibt.com

有人会问，如果我要中间的wikibt怎么办？

很简单的，两个方向：

从右数第二个分隔符的右边全部，再从左数的第一个分隔符的左边：

substring\_index(substring\_index(str, '.', -2), '.', 1);

👉 牛客@细雨喃喃

- 长度**length ()**：返回文本字段中的值的长度
- 连接**concat ()**：用于将两个或多个字符串连接起来，形成一个单一的字符串
- 窗口函数：**row\_number() over (partition by col1 order by col2)**
  - 函数的含义为先分组再排序,注意最终输出的是排序的序号

表: user\_profile

id	device_id	gender	age	university	gpa
1	2138	male	21	北京大学	3.1
2	3214	male	25	复旦大学	3.5
3	6543	female	18	北京大学	3.2
4	3111	male	21	复旦大学	3.3

代码:

```
Select device_id, university,gpa,
row_number() over (partition by university order by gpa desc) as
rank,--desc代表降序排列
From user_profile
```

上述代码含义为在每个学校的内部根据gpa进行一次排名,获得每个学生在学校的名次数据, desc代表是按照从大到小降序排列。

## 输出

device_id	university	Gpa	rank
6543	北京大学	3.2	1
2138	北京大学	3.1	2
3214	复旦大学	3.5	1
3111	复旦大学	3.3	2

## • 通配符 Wildcard

- %匹配0个或多个字符,不能表示NULL, 可用于任何位置, 同时使用多个
- \_匹配任意一个字符

```
-- 取得电话位数是335的客户 (like表示模糊查询, 不能用=代替)
• SELECT * FROM `client`
  WHERE `phone` like '%335' ;

-- 取12月生日的员工, 注意这里是5个下划线, 不能用%12%, 这个写法会包括2012-12-12
• SELECT * FROM `employee`
  WHERE `birthdate` like '_____12%';
```

- []: 匹配[]中的任意一个字符(若要比对的字符是连续的, 则可以用连字符“-”表达);
- [^]: 不匹配[]中的任意一个字符
- 通配符前用like 不用=

## • 并集union

- 用union连接多个语句: ①合并前后属性数量一致; ②各属性的数据类型一致;结果自动去重

```
SELECT `name` AS `total_name`, `emp_id` AS `total_id` FROM `employee`
UNION
SELECT `client_name`, `client_id` FROM `client`;
```

如果没有命名, 则以select第一句中的名称作为列名称

- union all 返回所有匹配行, 不去重

## • 连接join

- 内连接: 语句: **select .. from '表格1' join '表格2' on `属性名1`= `属性名2`**

```

208 • select `emp_id`,`name`,`branch_name`
209 from `employee`
210 join `branch`
211 on `employee`.`emp_id`=`manager_id`;

```

emp_id	name	branch_name
206	小黄	研发
207	小绿	行政
208	小黑	资讯

- 1、属性前可以加上表格名，例如`employee`.`emp\_id`可以区分属性来自哪个表格，避免连接的两个表格有相同的属性名称造成的混淆
- 2、多个表格连接，注意顺序，先连接外层再连接内层
- 3、只连接两个表格匹配的项目

## • LEFT JOIN

```

214 • select `emp_id`,`name`,`branch_name`
215 from `employee`
216 left join `branch`
217 on `employee`.`emp_id`=`manager_id`;

```

emp_id	name	branch_name
206	小黄	研发
207	小绿	行政
208	小黑	资讯
209	小白	NULL
210	小兰	NULL

以左表格的数据为基准

## • RIGHT JOIN

```

214 • select `emp_id`,`name`,`branch_name`
215 from `employee`
216 right join `branch`
217 on `employee`.`emp_id`=`manager_id`;

```

emp_id	name	branch_name
206	小黄	研发
207	小绿	行政
208	小黑	资讯
NULL	NULL	摸鱼

以右表格数据为基准

## • 子查询 subquery

- 查询子条件为单一行用=

```

219 -- 找出研发部门经理名字
220 • select `name`
221    from `employee`
222   where `emp_id`=(
223       select `manager_id`
224       from `branch`
225       where `branch_name`='研发'
226   );

```

Result Grid		Filter Rows:	Export
	name		
▶	小黄		

- 查询子条件为多行用in

```

232 • select `name`
233    from `employee`
234   where `emp_id` in(
235       select `emp_id`
236       from `works_with`
237       where `total_sales`>50000
238   );

```

Result Grid		Filter Rows:	Export
	name		
▶	小黄		
	小兰		

- 作为子查询的select语句只能查询单个列

## • ON DELETE

- set null表示引用源删除后，引用处显示null
- cascade表示引用源删除后，引用处所在的数据行均删除

## • 创建公司表格

- 原表



## Employee

emp_id	name	birth_date	sex	salary	branch_id	sup_id
206	小黃	1998-10-08	F	50000	1	NULL
207	小綠	1985-09-16	M	29000	2	206
208	小黑	2000-12-19	M	35000	3	206
209	小白	1997-01-22	F	39000	3	207
210	小蘭	1925-11-10	F	84000	1	207

## Branch

branch_id	branch_name	manager_id
1	研發	206
2	行政	207
3	資訊	208

## Client

client_id	client_name	phone
400	阿狗	254354335
401	阿貓	25633899
402	旺來	45354345
403	露西	54354365
404	艾瑞克	18783783

## Works\_With

emp_id	client_id	total_sales
206	400	70000
207	401	24000
208	400	9800
208	403	24000
210	404	87940

Primary Key
Foreign Key
Attribute

### SQL表格创建思路

- ①正常创建employee的表格，不考虑foreign key

```
CREATE TABLE `employee`(  
  `emp_id` INT PRIMARY KEY,  
  `name` VARCHAR(20),  
  `birthdate` DATE,  
  `sex` VARCHAR(20),  
  `salary` INT,  
  `branch_id` INT,  
  `sup_id` INT);
```

- ②创建branch表格，纳入以表格employee为引用源的foreign key

```
CREATE TABLE `branch`(  
  `branch_id` INT PRIMARY KEY,  
  `branch_name` VARCHAR(20),  
  `manager_id` INT,  
  FOREIGN KEY(`manager_id`) REFERENCES `employee`(`emp_id`) ON DELETE SET NULL  
);
```

- ③在表格employee中增加以本身和表格branch为引用源的foreign key

```
ALTER TABLE `employee`  
  ADD FOREIGN KEY(branch_id)  
  REFERENCES `branch`(`branch_id`)  
  ON DELETE SET NULL;  
  
ALTER TABLE `employee`  
  ADD FOREIGN KEY(`sup_id`)  
  REFERENCES `employee`(`emp_id`)  
  ON DELETE SET NULL;
```

- ④正常创建表格client

- ```
CREATE TABLE `client`(  
  `client_id` INT PRIMARY KEY,  
  `client_name` VARCHAR(20),  
  `phone` varchar(20)  
);
```

- ⑤创建表格works\_with，并添加两个foreign key

- ```
CREATE TABLE `works_with`(  
  `emp_id` INT,  
  `client_id` INT,  
  `total_sales` INT,  
  PRIMARY KEY(`emp_id`, `client_id`),  
  FOREIGN KEY(`emp_id`) REFERENCES `employee`(`emp_id`) ON DELETE CASCADE,  
  FOREIGN KEY(`client_id`) REFERENCES `client`(`client_id`) ON DELETE CASCADE  
);
```

注：这里的emp\_id和client\_id既是主键又是外键，所以在设置外键时，不能写set null，而是写cascade

以上内容整理于 [幕布文档](#)