MySQL详解

一、MySQL简介

MySQL是一个轻量级关系型数据库管理系统，由瑞典MySQL AB公司开发，目前属于Oracle公司。目前MySQL被广泛地应用在Internet上的中小型网站中，由于体积小、速度快、总体拥有成本低，开放源码、免费，一般中小型网站的开发都选择Linux + MySQL作为网站数据库。MySQL是一个关系型数据库管理系统，MySQL是一种关联数据库管理系统，关联数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，就增加了速度并提高了灵活性。由于分布式和集群的出现，mysql也用于大型的网站上。

MySQL官方文档

本文章参考博客

二、MySQL学习前言

学习流程：操作数据库 ->操作数据库中的表 ->操作数据库中表中的数据

sql语言分类

名称	解释	命令
DDL	定义和管理数据对象，如：数据库，数据表等	create,drop,alter
DML	用于操作数据库对象所包含的数据	insert,delete,update
DQL	用于查询数据库对象所包含的数据	select
DCL	用于管理数据库，包括管理权限和数据更改	grant,commit,rollback

三、数据库操作（DDL）

1、数据库操作

1.1、链接数据库

mysql -u root -p --链接本机mysql：进入目录mysql\bin，再键入命令mysql -u root -p，回车后提示你输密码.注意用户名前可以有空格也可以没有空格，但是密码前必须没有空格，否则让你重新输入密码。

update user set password=password('123456')where user='root'; --修改密码

flush privileges; --刷新数据库

show databases; --显示所有数据库

use dbname；--打开某个数据库

show tables; --显示数据库mysql中所有的表

describe user; --显示表mysql数据库中user表的列信息

create database name; --创建数据库

use databasename; --选择数据库

exit; --退出Mysql

? --命令关键词 : 寻求帮助

-- 表示注释

1.2、操作数据库

（1）创建数据库

```
CREATE DATABASE [IF NOT EXISTS] test;
```

1

（2）删除数据库

```
DROP DATABASE [IF EXISTS] test;
```

1

（3）查询数据库

```
SHOW DATABASES;
```

1

（4）使用数据库

```
USE mysql;
```

1

2、对数据库中表的操作

2.1、数据库中的表的列类型（列的数据类型）

列类型 : 规定数据库中该列存放的数据类型

（1）、数值类型

类型	说明	取值范围	存储需求
tinyint	非常小的数据	有符值：-2^7 ~ 2^7-1 无符号值：0 ~ 2^8-1	1字节
smallint	较小的数据	有符值：-2^{15} ~ $2^{15}-1$ 无符号值：0 ~ $2^{16}-1$	2字节
mediumint	中等大小的数据	有符值：-2^{23} ~ $2^{23}-1$ 无符号值：0 ~ $2^{24}-1$	3字节
int	标准整数	有符值：-2^{31} ~ $2^{31}-1$ 无符号值：0 ~ $2^{32}-1$	4字节
bigint	较大的整数	有符值：-2^{63} ~$2^{63}-1$ 无符号值：0 ~$2^{64}-1$	8字节
float	单精度浮点数	±1.1754351e -38	4字节
double	双精度浮点数	±2.2250738585072014e -308	8字节
decimal	字符串形式的浮点数	decimal(m, d)	个字节

（2）、字符串类型

类型	说明	最大长度
char[(M)]	固定长字符串，检索快但费空间，0 <= M <= 255	M字符
varchar[(M)]	可变字符串 0 <= M <= 65535	变长度
tinytext	微型文本串	2^8 - 1字节
text	文本串	2^{16} - 1字节

（3）、日期和时间型数值类型

类型	说明	取值范围
DATE	YYYY-MM-DD，日期格式	1000-01-01~ 9999-12-31
TIME	Hh:mm:ss ，时间格式	-838:59:59~838:59:59
DATETIME	YY-MM-DD hh:mm:ss	1000-01-01 00:00:00 至 9999-12-31 23:59:59
TIMESTAMP	YYYYMMDDhhmmss格式表示的时间戳	197010101000000 ~2037年的某个时刻
YEAR	YYYY格式的年份值	1901~2155

（4）、null类型

理解为 “没有值” 或 “未知值”

不能用NULL进行算术运算 , 结果仍为NULL

2.2、字段属性(列的属性)

属性名	特点
UnSigned	**无符号**；声明后该数据列不允许负数
ZEROFILL	**0填充**；声明后该数据列不足位数的用0来填充，如int(3),5则为005**
Auto_InCrement	**自动增长**，声明后每添加一条数据，自动在上一个记录数上加 1(默认)； 通常用于设置**主键**，且为整数类型； 可定义起始值和步长
NULL 和 NOT NULL	**MySQL默认为NULL**，即没有插入该列的数值；如果设置为NOT NULL，则该列必须有值
DEFAULT	**默认值**；声明后用于设置默认值； 例如,性别字段,默认为"男"，否则为 "女"；若无指定该列的值，则默认值为"男"的值

注意： 当前表设置步长(AUTO_INCREMENT=100) : 只影响当前表；

而SET @@auto_increment_increment=5 ; 影响所有使用自增的表(全局)

1

2

2.3、创建数据库中的表

```
-- 目标 : 创建一个school数据库
-- 创建学生表(列,字段)
-- 学号int 登录密码varchar(20) 姓名,性别varchar(2),出生日期(datatime),家庭住址,email
-- 创建表之前 , 一定要先选择数据库
CREATE TABLE IF NOT EXISTS student (
id int(4) NOT NULL AUTO_INCREMENT COMMENT '学号',
name varchar(30) NOT NULL DEFAULT '匿名' COMMENT '姓名',
pwd varchar(20) NOT NULL DEFAULT '123456' COMMENT '密码',
sex varchar(2) NOT NULL DEFAULT '男' COMMENT '性别',
birthday datetime DEFAULT NULL COMMENT '生日',
address varchar(100) DEFAULT NULL COMMENT '地址',
email varchar(50) DEFAULT NULL COMMENT '邮箱',
PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
CREATE TABLE '表名' (
'字段名' 列类型 [属性] [索引] [注释],
'字段名' 列类型 [属性] [索引] [注释],
'字段名' 列类型 [属性] [索引] [注释]
)[表类型][字符集设置]
SHOW CREATE DATABASE student; -- 显示建数据库语句
SHOW CREATE TABLE student; -- 显示建表语句
//自定义练习
```

```
CREATE TABLE students (
id INT NOT NULL AUTO_INCREMENT COMMENT '学号',
name VARCHAR(10) NOT NULL DEFAULT '王老六' COMMENT '姓名',
classname INT NOT NULL DEFAULT '1905121' COMMENT '班级',
PRIMARY KEY (id,name) --主键可以为多个字段
) ENGINE=INNODB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8
```

引擎名称	MyISAM	InnoDB
事务支持（ACID）	不支持	支持
行级锁	不支持	支持
外键约束	不支持	支持
全文索引	支持	不支持
空间大小	比较小	约为MyISAM的两倍

经验（适用场合）：

适用 MyISAM：节约空间及相应速度

适用 InnoDB：安全性，事务处理及多用户操作数据表

2.4、修改和删除数据库中的表

（1）修改表

```
ALTER TABLE student RENAME AS teacher; -- 修改表名
alter table 旧表名 rename as 新表名
ALTER TABLE teacher ADD sex INT(11); -- 添加字段
alter table 表名 add 字段名 字段属性
ALTER TABLE teacher MODIFY sex VARCHAR(11); -- 修改字段属性
alter table 表名 modify 字段名 字段属性
ALTER TABLE teacher CHANGE sex sex1 VARCHAR(12);-- 修改字段名和属性
alter table 表名 change 旧字段名 新字段名 字段属性[]
ALTER TABLE teacher DROP sex1; -- 删除表的字段
alter table 表名 drop 字段名
```

（2）删除表

```
DROP TABLE teacher; --删除表
drop table if exists 表名
```

四、MySQL数据管理（DML）

1、主键和外键

1.1、主键

简介：

主关键字是一种唯一关键字，表定义的一部分。一个表的主键可以由多个关键字共同组成，并且主关键字的列不能包含空值。主关键字是可选的，并且可在 CREATE TABLE 或 ALTER TABLE 语句中定义。即：

主关键字(primary key)是表中的一个或多个字段

主键的值用于唯一的标识表中的某一条记录。

在两个表的关系中，主关键字用来在一个表中引用来自于另一个表中的特定记录。

特点：**唯一 不为空**

1.2、外键

简介：

如果公共关键字在一个关系中是主关键字，那么这个公共关键字被称为另一个关系的外键。由此可见，外键表示了两个关系之间的相关联系。以另一个关系的外键作主关键字的表被称为主表，具有此外键的表被称为主表的从表。外键又称作外关键字。即：

外键是用来连接数据库的，保证数据库的参照完整性。

表的外键是另一表的主键，外键是可以有重复的，可以是空值。

以另一个关系的外键作主关键字的表被称为主表，具有此外键的表被称为主表的从表。

保持数据一致性，完整性，主要目的是控制存储在外键表中的数据,约束。使两张表形成关联，外键只能引用外表中的列的值或使用空值。

删除表时，只能先删除从表，再删除主表。

在开发的时候不会使用外键

在创建表的时候创建外键：

```
CREATE TABLE grade (
gradeID INT(11) NOT NULL,
gradeName VARCHAR(11) NOT NULL,
PRIMARY KEY (gradeID)
)ENGINE=INNODB DEFAULT CHARSET=utf8
-- 定义约束名
-- 定义约束名的内容 外键和引用
CREATE TABLE IF NOT EXISTS student (
id INT(4) NOT NULL AUTO_INCREMENT COMMENT '学号',
name VARCHAR(30) NOT NULL DEFAULT '匿名' COMMENT '姓名',
pwd VARCHAR(20) NOT NULL DEFAULT '123456' COMMENT '密码',
sex VARCHAR(2) NOT NULL DEFAULT '男' COMMENT '性别',
birthday DATETIME DEFAULT NULL COMMENT '生日',
address VARCHAR(100) DEFAULT NULL COMMENT '地址',
```

email VARCHAR(50) DEFAULT NULL COMMENT '邮箱',
gradeID INT(11) NOT NULL COMMENT '年级名称',
PRIMARY KEY (id),
KEY FK_gradeID(gradeID), --定义外键
CONSTRAINT FK_gradeID FOREIGN KEY (gradeID) REFERENCES grade(gradeID) --链接从表，即将外键 REFERENCES（参考）从表中的主键
) ENGINE=INNODB DEFAULT CHARSET=utf8

在创建表之后创建外键：
ALTER TABLE student ADD CONSTRAINT FK_gradeID FOREIGN KEY (gradeID) REFERENCES grade(gradeID);
1
删除外键：
ALTER TABLE student DROP FOREIGN KEY FK_gradeID; -- 删除表的外键，但是索引还存在；注：这个索引是在创建外键是自动生成的
ALTER TABLE student DROP INDEX FK_gradeID; -- 删除索引
-- 即：先删除外键再删除自动生成的索引

2、操作数据库中表中的数据
2.1、添加表中数据
INSERT INTO 表名(字段1,字段2,...) VALUES (值1,值2,...),(值1,值2,...),...
1
2.2、修改表中数据
UPDATE grade SET gradeName = '研一' WHERE gradeID = 1;
UPDATE 表名 SET 字段名 = 新值,... WHERE 条件
1
2
2.3、删除表中数据
删除一行或者多行数据
-- 删除数据
DELETE FROM grade WHERE gradeID BETWEEN 1 AND 4
DELETE FROM 表名 WHERE 条件
1
2
3
删除整张表 delete 和 truncate
-- delete：

INSERT INTO grade(gradeName) VALUES ('大一'),('大二'),('大三'),('大四')

DELETE FROM grade

-- truncate：

INSERT INTO grade(gradeName) VALUES ('大一'),('大二'),('大三'),('大四')

TRUNCATE grade

NOTE :

delete删除表时，自动增量不会变,truncate 删除表时，自动增量会置一

使用TRUNCATE TABLE不会对事务有影响

五、数据库查询语句（DQL，重点）

select语法

```
SELECT [ALL | DISTINCT]
{* | table.* | [table.field1[as alias1][,table.field2[as alias2]][,...]]}
FROM table_name [as table_alias]
[left | right | inner join table_name2] -- 联合查询
[WHERE ...] -- 指定结果需满足的条件
[GROUP BY ...] -- 指定结果按照哪几个字段来分组
[HAVING] -- 过滤分组的记录必须满足的次要条件
[ORDER BY ...] -- 指定查询记录按一个或多个条件排序
[LIMIT {[offset,]row_count | row_countOFFSET offset}];
-- 指定查询的记录从哪条至哪条
```

SQL的执行顺序：

–第一步：执行FROM [JOIN ON]

–第二步：WHERE条件过滤

–第三步：GROUP BY分组

–第四步：执行SELECT投影列

–第五步：HAVING条件过滤

–第六步：执行ORDER BY 排序

查询所有(ALL)：返回SELECT查询的所有记录结果,无论结果是否重复；SELECT语句默认为ALL

```
SELECT name AS '姓名' FROM mystudent;
SELECT ALL name AS '姓名' FROM mystudent;
-- 上述两条语句返回结果一致
1
2
3
```

去重(DISTINCT)： 去掉SELECT查询返回的记录结果中重复的记录 (返回所有列的值都相同) , 只返回一条

-- 用as重命名 将结果中表意不清的数据列重新命名

SELECT DISTINCT studentno AS '参加过考试的学生' FROM result

1

2逻辑表达式

操作符名称	语法	描述
IS NULL	a IS NULL	若操作符为NULL，则结果为真
IS NOT NULL	a IS NOT NULL	若操作符不为NULL，则结果为真
BETWEEN	a BETWEEN b AND c	若 a 范围在 b 与 c 之间，则结果为真
LIKE	a LIKE b	SQL 模式匹配，若a匹配b，则结果为真
IN	a IN (a1，a2，a3，.....)	若 a 等于 a1,a2..... 中的某一个，则结果为真

3、模糊查询：比较操作符

操作符名称	语法	描述
IS NULL	a IS NULL	若操作符为NULL，则结果为真
IS NOT NULL	a IS NOT NULL	若操作符不为NULL，则结果为真
BETWEEN	a BETWEEN b AND c	若 a 范围在 b 与 c 之间，则结果为真
LIKE	a LIKE b	SQL 模式匹配，若a匹配b，则结果为真
IN	a IN (a1，a2，a3，.....)	若 a 等于 a1,a2..... 中的某一个，则结果为真

NOTE :

使用 like 操作符时 %：表示任意字符 **：表示一个字符；即：a%可查询出a开头的任意字符长度的结果，而a_只能查询出a开头的两个字符的结果，a** _只能查询出a开头的三个字符的结果

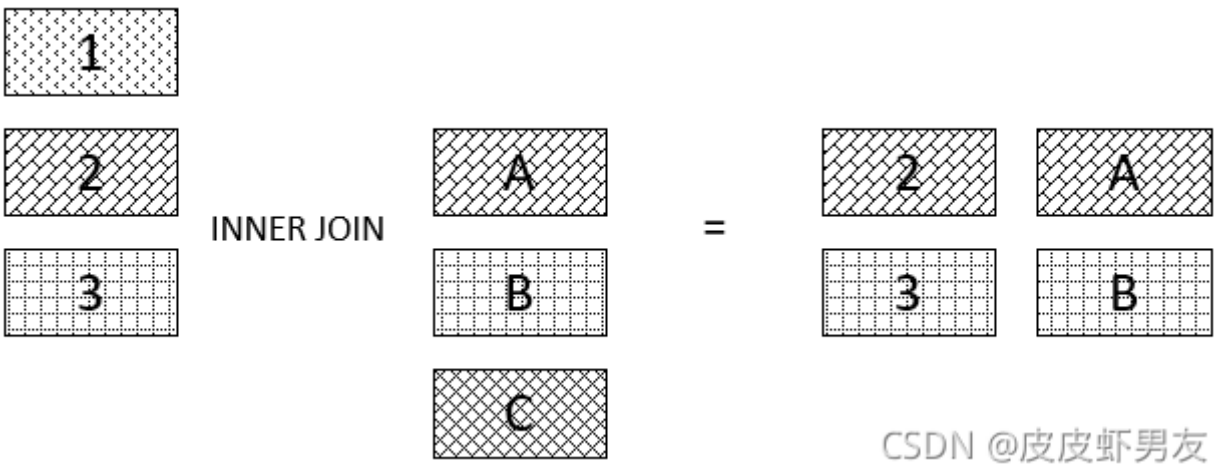
in(一个或者多个具体的结果)

1、连接查询

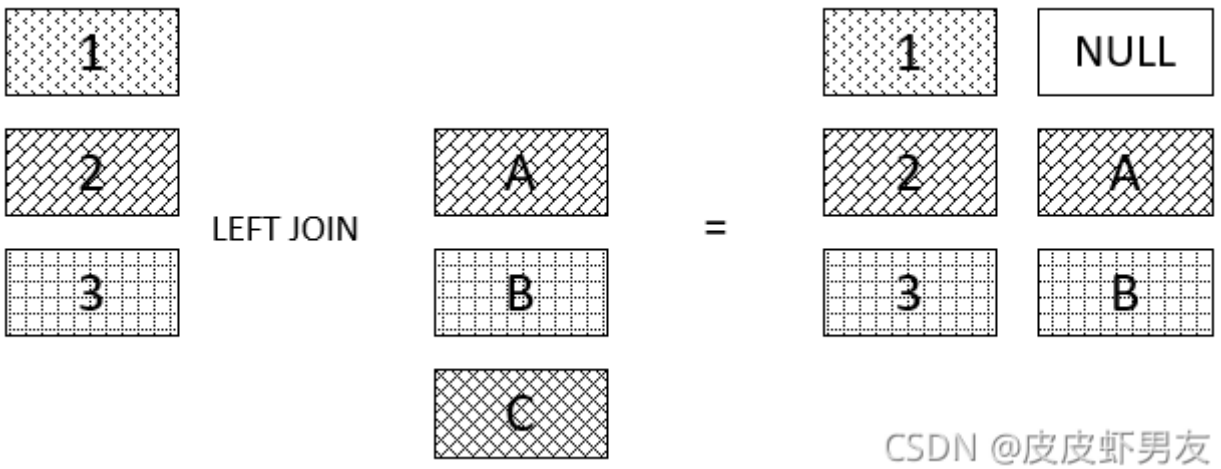
join 连接的区别：

操作符名称	描述
inner join	如果表中至少有一个匹配则返回行 ，即返回两表的交集
left join	(以左表作为基准,右边表来一一匹配,匹配不上的,返回左表的记录,右表以NULL填充)，即左表的记录均返回，右表中无左表记录的其余字段用null填充
right join	(以右表作为基准,左边表来一一匹配,匹配不上的,返回右表的记录,左表以NULL填充) ，即右表的记录均返回，左表中无左表记录的其余字段用null填充
full join	返回两表的并集
cross join	生成来自多个表的行的笛卡尔乘积

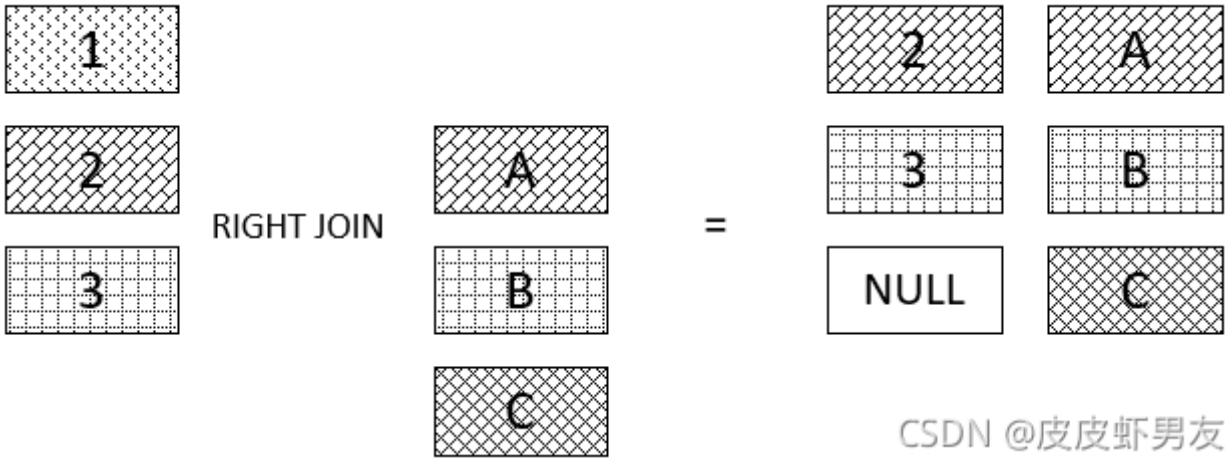
***inner join*（内连接）：**



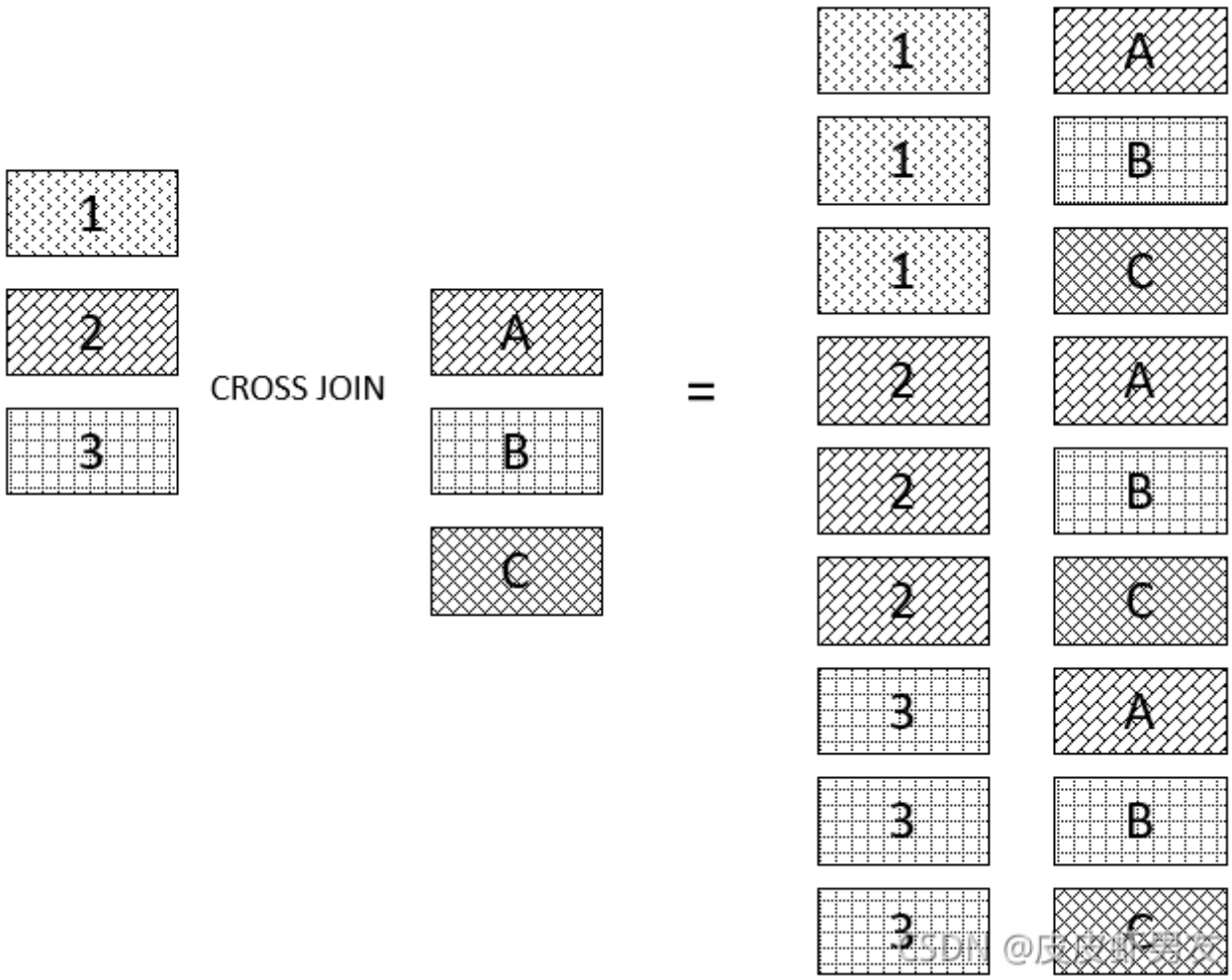
***left join*（左连接）：**



***right join*（右连接）：**



***cross join*（交叉连接）：**



思路：

分析需求,确定查询的列来源于那些表

是否使用连接查询，使用那种查询

确认表与表之间的交叉点

JOIN 语法：

SELECT [字段名1,字段名2,字段名3,...] --若左表和右表均有字段1，则需要指明字段1所属的一个表

FROM 左表 AS L

[INNER or LEFT or RIGHT or CROSS...] JOIN 右表 AS R ON L.字段名4 = R.字段名4----判别条件

```
-- 查询参加了考试的同学信息(学号,学生姓名,科目编号,分数)
-- 左连接实现
SELECT s.studentno,studentname,subjectno,studentresult
FROM student s
INNER JOIN result r ON s.studentno = r.studentno
-- 等值连接实现
SELECT s.studentno,studentname,subjectno,StudentResult
FROM student s , result r
WHERE r.studentno = s.studentno
-- 查询缺考学生(左连接实现)
SELECT s.studentno,studentname,subjectno,studentresult
FROM student s
LEFT JOIN result r ON s.studentno = r.studentno
WHERE studentresult IS NULL
-- 思考题:查询参加了考试的同学信息(学号,学生姓名,科目名,分数)
SELECT s.studentno,studentname,subjectname,studentresult
FROM student s
RIGHT JOIN result r ON s.studentno = r.studentno
INNER JOIN subject sub ON r.subjectno = sub.subjectno
```

where 与 on的区别

on 条件是生成临时表时使用的条件，它不管 on 中的条件是否为真，都会返回左边表中的记录。

where 条件是临时表生成好后，再对临时表进行过滤的条件。这时已经没有 left join 的含义（必须返回左边表的记录）了，条件不为真的就全部过滤掉。

2、排序和分页

```
/*============== 排序 ================
语法 : ORDER BY
ORDER BY 语句用于根据指定的列对结果集进行排序。
ORDER BY 语句默认按照ASC升序对记录进行排序。
如果您希望按照降序对记录进行排序，可以使用 DESC 关键字。
/SELECT s.studentno,studentname,subjectname,studentresultFROM student sINNER JOIN result r ON s.studentno = r.studentnoINNER JOIN subject sub ON sub.subjectno = r.subjectnoWHERE subjectname = '高等数学-1'ORDER BY studentresult DESC --按studentresult字段进行降序
12345678910111213/============== 分页 ================
语法 : SELECT * FROM table LIMIT (pageNo-1)*pageSzie,pageSzie
```

(pageNo-1)*pageSzie:起始值 pageSzie: 单页面显示条数

好处 : (用户体验,网络传输,查询压力)

推导:

第一页 : limit 0,5

第二页 : limit 5,5

第三页 : limit 10,5

.....

第N页 : limit (pageNo-1)*pageSzie,pageSzie

[pageNo:页码,pageSize:单页面显示条数]

*/

```
SELECT * FROM subject
LIMIT 0,4
```

3、子查询

子查询的定义：

子查询是将一个查询语句嵌套在另一个查询语句中；

在特定情况下，一个查询语句的条件需要另一个查询语句来获取，内层查询（inner query）语句的查询结果，可以为外层查询（outer query）语句提供查询条件。

由内及外. where 语句中的条件就是两张表的交叉点

```
-- 查询课程为 高等数学-1 且分数不小于80分的学生的学号和姓名
-- 连接查询
SELECT s.studentno,studentname
FROM student s
INNER JOIN result r ON s.studentno = r.studentno
INNER JOIN SUBJECT sub ON sub.subjectno = r.subjectno
WHERE subjectname = '高等数学-1' AND studentresult >= 80
-- 子查询：由内及外. where 语句中的条件就是两张表的交叉点
SELECT studentno,studentname
FROM student
WHERE studentno IN(
SELECT studentno FROM result WHERE subjectno = (
SELECT subjectno FROM subject WHERE subjectname = '高等数学-1'
)
)
```

五、MySQL函数

1、常用函数

1.1、数据函数

SELECT ABS(-8); /**绝对值**/

SELECT CEILING(9.4); /**向上取整**/

SELECT FLOOR(9.4); /**向下取整**/

SELECT RAND(); /**随机数,返回一个0-1之间的随机数**/

SELECT SIGN(0); /**符号函数: 负数返回-1,正数返回1,0返回0**/

1

2

3

4

5

1.2、字符串函数

SELECT CHAR_LENGTH('狂神说坚持就能成功'); /**返回字符串包含的字符数**/

SELECT CONCAT('我','爱','程序'); /**合并字符串,参数可以有多个**/

SELECT INSERT('我爱编程helloworld',1,2,'超级热爱'); /**替换字符串,从某个位置开始替换某个长度**/

SELECT LOWER('KuangShen'); /**小写**/

SELECT UPPER('KuangShen'); /**大写**/

SELECT LEFT('hello,world',5); /**从左边截取**/

SELECT RIGHT('hello,world',5); /**从右边截取**/

SELECT REPLACE('狂神说坚持就能成功','坚持','努力'); /**替换字符串**/

SELECT SUBSTR('狂神说坚持就能成功',4,6); /**截取字符串,开始和长度**/

SELECT REVERSE('狂神说坚持就能成功'); /*反转

1.3、日期和时间函数

SELECT CURRENT_DATE(); /**获取当前日期**/

SELECT CURDATE(); /**获取当前日期**/

SELECT NOW(); /**获取当前日期和时间**/

SELECT LOCALTIME(); /**获取当前日期和时间**/

SELECT SYSDATE(); /**获取当前日期和时间**/

-- 获取年月日,时分秒

SELECT YEAR(NOW());

SELECT MONTH(NOW());

SELECT DAY(NOW());

SELECT HOUR(NOW());

SELECT MINUTE(NOW());

SELECT SECOND(NOW());

1.4、系统信息函数

SELECT VERSION(); /**版本**/

SELECT USER(); /**用户**/

2、聚合函数

函数名	描述
count()	返回满足查询（Select）条件的 ***总和数***，如select count(*) [不建议使用，效率低]
min()	可以为**数值字段、字符字段或表达式列**做统计，***返回最小值***
max()	可以为**数值字段，字符字段或表达式列**作统计，***返回最大的值***
avg()	***返回一列的平均值***
sum()	***返回一列的总和***

NOTE: where不能使用聚合函数：聚集函数也叫列函数，它们都是基于整列数据进行计算的，而where子句则是对数据行进行过滤的，在筛选过程中依赖“基于已经筛选完毕的数据得出的计算结果”是一种悖论，这是行不通的。更简单地说，因为聚集函数要对全列数据时行计算，因而使用它的前提是：结果集已经确定。而where子句还处于“确定”结果集的过程中，因而不能使用聚集函数。与where子句不能出现聚集函数正相反的是，我们几乎看不到不使用聚集函数的having子句。为什么？因为在水平方向上根据外部指定条件的筛选（也就是对行的筛选），where子句可以独立完成，剩下的往往都是需要根据结果集自身的统计数据进一步筛选了，这时，几乎都需要通过having子句配合聚集函数来完成。/

count(*)包括了所有的列，统计行数，在统计结果的时候，不会忽略列值为NULL

count(1)忽略所有列，用1代表代码行，在统计结果的时候，不会忽略列值为NULL

count(列名)只包括列名那一列，在统计结果的时候，会忽略列值为空

（这里的空不是只空字符串或者0，而是表示null）的计数，即某个字段值为NULL时，不统计。

/**SELECT COUNT()** FROM student

SELECT COUNT(1) FROM student

SELECT COUNT(identitycard) FROM student

SELECT AVG(studentresult) AS 平均分 FROM result WHERE studentno = 1001

SELECT MIN(studentresult) AS 最低分 FROM result WHERE studentno = 1001

SELECT MAX(studentresult) AS 最高分 FROM result WHERE studentno = 1001

SELECT SUM(studentresult) AS 总分 FROM result WHERE studentno = 1001

3、分组

NOTE:在含有Group by子句的查询语句中，对select关键字后的目标列，存在以下规律：

使用group by 时，select 涉及的列要么是参与分组的列，要么列包含在聚合函数中

where将对分组前的所有数据进行筛选。having将对分组后的一组数据进行过滤。

```
-- 查询不同课程的平均分,最高分,最低分
-- 前提:根据不同的课程进行分组
SELECT subjectname,AVG(studentresult),MAX(studentresult),MIN(studentresult)
FROM result r
INNER JOIN subject sub ON r.subjectno = sub.subjectno
GROUP BY subjectname
```

4、MD5加密函数

4.1、MD5简介

MD5即Message-Digest Algorithm 5（信息-摘要算法5），用于确保信息传输完整一致。是计算机广泛使用的杂凑算法之一（又译摘要算法、哈希算法），主流编程语言普遍已有MD5实现。将数据（如汉字）运算为另一固定长度值，是杂凑算法的基础原理，MD5的前身有MD2、MD3和MD4。

4.2、实现数据加密

```
CREATE TABLE md5test (
id INT(11) NOT NULL ,
name VARCHAR(11) NOT NULL,
pwd VARCHAR(11) NOT NULL,
PRIMARY KEY (id)
)ENGINE=INNODB,DEFAULT CHARSET = utf8
INSERT INTO md5test VALUES (1,'zhangsan','123456'),(2,'lisi','123456'),(3,'wangwu','123456')
UPDATE md5test SET pwd = MD5(pwd) WHERE id = 1
INSERT INTO md5test VALUES(4,'kuangshen3',md5('123456'));
```

4.3、MD5实现数据匹配

```
SELECT * FROM md5test WHERE name = 'zhangsan' AND pwd = MD5('123456')
1
```

NOTE—MD5特点:

不可逆性 — 根据 MD5 值计算不出原始数据

唯一性 — 不同原始数据会有不同的 MD5 值

原文链接：https://blog.csdn.net/weixin_51356824/article/details/120744028