SQL知识总结

顺序

- 写作顺序：select-forom-where-group by-having-order by
- 执行顺序：from-where-group by-having-order by
- 筛选行-分组-筛选分组结果-排序

SQL基础

SELECT - 从数据库中提取数据
UPDATE - 更新数据库中的数据
DELETE - 从数据库中删除数据
INSERT INTO - 向数据库中插入新数据
CREATE DATABASE - 创建新数据库
ALTER DATABASE - 修改数据库
CREATE TABLE - 创建新表
ALTER TABLE - 变更（改变）数据库表

DROP TABLE - 删除表
CREATE INDEX - 创建索引（搜索键）
DROP INDEX - 删除索引

常用操作

1. 选择

```
SELECT name,country FROM Websites;
```

2. 去重

```
SELECT DISTINCT country FROM Websites;
```

3. 条件过滤

```
SELECT * FROM Websites WHERE country='CN';
```

```
=   等于
<>  不等于!=
>   大于
<   小于
>=  大于等于
<=  小于等于
BETWEEN 在某个范围内
LIKE    搜索某种模式
IN  指定针对某个列的多个可能值
```

4. AND & OR

```
SELECT * FROM Websites
WHERE alexa > 15
AND (country='CN' OR country='USA');
```

5. 排序

```
SELECT * FROM Websites
ORDER BY alexa DESC;
```

6. 插入

```
INSERT INTO Websites (name, url, country)
VALUES ('stackoverflow', 'http://stackoverflow.com/', 'IND');
```

7. 更新

```
UPDATE Websites
SET alexa='5000', country='USA'
WHERE name='菜鸟教程';
```

8. 删除表中的行

```
DELETE FROM Websites
WHERE name='百度' AND country='CN';
```

高级操作

1. 返回的记录数目 **SELECT TOP, LIMIT, ROWNUM**

```
SELECT TOP number|percent column_name(s)
FROM table_name;
```

```
SELECT TOP 50 PERCENT * FROM Websites;
```

2. 搜索列中的指定模式

```
SELECT * FROM Websites
WHERE name LIKE '%k';
```

```
SELECT * FROM Websites
WHERE name NOT LIKE '%oo%';
```

3. 通配符

```
%            替代 0 个或多个字符
_            替代一个字符
[charlist]   字符列中的任何单一字符
[^charlist]
或[!charlist] 不在字符列中的任何单一字符
```

4. 规定多个值

```
SELECT * FROM Websites
WHERE name IN ('Google','菜鸟教程');
```

5. 选取介于两个值之间

```
SELECT * FROM Websites
WHERE alexa NOT BETWEEN 1 AND 20;
```

6. 别名

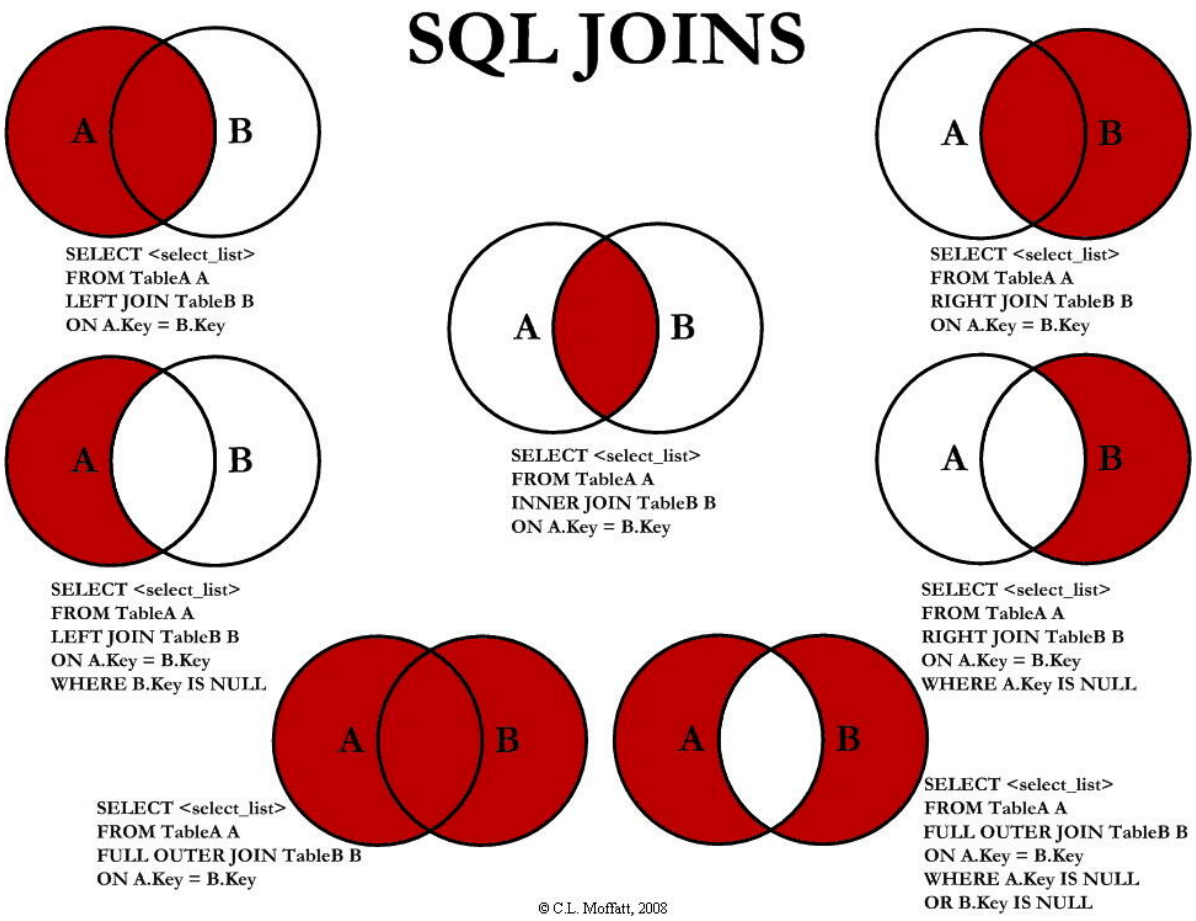
```
SELECT name, CONCAT(url, ', ', alexa, ', ', country) AS site_info
FROM Websites;
```

```
SELECT w.name, w.url, a.count, a.date
FROM Websites AS w, access_log AS a
WHERE a.site_id=w.id and w.name="菜鸟教程";
```

在下面的情况下，使用别名很有用：

- 在查询中涉及超过一个表
- 在查询中使用了函数
- 列名称很长或者可读性差
- 需要把两个列或者多个列结合在一起

7. 两个或多个表的行连接**JOIN ON**




```
SELECT Websites.id, Websites.name, access_log.count, access_log.date
FROM Websites
INNER JOIN access_log
ON Websites.id=access_log.site_id;
```

- INNER JOIN：如果表中有至少一个匹配，则返回行
- LEFT JOIN：即使右表中没有匹配，也从左表返回所有的行
- RIGHT JOIN：即使左表中没有匹配，也从右表返回所有的行
- FULL JOIN：只要其中一个表中存在匹配，则返回行

8. 合并两个或多个SELECT语句**UNION**
   请注意，UNION内部的每个SELECT语句必须拥有相同数量的列。列也必须拥有相似的据类型。同时，每个SELECT语句中的列的顺序必须相同。

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

> **注释**：默认地，UNION 操作符选取不同的值。如果允许重复的值，请使用 **UNION ALL**。

```
SELECT country, name FROM Websites
WHERE country='CN'
UNION ALL
SELECT country, app_name FROM apps
WHERE country='CN'
ORDER BY country;
```

9. 一个表复制数据信息到另一个表

```
SELECT *
INTO WebsitesBackup2016
FROM Websites;
```

> select into from 和 insert into select 都是用来复制表

> 两者的主要区别为： select into from 要求目标表不存在，因为在插入时会自动创建； insert into select from 要求目标表存在。

10. 创建数据库

```
CREATE DATABASE my_db;
```

```
CREATE TABLE Persons
(
PersonID int,
LastName varchar(255),
FirstName varchar(255),
Address varchar(255),
City varchar(255)
);
```

11. 约束

- NOT NULL - 指示某列不能存储 NULL 值。
- UNIQUE - 保证某列的每行必须有唯一的值。
- PRIMARY KEY - NOT NULL 和 UNIQUE 的结合。确保某列（或两个列多个列的结合）有唯一标识，有助于更容易更快速地找到表中的一个特定的记录。
- FOREIGN KEY - 保证一个表中的数据匹配另一个表中的值的参照完整性。
- CHECK - 保证列中的值符合指定的条件。
- DEFAULT - 规定没有给列赋值时的默认值。

> UNIQUE 约束唯一标识数据库表中的每条记录。
>
> UNIQUE 和 PRIMARY KEY 约束均为列或列集合提供了唯一性的保证。
>
> PRIMARY KEY 约束拥有自动定义的 UNIQUE 约束。

> 请注意，每个表可以有多个 UNIQUE 约束，但是每个表只能有一个 PRIMARY KEY 约束。

12. 索引

- 简单索引

```
CREATE INDEX index_name
ON table_name (column_name)
```

- 不重复索引

```
CREATE UNIQUE INDEX index_name
ON table_name (column_name)
```

```
CREATE INDEX PIndex
ON Persons (LastName)
```

数据越离散，索引越有用

索引类型

- 普通索引：仅加速查询
- 唯一索引：加速查询 + 列值唯一（可以有null）
- 主键索引：加速查询 + 列值唯一（不可以有null）+ 表中只有一个
- 组合索引：多列值组成一个索引，专门用于组合搜索，其效率大于索引合并
- 全文索引：对文本的内容进行分词，进行搜索

13. 撤销

- DROP INDEX 语句用于删除表中的索引
- DROP TABLE 语句用于删除表。
- DROP DATABASE 语句用于删除数据库。

14. 在已有的表中添加、删除或修改列

```
ALTER TABLE table_name
ADD column_name datatype

ALTER TABLE table_name
DROP COLUMN column_name

ALTER TABLE table_name
MODIFY COLUMN column_name datatype
```

15. 新记录插入表中时生成一个唯一的数字

```
CREATE TABLE Persons
(
ID int IDENTITY(1,1) PRIMARY KEY,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)
```

16. 视图

视图是基于 SQL 语句的结果集的可视化的表

```
CREATE VIEW view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition
```

注释：视图总是显示最新的数据！每当用户查询视图时，数据库引擎通过使用视图的 SQL 语句重建数据。

> **视图的作用**：
>
> 1、视图隐藏了底层的表结构，简化了数据访问操作，客户端不再需要知道底层表的结构及其之间的关系。
>
> 2、视图提供了一个统一访问数据的接口。（即可以允许用户通过视图访问数据的安全机制，而不授予用户直接访问底层表的权限）
>
> 3、从而加强了安全性，使用户只能看到视图所显示的数据。
>
> 4、视图还可以被嵌套，一个视图中可以嵌套另一个视图。

17. 日期

MYSQL

```
NOW()    返回当前的日期和时间
CURDATE()    返回当前的日期
CURTIME()    返回当前的时间
DATE()  提取日期或日期/时间表达式的日期部分
EXTRACT()    返回日期/时间的单独部分
DATE_ADD()  向日期添加指定的时间间隔
DATE_SUB()  从日期减去指定的时间间隔
DATEDIFF()  返回两个日期之间的天数
DATE_FORMAT()    用不同的格式显示日期/时间
```

SQL SEVER

```
GETDATE()    返回当前的日期和时间
DATEPART()   返回日期/时间的单独部分
DATEADD()    在日期中添加或减去指定的时间间隔
DATEDIFF()   返回两个日期之间的时间
CONVERT()    用不同的格式显示日期/时间
```

数据类型

- DATE - 格式：YYYY-MM-DD
- DATETIME - 格式：YYYY-MM-DD HH:MM:SS
- TIMESTAMP - 格式：YYYY-MM-DD HH:MM:SS
- YEAR - 格式：YYYY 或 YY

提示：如果您希望使查询简单且更易维护，那么请不要在日期中使用时间部分！

取日期部分：使用Convert()函数

```
select convert(char(10),GetDate(),120) as Date
```

> 第3个参数就是用来设置日期类型数据的显示样式的

```
SELECT CONVERT(varchar(100), GETDATE(), 5) 09-05-11
```

18. 空缺值null

```
SELECT LastName,FirstName,Address FROM Persons
WHERE Address IS NOT NULL
```

```
--如果alexa列为null值，则赋予0，否则，取原值
select id,name,url,ifnull(alexa,0)from websites;
select id,name,url,COALESCE(alexa,0) from websites;
```

19. SQL 通用数据类型

```
CHARACTER(n)    字符/字符串。固定长度 n。
VARCHAR(n) 或
CHARACTER VARYING(n)    字符/字符串。可变长度。最大长度 n。
BINARY(n)   二进制串。固定长度 n。
BOOLEAN 存储 TRUE 或 FALSE 值
VARBINARY(n) 或
BINARY VARYING(n)   二进制串。可变长度。最大长度 n。
INTEGER(p)  整数值（没有小数点）。精度 p。
SMALLINT    整数值（没有小数点）。精度 5。
INTEGER 整数值（没有小数点）。精度 10。
```

```
BIGINT  整数值（没有小数点）。精度 19。
DECIMAL(p,s)    精确数值，精度 p，小数点后位数 s。例如：decimal(5,2) 是一个小数点前有 3 位数，小数点后有 2 位数的数字。
NUMERIC(p,s)    精确数值，精度 p，小数点后位数 s。（与 DECIMAL 相同）
FLOAT(p)    近似数值，尾数精度 p。一个采用以 10 为基数的指数计数法的浮点数。该类型的 size 参数由一个指定最小精度的单一数字组成。
REAL    近似数值，尾数精度 7。
FLOAT   近似数值，尾数精度 16。
DOUBLE PRECISION    近似数值，尾数精度 16。
DATE    存储年、月、日的值。
TIME    存储小时、分、秒的值。
TIMESTAMP   存储年、月、日、小时、分、秒的值。
INTERVAL    由一些整数字段组成，代表一段时间，取决于区间的类型。
```

函数

1. 有用的 Aggregate 函数：

- AVG() - 返回平均值
- COUNT() - 返回行数
- FIRST() - 返回第一个记录的值
- LAST() - 返回最后一个记录的值
- MAX() - 返回最大值
- MIN() - 返回最小值
- SUM() - 返回总和

2. 有用的 Scalar 函数：

- UCASE() - 将某个字段转换为大写
- LCASE() - 将某个字段转换为小写
- MID() - 从某个文本字段提取字符，MySql 中使用
- SubString(字段，1，end) - 从某个文本字段提取字符
- LEN() - 返回某个文本字段的长度
- ROUND() - 对某个数值字段进行指定小数位数的四舍五入
- NOW() - 返回当前的系统日期和时间
- FORMAT() - 格式化某个字段的显示方式

注意：使用函数可以不用group by,使用group by 必须接函数。使用函数返回单一值。

3. 指定列不同计数

```
SELECT COUNT(DISTINCT column_name) FROM table_name;
```

4. 聚合函数

```
SELECT site_id, SUM(access_log.count) AS nums
FROM access_log GROUP BY site_id;
```

```
SELECT Websites.name,COUNT(access_log.aid) AS nums FROM access_log
LEFT JOIN Websites
ON access_log.site_id=Websites.id
GROUP BY Websites.name;
```

HAVING 子句可以让我们筛选分组后的各组数据。