#今日内容
1、数据库的基本概念

2、MySQL数据库软件

```
1、安装
2、卸载
3、配置
```

3、SQL

##数据库的基本概念：
1、数据库的英文单词：DataBase 简称： DB

```
2、什么是数据库？
    *用于存储和管理的仓库
3、数据库的特点：
    1、持久化储存数据的。其实数据库就是一个文件系统
    2、方便存储和管理数据
    2、使用同意的方式操作数据库---->SQL
```

1、安装

2、配置

```
参见 mysql 5.27.9配置环境.png
启动服务启动：
        1、手动，windows搜索服务且进入，选择mysql，运行
        2、cmd---->services.msc进入服务，选择mysql，运行
cmd打开mysql服务：(管理员打开)
        net start mysql
cmd停止mysql服务:(管理员打开)
        net stop mysql
mysql登录：
        1、mysql -u用户名 -p密码
        2、mysql -i用户名 -p；敲回车后，输入密码
        3、mysql --host=所需ip(自己的是12.0.0.1) --user=用户名 --password=密码
远程登陆mysql：
        mysql -h所需ip地址 -u用户名 -p密码
mysql退出：
        1、exit
        2、quit
```

3、卸载

```
*复制 datadir = "c:/ProgramData/MySQL Server 2.2/Data"(或者是自己配置的data文件夹)
```

4、mysql目录结构

```
1、mysql安装目录
    bin：一些2进制的可执行文件
    data：数据目录，包含日志文件和数据文件
    include：c语言的一些头信息
```

```
    lib：mysql所需的一些架包
    share：mysql的一些错误信息
    my.ini：mysql的配置文件

2、mysql数据库目录---->卸载时所需的目录
    mysql：以frm结尾的都是数据库的表

结论：
    数据库：文件夹
    表：文件
    数据：数据
```

#SQL

1、什么是SQL：(Relational DBMS都是关系型数据库)

Structured Query Language：结构化查询语言

其实就是定义了操作所有关系型数据库的规则

每一种数据库操作的方式c存在不一样的地方，称为"方言"

```
2、通用语法
    1、SQL语句可以单行或多行书写，以分号结尾
    2、可使用tab键以缩进增强预计的可读性
    3、MySQL数据库的SQL预计不区分大小写，关键字建议使用大写
    4、注释：
        *单行注释：  -- (两个减号后面要加空格)  或   #(MySQL特有)
        *多行注释：/*注释*/
3、SQL分类：
    1、DDL(Data Definition language)数据定义语言
        用来定义数据库对象：数据库，表，列等等
        关键字：create，drop，alter等
    2、DML(Data Manipulation language)数据操作语言
        用来对数据库中表的数据进行增删改
        关键字：insert，delete，uodate等
    3、DQL(Data Query Language)数据查询语言
        用来查询数据库中表的记录
        关键字：select，where等
    4、DCL(授权):(Data Control Language)：数据控制语言(了解)
```

##DDL：操作数据库、表

1、操作数据库：CRUD

1、C(Create)：创建

```
        create database db1(db1是数据库的名称)：表示创建一个数据库

        create database if not exists dbq;：判断要创建的数据库是否存在(如果不用判断，且有创建的有重复的数据库，则会包异常)

        create database db3 character set gbk;创建一个gbk的db3数据库

        create database if not exists db4 character set gbk;：判断创建的gbk格式的db4数据库是否已经存在

    2、R(Retrieve)：查询
        1、查询所有数据库的名称：
        show databases;(大写)
```

```
            information_schema:数据库中的信息，有哪些表，库的名字
            mysql:mysql中服务器的核心数据库
            performance_schema：对性能提升，做一些操作的一些表(不是对mysql很熟悉，那么不建议动用)

        show create database mysql(mysql是数据库的名称);：查询创建数据库mysql语法(查询某个数据库的字符集)
```

```
    3、U(Update)：修改
    4、D(Delete)：删除
    5、使用数据库
```

2020/3/3 11:41:31

##使用数据库：

```
3、U(Update)：修改
    alter database 数据库名称 character set 字符集名称
4、D(Delete)：删除
    drop database 数据库名称;
    如果数据库存在，则可以删除
        drop database if exists 数据库名称;

5、使用数据库
    查询当前正在使用的数据库
        selece database();
    使用数据库
        use 数据库名称;
```

##操作表

```
1、创建用户
    CREATE USER 用户名@IP地址 IDENTIFIED BY '密码';
    用户只能在指定的IP地址上登录
    CREATE USER 用户名@'%' IDENTIFIED BY '密码';
    用户可以在任意IP地址上登录

2、修改用户密码
update user set passeord = password('新密码') where user='用户名';    -- 普通方式
SET PASSWORD FOR '用户名'@'主机名' = PASSWORD('新密码');     -- 特有的简化方式

3、 给用户授权
GRANT 权限1, … , 权限n ON 数据库.* TO 用户名@IP地址
    权限、用户、数据库
    给用户分派在指定的数据库上的指定的权限
    例如：GRANT CREATE,ALTER,DROP,INSERT,UPDATE,DELETE,SELECT ON mydb1.* TO user1@localhost;

    给user1用户分派在mydb1数据库上的create、alter、drop、insert、update、delete、select权限
    GRANT ALL ON 数据库.* TO 用户名@IP地址;
    给用户分派指定数据库上的所有权限

4、 撤销授权
    REVOKE 权限1, … , 权限n ON 数据库.* FROM 用户名@IP地址;
```

撤消指定用户在指定数据库上的指定权限
例如：REVOKE CREATE,ALTER,DROP ON mydb1.* FROM user1@localhost;
撤消user1用户在mydb1数据库上的create、alter、drop权限

5、 查看权限
SHOW GRANTS FOR 用户名@IP地址
查看指定用户的权限

6、C(Create)：创建(表)
1.1、语法

```
create table 表名(
    列名1 数据类型,
    列名2 数据类型,
    ...
    列名n 数据类型     #注意此处不能再有逗号
);
```

数据库类型

分类	类型名称	类型说明
整数	tinyInt	微整型：很小的整数(占 8 位二进制)
	smallint	小整型：小的整数(占 16 位二进制)
	mediumint	中整型：中等长度的整数(占 24 位二进制)
	int(integer)	整型：整数类型(占 32 位二进制)
	age int,	
小数	float	单精度浮点数，占 4 个字节 小数
	decimal（p(1~65),d(0~30)）	d<=9;用于在数据库中存储精确的数值，常用于银行帐目计算(对应java：java.math.BigDecimal)
	以串存放,p(旧版本中1～254)占两个字节	
	double	双精度浮点数，占 8 个字节
	money double(5,2),	#表示共有5位，其中2位是小数
日期	time	表示时间类型
	date	表示日期类型
	date yyyy-MM-dd,	#包含年月日
	datetime	同时可以表示日期和时间类型
	age yyyy-MM-dd HH:mm:ss,	#包含年月日和时分秒
	timestamp	时间戳。范围是'1970-01-01 00:00:00'到2037年。
	age yyyy-MM-dd HH:mm:ss,	#如果将来不给此字段赋值,或赋值为null,则默认使用当前的系统时间，自动赋值
字符串	char(m) 固定长度的字符串，无论使用几个字符都占满全部，M 为 0~255 之间的整数	
	varchar(m) 可变长度的字符串，使用几个字符就占用几个，M 为 0~65535 之间的整数	
	name varcahr(20)	#表示最多有20个字符
大二进制	tinyblob	
	Big Large Object	允许长度 0~255 字节
	blob	允许长度 0~65535 字节
	mediumblob	允许长度 0~167772150 字节
	longblob	允许长度 0~4294967295 字节
大文本	tinytext	允许长度 0~255 字节
	text	允许长度 0~65535 字节
	mediumtext	允许长度 0~167772150 字节
	longtext	允许长度 0~4294967295 字节

```
        1.2、创建表
            create table student(
                id int,
                name varchar(32),
                age int,
                score double,
                birthday date,
                insert_time timestamp
            );

        1.3、复制表;
            create table 新表名 like 要复制的表名;
```

```
7、R(Retrieve)：查询
    查询某个表中所有表的名称
        show tables;
    查询表结构
        desc 表名;
    2、查看表的字符集(所有定义的字段及属性)
        show create table 表名;

8、U(Update)：修改
    1、修改表名
        alter table 表名 rename to 新表名;
    2、修改表的字符集
        alter table 表名 character set 字符集;
    3、添加一列
        在表的最后一列增加新的一列：
            alter table 表名 COLUMN add 列名 数据类型 约束;
        在指定的位置增加新的一列：
            alter table 表名 COLUMN add 列名 数据类型 约束 after 列名;
        在第一列增加新的一列：
            alter table 表名 COLUMN add 列名 数据类型 约束 first;
    4、修改列名称(还有类型)
        4.1、alter table 表名 change 列名称 新的列名 数据类型;
        4.2、alter table 表名 modify 列名 数据类型;       #不能修改列名称,只能修改属性
    5、删除列
        alter table 表名 drop 列名;

9、D(Delete)：删除
    drop table if exists 表名;     #进行判断，在删除;也可以不判断删除(drop table 表名;)

10、重置 root 用户密码

    1、命令提示符在管理员模式下停止 MySQL 服务： net stop mysql
    2、管理员模式下使用无验证方式启动MySQL服务：  mysqld --skip-grant-tables
    3、开启新的命令行窗口进入MySQL： mysql
    4、进入 'mysql' 数据库： USE mysql;
    5、使用修改用户密码语句对root密码进行重置
    6、启动"运行"窗口，运行命令结束"mysqld"进程： taskkill /im mysqld.exe /f
    7、命令提示符在管理员模式下启动 MySQL 服务： net start mysql
    8、可以用新密码登陆 root 用户。
```

#DML:增删改表中的数据

```
1、添加数据
        语法：注意以下三点：
        1、(列名和值要11对应，少一个则添加失败)
        2、表名后，不写列名，则默认给所有的列添加值
        3、除了数字类型，其他都要使用引号引起来(单双都可以)
            insert into 表名 (列名1,列名2,...列名n) values (值1,值2,...值n);
            添加全部的列数据，简写---->不知道，某列给什么值的时候，可以写null
            insert into 表名 values (值1,值2,...值n)；
            添加多行数据
            insert into 表名 values (值1,值2,...值n)，(值1,值2,...值n)...；
2、删除数据

3、更改数据
```

#DQL:查询表中的记录

```
select * from 表名;        #查询表中所有的数据
```

2020/4/2 23:05:16

##DML:增删改表中的数据

```
2、删除数据
    语法：
        delete from 表名 [where 条件];
    注意：
        1、如果不加条件，则删除整个表
        2、如果要删除所有的记录
        有多少条记录，则会删除多少次，效率低；且删不掉自增主键
        delete from 表名;-->  不推荐，(删除表全部数据，表结构不变，的删除操作作为事务记录在日志中保存以便进行进行回滚操作；操作不会减少表或索引所占用的空间)

        先删除整个表,类似调用drop语句删除，再创建一个一模一样的表
        truncate table stu;-->推荐，(删除表全部数据，保留表结构，在删除的过程中不会激活与表有关的删除触发器。执行速度快；表和索引所占用的空间会恢复到初始大小)

        删除整个表，且所有的结构都没了
        drop  table  表名;
```

3、更改数据

语法：

update 表名 set 列名1=值1,列名2=值2,...[where 条件];

注意：

1、如果不加条件，则会将表中(列)的所有数据全部修改

#DQL:查询表中的记录

```
select * from 表名;        #查询表中所有的数据

1、语法
    select  字段列表1,字段列表2,...
```

```
    from    表名列表1，表名列表2,...
    where   条件列表1,条件列表2,...
    group by    分组字段
    having  分组之后的条件
    order by    排序
    limit   分页限定
2、基础查询
    1、多个字段的查询
        SELECT 字段1,字段2,字段3... FROM 表名;
        注意：
            如果查询所有的字段，可以使用*代替字段列表
    2、去除重复
        SELECT DISTINCT 字段1,字段2... FROM 表名;
        注意：
            必须要查询的字段中的一行或多行数据，完全一样时，才会去重复
    3、计算列
        一般可以使用四则运算（+，-，*，/）计算一些列的值。(一般只会进行数值型的计算)
        注意：
            null参与的运算，运算结果都为null
    4、把可能是null的字段替换为0(或某个值)
        SELECT 字段1,字段2,IFNULL(可能为null的字段3,0) FROM 表名;
    5、起别名
        SELECT 字段1 AS 别名,字段2 AS 别名... FROM 表名;
3、条件查询
    1、where后面跟条件
    2、运算符
        <(小于),>(大于),<=(小于等于),>=(大于等于),=(等于),<>（不等于）
        BETWEEN..AND---->在...范围---->between 100 and 200
        IN(集合)---->等于集合中的数据-->name in('张三','李四')
        LIKE '张%'---->模糊查询-->以张开头的
            占位符：
                _：单个任意字符
                %：任意多个字符
                []：表示括号内所列字符中的一个（类似正则表达式）。指定一个字符、字符串或范围，要求所匹配对象为它们中的任一个。
                    例：SELECT * FROM [user] WHERE u_name LIKE '[张李王]三';

                [^] ：表示不在括号所列之内的单个字符。其取值和 [] 相同，但它要求所匹配对象为指定字符以外的任一个字符
                    例：SELECT * FROM [user] WHERE u_name LIKE '[^张李王]三';

        IS NULL---->查询是null的数据（is not null 表示不为空）
        and或&&---->一样的效果，都是要同时满足
        or或||---->一样的效果，都是只需满足一个条件即可
        not或！---->一样的效果，都是进行取反
```

2020/4/3 22:46:21

#DQL：查询语句

4、查询语句

1、排序查询

语法：order by 子句

order by 排序字段1 排序方式1, 排序字段2 排序方式2...

select * from stu order by math ASC, english DESC; #数学升序，当排序的值一样时，英语降序排列

排序方式

ASC:升序，默认

DESC：降序

注意：

如果有多个排序条件，则当前面的条件值一样时，才会进行(依次)判断后面的条件

```
2、聚合函数：将一列数据作为一个整体，进行纵向的计算
    1、count：计算个数
        select COUNT(name) from stu;    #统计人数
    2、max：计算最大值
        select MAX(math) from stu;
    3、min：计算最小值
        select MIN(math) from stu;
    4、sum：计算和
        select SUM(math) from stu;      #会将null排除
    5、avg：计算平均值
        select AVG(math) from stu;
    6、LTRIM(str)去除左空格函数
        select LTRIM(subject) from stu;
    7、RTRIM(str)去除右空格函数
        select RTRIM(subject) from stu;

    注意：
        聚合函数的计算，排除null值 #null值，不算在其中
        解决方案：
            1、选择不包含非空的列进行计算(一半选择非空的列：主键)；也可以使用count(*)：表示只要有一列不为null，就算一条数据
            2、使用IFNULL()函数-->select COUNT(IFNULL(name,0)) from stu;

3、分组查询
    1、语法：group by 分组字段;
        SELECT 字段 FROM 表名 GROUP BY 分组的字段;

        select  字段.....
        from    表名.
        where   筛选条件.
        group by    分组字段
        having  分组之后的筛选
        limit   限定显示的行
        order by    排序字段

    注意:
        1、分组之后查询的字段：分组字段、聚合函数
        2、where和having的区别
            select 分组字段,分组函数, from 表名 group by 分子字段;
            1、where在分组之前进行限定，如果不满足条件，则不参与分组
               having在分组之后进行限定，如果不满足结果，则不会被查询出来
            2、where后不能跟聚合函数，having可以进行聚合函数的判断

4、分页查询
    1、语法：limit 开始的索引, 每页查询的条数;
        SELECT * FROM stu LIMIT 0,3;
    2、公式：开始的索引 = (当前页码 - 1) * 每页显示的条数
        SELECT * FROM stu LIMIT 3,3;
        SELECT * FROM stu LIMIT 6,3;
    3、limit 是mysql的方言
```

5、约束

概念：对表中的数据进行限定，保证数据的正确性、有效性和完整性

分类：

1、主键约束：primary key

1、在创建表时添加主键约束

```
create table stu(
    id int primary key,    #给id添加主键约束
    name varchar(20)
);
```

2、删除主键

```
alter table stu4 drop primary key;
```

3、注意：

1、含义：非空且唯一

2、一张表只能有一个字段为主键

3、主键就是表中记录的唯一标识

2、非空约束：not null

1、创建表时，添加约束

```
CREATE TABLE stu2(
    id INT,
    NAME VARCHAR(20) NOT NULL    -- name列不能为空
);
```

2、创建表完成之后，添加非空约束

```
ALTER TABLE 表名 MODIFY 列名 数据类型 NOT NULL;
```

3、删除非空约束(不跟not null约束即可)

```
ALTER TABLE 表名 MODIFY 列名 数据类型;
```

```
3、唯一约束：unique
    1、创建表时，添加约束
        CREATE TABLE stu3(
            id INT,
            phone_number VARCHAR(20) UNIQUE      -- 添加了唯一约束
        );
    2、唯一约束中，可以添加null(此处的null可以理解为不确定因素)，且可以有多个
        INSERT INTO stu3 VALUES (1,NULL);
        INSERT INTO stu3 VALUES (1,NULL);
    注意：mysql中，唯一约束限定的列的值可以有多个null
    3、删除唯一约束
        ALTER TABLE 表名 DROP INDEX 唯一约束的列名;

4、外键约束：foreign key

5、自动增长：(一般配合int值的主键一起使用)
    1、概念：如果某一列是数值类型的，使用auto_increment 可以来完成值的自动增长
    2、创建主键，且完成主键的自增长
        CREATE TABLE stu5(
            id int primary key auto_increment,
            name varchar(20)
        );
```

6、多表之间的关系

7、范式

8、数据库的备份和还原

2020/4/5 18:45:43

#DQL：查询语句

4、外键约束：foreign key；让表与表产生关系，从而保证数据的正确性。

1、创建表时，可以添加外键

语法：

create table 表名(

...

外键列

constraint 外键名称(自定义) foreign key 外键列名称(本表) references 主表名称(主表列名称);

)

2、查看外键名称

SHOW CREATE TABLE 表名;

3、删除外键

alter table 表名 drop foreign key 外键名称;

```
    4、创建表之后，添加外键
        alter table 表名 add constraint 外键名称 foreign key (外键字段名称)
references 主表名称(主表列名称);
    5、级联操作-->需谨慎使用，因为表与表之间，同时更新和删除，比较危险
        1、添加级联操作
            语法-->后面两个条件：级联更新和级联删除
            alter table 表名 add constraint 外键名称
            foreign key (外键字段名称) references (主表名称) on update cascade on
delete cascade;
        2、分类-->可同时创建更新和删除；也可以单独更新
            单独删除：那么单独更新和删除时，自定义的列名不能相同，且要分开删除级联更新和级联删
除
            1、级联更新：on update cascade
            2、级联删除：on delete cascade
```

6、多表之间的关系

7、范式

8、数据库的备份和还原

#数据库的设计：

```
1、多表之间的关系
    1、分类
        1、一对一(了解)：
```

如：人和身份证
分析：一个人只有一个身份证,一个身份证只能对应一个人
2、一对多(多对一)：
如：部门和员工
分析：一个部门有多个员工，一个员工只能对应员工部门
3、多对多：
如：学生和课程
分析：一个学生可以选择很多门课程，一个课程可以被很多学生选择
2、实现方式：
1、一对多：
如：部门和员工
实现方式：在多的一方建立外键，指向一的一方的主键
2、多对多：
如：学生和课程
实现方式：多对多关系需要借助第三张中间表，中间表至少包含两个字段，则两个字段作为第三张表的外键，分别指向两张表的主键
3、一对一(了解)：-->最好是合成一张表
如：人和身份证
实现方式：可以在任意一方添加唯一外键指向另一方的主键

2、数据库设计的范式
概念：设计数据库时，需要遵循的一些规范

设计关系数据库时，遵从不同的规范要求，设计出合理的关系型数据库，这些不同的规范要求被称为不同的范式，
各种范式呈递次规范，越高的范式数据库冗余越小。

目前关系数据库有六种范式：第一范式（1NF）、第二范式（2NF）、第三范式（3NF）、
巴斯-科德范式（BCNF）、第四范式(4NF）和第五范式（5NF，又称完美范式）

分类：
1、第一范式（1NF）：每一列都是不可分割的原子数据项
2、第二范式（2NF）：在1NF的基础_上，非码属性必须完全依赖于码(在1NF基础上消除非主属性对主码的部分函数依赖)
掌握几个概念：
1、函数依赖：A---->B，两个通过A属性(属性组),可以确定一个B属性的值,则称B依赖于A
例如：学号-->姓名 (学号,课程名称)-->分数
2、完全函数依赖：A---->B,如果A是一个属性组,则B属性值得确定需要依赖于A属性组中所有的属性值
例如：(学号,课程名称)-->姓名
4、传递函数依赖：A-->B,B-->C；如果通过A属性(属性组)的值,可以确定唯一B属性的值,在通过B属性(属性组)的值可以确定唯一C属性的值，则称C传递函数依赖A
例如：学号-->系名，系名-->系主任
5、码：如果在一张表中，一个属性或属性组，被其他所有属性完全依赖，则称这个属性(属性组)为该表的码
例如：改表中码为：(学号，课程名称)
主属性：码属性组中的所有属性
属性：除过码属性组的属性

3、第三范式（3NF）：在2NF基础上，任何非主属性不依赖于其它非主属性(在2NF基础上消除传递依赖)

#数据库的备份和还原
1、命令行：(处于未登录时，且不能有分号)
备份语法：mysqldump -u用户名 -p密码 数据库名称 > 保存的路径.sql
备份多个数据库：mysqldump -uroot -proot --databases 数据库1 数据库2 > 保存路径.sql

备份所有数据库：mysqldump -u用户名 -p密码 --all-databases > 保存的路径.sql
备份单表：mysqldump -uroot -proot 数据库名 表名 > 保存的路径.sql
备份多表:mysqldump -uroot -proot 数据库名 表1 表2 表3 > 保存的路径.sql
还原语法：
1、登录数据库
2、创建数据库
3、使用数据库
4、执行文件：sourse 文件路径://文件名称.文件后缀(登陆后，创建相同数据库名，进入数据库，然后还原，且不带分号)
2、图形化工具

2020/4/7 22:29:14

#多表查询：
查询语法：
select
列名列表
from
表名列表
where...

##多表查询分类：
1、多表查询
创建表(两个)

```
-- 创建部门表
CREATE TABLE dept(
    id INT PRIMARY KEY AUTO_INCREMENT,
    NAME VARCHAR(20)
);
INSERT INTO dept (NAME) VALUES ('开发部'),('市场部'),('财务部');
-- 创建员工表
CREATE TABLE emp (
    id INT PRIMARY KEY AUTO_INCREMENT,
    NAME VARCHAR(10),
    gender CHAR(1), -- 性别
    salary DOUBLE, -- 工资
    join_date DATE, -- 入职日期
    dept_id INT,
    FOREIGN KEY (dept_id) REFERENCES dept(id) -- 外键，关联部门表(部门表的主键)
);
INSERT INTO emp(NAME,gender,salary,join_date,dept_id) VALUES('孙悟空','男',7200,'2013-02-24',1);
INSERT INTO emp(NAME,gender,salary,join_date,dept_id) VALUES('猪八戒','男',3600,'2010-12-02',2);
INSERT INTO emp(NAME,gender,salary,join_date,dept_id) VALUES('唐僧','男',9000,'2008-08-08',2);
INSERT INTO emp(NAME,gender,salary,join_date,dept_id) VALUES('白骨精','女',5000,'2015-10-07',3);
INSERT INTO emp(NAME,gender,salary,join_date,dept_id) VALUES('蜘蛛精','女',4500,'2011-03-14',1);
```

```
查询语法：
    select
```

列表名称
from
表名列表
where...
笛卡尔积：
有两个集合A,B：取这两个集合的所有组成情况(总共有：A*B条数据)
要完成多表查询，需要消除无用的数据

多表查询的分类：
1、内连接查询：
1、隐式内连接：使用where条件消除无用数据
- 查询员工表的名称，性别。部门表的名称
SELECT emp.name,gender,dept.name FROM emp,dept WHERE emp.dept_id=dept.id;

-- 给表名取别名(比较正经的写法)
SELECT
e.name, -- 员工表的姓名
e.gender, -- 员工表的性别
d.name -- 部门表的名称
FROM
emp e, -- 员工表
dept d -- 部门表
WHERE
e.dept_id=d.id;

2、显示内连接(sql标准的)：
语法：select 字段列表 from 表名1 inner join 表名2 on 条件(inner可选择性省略)
SELECT * FROM emp INNER JOIN dept ON emp.dept_id = dept.id;
SELECT * FROM emp JOIN dept ON emp.dept_id = dept.id;
3、自然连接(用的少，不建议)：
/*
自然连接是内连接的特殊方式，他会自动的在所连接的表中寻找表和表之间关联的条件，找到之后会将这些关联条件自动关联起来
自然连接，慎用，会引起不必要的查询口径错乱
*/
SELECT 字段列表 FROM 表1 别名1 NATURAL JOIN 表2 别名2
4、内连接查询：
1、从那些表中查询数据
2、条件是什么
3、查询哪些字段

2、外连接查询：
1、左外连接
语法：select 字段列表 from 表1(主表) left outer join 表2(从表) on 条件;(outer可选择性省略)
左外自然：SELECT * FROM 表1 别名1 NATURAL LEFT OUTER JOIN 表2 别名2 ON 别名1.xx=别名2.xx(用的较少，易出错)
查询的是左表所有数据以其交集部分

2、右外连接：
语法：select 字段列表 from 表1(从表) right outer join 表2(主表) on 条件;(outer可选择性省略)
右外自然：SELECT * FROM 表1 别名1 NATURAL RIGHT OUTER JOIN 表2 别名2 ON 别名1.xx=别名2.xx(用的较少，易出错)
查询的是右表所有数据以其交集部分

3、全链接：(需要表之间的结构一致)

全链接：可以使用UNION来完成全链接

例：

```
SELECT * FROM stu
UNION
SELECT * FROM score
```

3、子查询：

概念：查询中嵌套查询，称嵌套查询为子查询

```
-- 查询工资最高的员工信息
SELECT MAX(salary) FROM emp;             -- 9000

-- 查询员工信息，并且工资等于9000的
SELECT * FROM emp WHERE emp.salary=9000;    -- 唐僧

-- 简化为一条sql语句
SELECT * FROM emp WHERE emp.salary=(SELECT MAX(salary) FROM emp);
```

子查询不同情况

1、子查询的结果是单行单列的：

子查询可以作为条件，使用运算符去判断；运算符：>,>=,<,<=,=

```
-- 查询员工工资小于平均工资的人
```

未简化：

```
-- 查询工资最高的员工信息
SELECT MAX(salary) FROM emp;
-- 查询员工信息，并且工资等于9000的
SELECT * FROM emp WHERE emp.salary=9000;
```

子查询简化：

```
SELECT * FROM emp WHERE emp.salary < (SELECT AVG(salary) FROM emp); -- 三条数据
```

2、子查询的结果是多行单列的：

子查询可以作为条件，使用运算符in来判断

```
-- 查询'财务部'和'市场部'所有的员工信息
SELECT id FROM dept WHERE NAME = '财务部' OR NAME = '市场部';     -- 2,3
SELECT * FROM emp WHERE dept_id = 3 OR dept_id = 2;         -- 共三条数据
-- 简化为一条sql语句----->子查询
SELECT * FROM emp WHERE dept_id IN (SELECT id FROM dept WHERE NAME IN ('财务部','市场部'));
```

3、子查询的结果是多行多列的：(日期处要用引号引上)

子查询可以作为一张虚拟表，参与查询

```
查询员工入职日期为2011-11-11日之后的员工信息和部门信息
SELECT * FROM emp WHERE emp.join_date > '2011-11-11';
-- 普通内连接
SELECT
    *
FROM
    emp e,
    dept t
WHERE
    e.dept_id = t.id AND e.join_date > '2011-11-11';
-- 使用子查询简化后
SELECT * FROM dept t, (SELECT * FROM emp WHERE emp.join_date > '2011-11-11') WHERE t.id = e.dept_id;
```

```
2、事务
    1、事务的基本介绍
        1、概念：
            两个一个包含多个步骤的业务操作，被事务管理，那么这些操作要么同时成功，要么同时失败
    2、操作：
        1、开启事务：start transaction;
        2、回滚：rollback;
        3、提交：commit;
            例：
                CREATE TABLE account (
                    id INT PRIMARY KEY AUTO_INCREMENT,
                    NAME VARCHAR(10),
                    balance DOUBLE
                );
                -- 添加数据
                INSERT INTO account (NAME, balance) VALUES ('zhangsan', 1000),
('lisi', 1000);
```

```
            SELECT * FROM account;
            UPDATE account SET balance = 1000;

            -- 模拟转账
            -- 张三给李四转账 500 元
            -- 1. 张三账户 -500
            UPDATE account SET balance = balance - 500 WHERE NAME = 'zhangsan';
            -- 2. 李四账户 +500
            UPDATE account SET balance = balance + 500 WHERE NAME = 'lisi';


            -- 模拟转账异常
            -- 1. 张三账户 -500
            UPDATE account SET balance = balance - 500 WHERE NAME = 'zhangsan';
            异常...
            -- 2. 李四账户 +500
            UPDATE account SET balance = balance + 500 WHERE NAME = 'lisi'; -- zhangsan 500,lisi
1000


            -- 利用事务控制
            START TRANSACTION;
            UPDATE account SET balance = balance - 500 WHERE NAME = 'zhangsan';
            -- 异常...
            -- 2. 李四账户 +500
            UPDATE account SET balance = balance + 500 WHERE NAME = 'lisi'; -- zhangsan 500,lisi
1000(此处的只是临时数据，重新关闭窗口后查询仍然是之前的数据)

            -- (异常注释了之后)执行没问题，提交事务
            COMMIT;
            -- 3、回滚全部数据(在哪里开启的事务，就会回滚到哪里)
            -- ROLLBACK TO 事务保存点：回滚至保存点
            ROLLBACK;
        4、mysql数据库中事务默认自动提交
          事务提交的两种方式：
```

自动提交

mysql就是自动提交

一条DML(增删改)语句会自动提交一次事务

手动提交：

oracle数据库默认是手动提交事务

需要先开启事务，在提交

修改事务的默认提交方式：

查看事务的默认提交方式：select @@autocommit;(1：代表自动提交；0：代表手动提交)

修改默认提交方式：set @@autocommit = 0;(会有一个临时修改；但是重新开启窗口时，数据仍然是没修过的)；需要自己写commit;

2、事务的四大特征

1、原子性：是不可分割的最小操作单位，要么同时成功，要么同时失败

2、持久性：但事务提交或回滚后，数据库会持久化的保存数据

3、隔离性：多个事务之间：相互影响

4、一致性：事务操作前后，数据总量不变

3、事务的隔离级别(了解)

概念：多个事务之间隔离的，相互独立的，但是如果多个事务操作同一批数据，则会引发一些问题，设置不同的隔离级别就可以解决这些问题

存在问题：

1、脏读：一个事务，读取到哪一个事务中没有提交的数据

2、不可重复(虚读)：在同一个事务中，两次读取到的数据不一样

3、幻读：一个事务(DML)，数据表中所有的记录，另一个事务添加了一条数据，则第一个事务查询不到自己的修改

隔离级别：

1、read uncommitted：读未提交

产生的问题：脏读、不可重复读、幻读

2、read committed：读已提交(Oracle默认隔离级别)

产生的问题：不可重复，幻读

3、repeatable read：幻读(mysql默认隔离级别)

产生的问题：幻读

4、serializable：串行化(另一边提交后，马上就能查出数据；否则另一个表查询一直处于等待状态)

可以解决所有的问题

注意：隔离级别从小到大安全性越来越高，但是效率越来越低

数据库查询隔离级别：

```
select @@tx_isolation;
```

数据库设置隔离级别：(设置后，需要重启软件，才会生效)

```
set global transaction isolation level 级别字符串;
```

演示：

```
-- 设置隔离级别为：读未提交的
set global transaction isolation level read uncommitted;
-- 开启事务
start transaction;
-- 转账操作
update account set balance = balance - 500 where id = 1;
update account set balance = balance + 500 where id = 2;
```

#DCL

DCL：管理用户，授权

1、管理用户

1、添加用户：

CREATE USER '用户名'@'主机名ip(%通配符，表示所有机器都可以访问)' IDENTIFIED BY '密码';

2、删除用户：(host列的名称)

drop USER '用户名'@'主机名';

3、修改用户密码：

5.7版本前

UPDATE USER SET PASSWORD = PASSWORD('新密码') WHERE USER = '用户名';

此方法通用(简化版)

SET PASSWORD FOR '用户名'@'主机名' = PASSWORD('新密码');

```
                mysql中忘了root用户的密码？
                1、停止mysql服务(需要管理员权限)-->net stop mysql
                2、使用无验证方式启动mysql服务：mysqld --skip-grant-tables(会无反应)
                3、新开cmd窗口，直接输入mysql，回车，即登录成功
                4、use mysql
                5、SET PASSWORD FOR '用户名'@'主机名' = PASSWORD('新密码');
                6、退出mysql后，手动把mysqld服务结束掉
            4、查询用户：
                -- 进入mysql数据库
                USE mysql;
                -- 查询user表
                SELECT * FROM USER;

                -- 通配符:%表示可以在任意主机名使用用户登录数据库

        2、权限管理：
            1、查询权限：
                SHOW GRANTS FOR '用户名'@'主机名';     -- USAGE:可登录
            2、授予权限：（权限列表用逗号隔开）
                grant 权限列表 on 数据库.表名 to '用户名'@'主机名';---->可参考root权限
                GRANT SELECT ON db3.account TO 'zhangsan'@'localhost';

                -- 给指定用户授予所有权限，在任意数据库任意表上
                GRANT ALL ON *.* TO '用户名'@'主机ip';
            3、撤销权限：
                REVOKE 权限列表1,权限列表2... ON 数据库.表名 FROM '用户名'@'主机名';
                撤销指定权限REVOKE SELECT ON db3.account FROM
'zhangsan'@'localhost';
                撤销所有权限：REVOKE ALL ON *.* FROM 'zhangsan'@'localhost';
```

mysql开启binlog

```
[mysqld]

# log_bin 是指定binlog的日志文件放到哪里(data下mysql_bin为前缀的文件)
log_bin = mysql_bin
# binlog-format 有三种形式的，MIXED 会把执行的语句也记录下来，ROW 和 MIXED 的对比(对于插入
一条记录的对比)
binlog-format=Row
# 指定一个不能和其他集群中机器重名的字符串，如果只有一台机器，那就可以随便指定了
server-id=202009
```