Sql语句的分类

1. 数据库查询语言（DQL）,代表关键字select
2. 数据库操作语言（DML）,代表关键字：insert、delete、update
3. 数据库定义语言（DDL）,代表关键字create、drop、alter
4. 事物控制语言（TCL）,代表关键字：commit、rollback
5. 数据控制语言（DCL）,代表关键字：grant、revoke

DML和DDL的区别：DML是修改数据库表中的数据，DDL是修改表的结构

Sql条件查询中where子句用到的运算符



数据库中的三种主要数据类型

1. 数值型：可以有小数，也可以没有小数
2. 字符串型：可以表示一个或多个字符
3. 日期型：表示特定日期

注：在使用字符串类型数据和日期类型数据时必须用单引号或双引号括起来 where name=”daruan”

对于字符型的数据，是严格区分大小写的

Oracle在任何平台上都是严格区分字符型数据大小写的

Mysql在linux、unix平台上严格区分大小写，但是在window平台上不区分字符型数据的大小写（where name=”daruan” 和where name=”DARUAN”查询结果是一样的），我们可以使用BINARY关键字解决这个问题WHERE BINARY NAME=”daruan”

使用between and作为查询条件

作用于数值型数据和日期型数据的时候，是包含两个边界值的，可以理解为闭区间

作用于字符型数据的时候，包含第一个边界值，但是不包含第二个边界值，左闭右开（当我们需要查询名字首字母在’A’-’F’之间的员工信息，正确语句如下where name between ‘A’ and ‘G’）

使用and和or作为查询条件时，and和or的优先级：先执行and，后执行or

使用in作为查询条件，in表示包含的意思，可以使用or来替代in

Like作为查询条件进行模糊查询，即查询条件不是很明确，作用于字符型数据，配套的有两个通配符，%表示0个或任意个字符；\_表示一个任意字符

如果name字段中有通配符，需要用转义符\,where name like “%\%%”

使用order by子句给查询结果排序，可按照单个字段也可按照多个字段排序，默认按照升序排序ASC,也可以指定按降序DESC排序

数据库处理函数：

数据库函数类似于java中的方法，可以接收一个或多个参数，在函数内部完成计算， 并最终返回计算的结果

使用函数的注意事项：

1. 使用函数的时候是把查询结果当做参数输入给函数，所以函数的执行结果并不会影 响表中的数据
2. 函数并不是通用的，如有的函数在mysql中可用，但在orcal中不可用

函数的分类：

1. 单行函数：每次将单条纪录当做参数输入给函数，得到单行纪录对应的单行结果
2. 多行函数：每次将多条纪录当做参数输入给函数，得到多行纪录对应的单行结果

如：查询员工最高工资，纪录14条，我们要把14个工资一次性输入给函数，最终 得到14个工资对应的一个最高工资

单行函数：lower(str), upper(str), substr(str,index1,length) , length(str)

Trim(str)去除字符串型数据的首位空格，但不会去除中间的空格

Round(num,0)对num四舍五入，精确到小数点后0位即精确到个位，-1即十位

Rand()产生0-0.9999之间的随机小数

Ifnull(x,y)如果x为null，则返回y（注：算数运算中如出现null，结果肯定是null）

Case when then when then end函数，执行分支语句查询

str\_to\_date(str,format) 字符数据转为日期数据

date\_format(date,format) 日期转换为字符串

Now() 获取当前时间点的日期数据

多行函数：（每次将多条纪录当做参数，输入给函数，得到多条纪录对应的单行结果；多行 函数也称为分组函数，也称为组函数，也称为聚合函数）

多行函数的注意事项：

1. 多行函数会自动的忽略空值（null），不需要手动用WHERE排除空值
2. 多行函数不能出现在WHERE子句中

Select \* from user where sal < avg(sal) 查询工资低于平均工资的人，这种写法错误

1. 多行函数不能嵌套
2. 使用组函数的时候，select后的字段都必须在组函数中，多个组函数可以一起使用， 如果字段不在组函数中，查询虽然报错但结果肯定不对

sum()对数值型数据求总和

count()查询纪录总数的函数

Count(\*) 查询所有的纪录总数

Count(字段) 查询指定字段不为null的纪录总数

AVG() 查询平均值，只能对数值型的数据

MAX() 查询最大值的MAX()函数，可使用于三种主要类型

MIN() 查询最小值的函数，与MAX()相反

DISTINCT去除查询结果中重复的纪录

1. DISTINCT可以去除单列重复的数据，也可以去除多列重复的数据
2. DISTINCT只能出现在SELECT之后，查询列表之前

分组查询：

1. 把数据按照特定的条件划分为多组，然后分别使用组函数进行查询；把数据划分为 多 少组，最终就会得到多少条结果
2. 例如把所有员工按照部门编号分组，如有三个部门则分为三组，然后查询每组的最 高工资，得到三条纪录
3. 分组查询需要使用group by子句
4. 分组的时候，可以按照单个字段分组，也可按照多个字段分组

分组查询的语句结构

SELECT 查询列表

FROM 表

GROUP BY 分组字段

分组查询的规则：

1. 出现在查询列表中的字段，要么出现在组函数中，要么出现在GROUP BY子句中， 这是首选方式
2. 或者分组字段仅仅出现在GROUP BY子句中，这种方式查询的结果不明确，尽量不 要用这种方式
3. 分析以下sql

\*SELECT DEPTNO,MAX(SAL) MAXSAL FROM EMP GROUP BY DEPTNO;查询每组最高工 资

分析：出现在查询列表中有两个字段，DEPTNO和SAL；SAL出现在组函数中，DEPTNO 出现在GROUP BY子句中，所以是符合规则的

\*SELECT MAX(SAL) FROM EMP GROUP BY DEPTNO;

分析：分组字段DEPTNO仅仅出现在GROUP BY子句中，也是符合规则的

\*SELECT DEPTNO,MAX(SAL) MAXSAL,ENAME FROM EMP GROUP BY DEPTNO;

按照部门编号分组，查询最高工资及其最高工资的员工姓名

分析：ENAME字段既没有出现在组函数中，也没有出现在GROUP BY子句中，这是 不符合规则的，这样的语句在orcal中根本无法执行，在Mysql中可以执行，但是 查询的结果是错误的

对分组之后的数据进行过滤：不能使用where子句，而是使用having子句

按照部门编号分组，查询平均工资；并且显示平均工资大于2000的纪录，但以下写法 是错误的

SELECT DEPTNO,AVG(SAL) AVGSAL

FROM EMP

GROUP BY DEPTNO

WHERE AVG(SAL) > 2000; 多行函数不能出现在where子句中

正确写法：

SELECT DEPTNO,AVG(SAL) AVGSAL

FROM EMP

GROUP BY DEPTNO

HAVING AVG(SAL) > 2000

简单查询语句分析

完整语法

SELECT 查询列表

FROM 表名

WHERE 查询条件

GROUP BY 分组字段

HAVING 查询条件

ORDER BY 排序字段

以上语句的执行顺序

1. SELECT FROM查询出全部的数据
2. 使用WHERE对查询结果进行过滤
3. 过滤后的数据使用GROUP BY进行分组
4. 分组后的数据使用HAVING再次过滤
5. 再次过滤后的数据用ORDER BY排序

注意：以上语句的写法顺序是固定的，不能改变；对数据过滤的时候，优先使用WHERE 进行过滤，这样的效率比较高

数据库中表的来源：现实中的实体对应着一个数据库表；实体中的属性对应表中的字段；一个具体的学生对应表中的一条记录

一个实体对应一张表，每个表中都要有一个特殊的字段，这个字段不能重复，不能为null，通过该字段的值，可以确定唯一的一条记录，这个特殊字段就称为主键字段

如何在数据库表中体现实体和实体之间一对多（多对一）的关系，可以按照如下套路：把两个表分成父子表，一的一方是父表，多的一方是子表，把父表的主键引到子表中作为外键，父表中的一条记录对应子表中的多条记录

外键的作用：1.子表的外键值必须要在父表的主键值的范围之内；2.外键值可以重复，可以为null

连接查询：也称为多表查询，或跨表查询，也就是从多个表中查询相关的数据

注意：多表查询的时候，使用的表越多，查询的速度就越慢

多表查询的分类：

1. 按照连接的方式分
   1. 内连接：可以查询满足一一对应关系的数据；例如这个员工的所属的部门，这个部门有所属的员工，这样的数据满足一一对应的关系，可以使用内连接查询出来
   2. 外连接：可以查询不满足一一对应关系的数据；例如有的员工没有所属部门，有的部门没有员工，这样的数据不满足一一对应的关系，使用内连接是无法查询的
2. 按照语法的年代划分
   1. SQL1992标准：FROM后面是多个表，多表之间通过逗号分隔
   2. SQL1999标准：FROM后面是一个表，通过JOIN方式连接其他的表

内连接的分类：

1. 等值连接
   1. 建立在父子表关系上，用等号来连接两个表，如查询员工信息和部门信息
      1. SQL1992语法

SELECT E.EMPNO, E.JOP,D.DEPTNO,D.DNAME

FROM EMP E,DEPT D

这个查询是错误的，如EMP中有n条记录，DEPT中有m条记录，查询出来的结果一共有n\*m条记录，这种现象称为笛卡尔积

如何避免笛卡尔积呢，即加入正确的连接条件等值连接，如下

SELECT E.EMPNO, E.JOP,D.DEPTNO,D.DNAME

FROM EMP E,DEPT D

WHERE E.DEPTNO = D.DEPTNO;

在等值连接的基础上，我们可以使用AND或再加一个WHERE子句添加查询条件

* + 1. SQL1999语法

SELECT E.EMPNO, E.JOP,D.DEPTNO,D.DNAME

FROM EMP E

INNER JOIN DEPT D ON (E.DEPTNO = D.DEPTNO)

使用内连接的时候，INNER可以省略

在等值连接的基础上，我们可以使用AND或再加一个WHERE子句添加查询条件

1. 非等值连接
   1. 两个表之间没有父子关系，用非等号来连接两个表
      1. SQL1992语法

SELECT E.EMPNO,E.ENAME,E.SAL,S.GRADE,S.LOSAL,S.HISAL

FROM EMP E,SALGRADE S

WHERE (E.SAL BETWEEN S.LOSAL AND S.HISAL);

在非等值连接的基础上，我们可以使用AND或再加一个WHERE子句添加查询条件

* + 1. SQL1999语法

SELECT E.EMPNO,E.ENAME,E.SAL,S.GRADE,S.LOSAL,S.HISAL

FROM EMP E

JOIN SALGRADE S ON (E.SAL BETWEEN S.LOSAL AND S.HISAL);

在非等值连接的基础上，我们可以使用AND或再加一个WHERE子句添加查询条件

1. 自连接
   1. 使用别名将一个表虚拟成两个表（父子表），然后在这两个表上做等值连接，如查询员工的信息及其直接领导的信息
      1. SQL1992语法

SELECT E.EMPNO ‘员工编号’,E.ENAME ‘员工姓名’,M.EMPNO ‘经理编号’,M.ENAME ‘经 理姓名’

FROM EMP E, EMP M

WHERE (E.MGR = M.EMPNO)

* + 1. SQL1999语法

SELECT E.EMPNO ‘员工编号’,E.ENAME ‘员工姓名’,M.EMPNO ‘经理编号’,M.ENAME ‘经理姓名’

FROM EMP E

JOIN EMP M

ON (E.MGR = M.EMPNO)

外连接的分类：（使用外连接查询出的数据条数大于等于内连接查询出的数据条数，数据库把不匹配的数据设为null）,员工的部门字段可能为null，部门下面可能没有员工

1. 左外连接（LEFT OUTER JOIN）OUTER可省略

SELECT XXX

FROM EMP E

LEFT JOIN DEPT D ON (E.DEPTNO = D.DEPTNO)

EMP表为左表，左外连接可以把坐表中不满足条件的数据查出来，此时查询出来的DEPT中字段设为null

1. 右外连接（RIGHT OUTER JOIN）

SELECT XXX

FROM EMP E

RIGHT JOIN DEPT D ON (E.DEPTNO = D.DEPTNO)

DEPT为右表，右外连接可以把坐表中不满足条件的数据查出来，此时查询出来的EMP中字段设为null

1. 全外连接（FULL OUTER JOIN）：可以把左右两个表中不满足对应关系的数据都查询 出来（注意：orcal支持全外连接，mysql不支持全外连接）

子查询的分类：

1.出现在WHERE子句中的子查询，用来给主查询提供查询条件而首先执行的一个查询, 主查询使用子查询的结果，子查询必须放在（）里面

例如查询工资比平均工资低的员工信息

SELECT \*

FROM EMP

WHERE EMP.SAL < (

SELECT AVG(SAL)

FROM EMP)

2.出现在FROM后面的子查询，用来给主查询提供查询数据的，把子查询查询的结果当作一个虚表，然后主查询从这个虚表中查询数

3.出现在查询列表中的子查询，功能类似于外连接的效果，了解即可

UNION:把两个查询结果合并成一个查询结果

注意：合并两个查询结果的时候，要求两个查询结果的结构必须要一致（查询字段的个数，字段的类型，字段的顺序必须要一致）

UNION两种使用形式

1. UNION:会合并相同的数据；
2. UINON ALL:不会合并相同的数据
3. 使用UNION把一个左外连接的结果和右外连接的结果合并起来，实现全外连接的效果（mysql不支持全外连接，此处为解决办法）

LIMIT：可以查询表中最前面几条记录或中间某几条记录

使用：LIMIT m,n ：m代表开始的索引(索引从0开始），n代表要查询多少条记录

注意：LIMIT只能在MYSQL和SqlServer中有效

1. 使用LIMIT查询表中最前面的几条记录
   * 1. 查询工资最高的前4个员工的信息

SELECT \* FROM EMP ORDER BY SAL DESC LIMIT 0,4(0可以省略，n默认从0开始)

1. 使用LIMIT取得中间某几条记录
   * 1. 按工资降序，查询3-8位员工的信息

SELECT \* FROM EMP ORDER BY SAL DESC LIMIT 2,6; --从2开始，连续的6条记录

1. 使用LIMIT实现分页查询（每次从数据库中查询固定数量的数据，分批次的查询）

分页查询的规则：

Pagesize:表示每页现实的数据的条数

Pageno：表示页号

确定LIMIT参数的规则：LIMIT (pageno-1)\*pagesize,pagesize;

注意：使用分页查询的时候，必须要确定LIMIT的起始下标，不能使用表达式表示m或n

DML:数据库操作语句，包括INSERT,DELETE,UPDATE

我们执行DML语句的时候，数据库返回的是受到影响的数据的条数，称为更新计数器

DDL:数据库定义语句，包括CREATE(创建表），DROP(删除表），ALTER(修改表）

我们执行DDL语句的时候，数据库返回0

创建表的时候，需要指明字段的信息，字段的信息包括：字段名称 字段类型 字段长度 字段约束

DROP TABLE STUS; 删除STUS表，会连同表中的数据一起删掉，但STUS不存在时会报错

DROP TABLE IF EXISTS STUS; 只在mysql中有效

关于DATE类型：

在不同的数据库中，日期类型的数据使用方式是不同的，如str\_to\_date()只在mysql中 有效

不同的数据库中，日期格式也不同，mysql默认格式为“年-月-日”，orcale默认格式为 “日-月-年”

以上两个原因导致不同数据库中使用日期类型的sql语句都是不同的，sql语句不能移植

所以在实际的开发中，日期类型经常会被定义为字符类型

我们可以把建表的语句和查询的语句结合起来，把查询结果中的字段当做新建表的字段，并且把查询语句的结果插入到新表中，也就是复制表

CREATE TABLE EMP\_DEPT\_INFO

AS

SELECT XXX

FROM EMP E

JOIN DEPT D ON (E.DEPTNO = D.DEPTNO)

WHERE E.SAL BETWEEN 1600 AND 3000;

如果查询语句没有查询结果，则仅将查询结果中的字段当做新建表中的字段

我们可以把一个插入语句和一个查询语句结合起来，也就是把查询语句中的结果插入表中，但是结构必须一致

INSERT INTO EMP SELECT \* FROM EMP WHERE DEPTNO = 20

Alter语句，可以在不影响表中数据的情况下，对表进行修改；对表的修改主要是对字段的修改，包括：添加字段、修改字段、删除字段

1. 添加字段，新的字段会被添加到字段的最后面

ALTER TABLE STUS ADD TEL BIGINT(11);

1. 修改字段
   * 1. 当字段中没有数据时，字段的类型和长度都是可以随便修改的

ALTER TABLE STUS MODIFY TEL CHAR(11)

* + 1. 当字段中有数据时，增大字段的长度总是可以的

ALTER TABEL STUS MODIFY SNAME CHAR(20)

* + 1. 当字段中有数据，减少字段长度，要根据该字段最大数据来决定

例如SNAME最长字段是4，我们把SNAME字段的长度减少为4是可以的

* + 1. 当字段中有数据时，修改字段的类型要看字段中已有数据能否转换为新的类型

1. 删除字段

ALTER TABLE STUS DROP TEL;

约束：

字段信息包括：字段名称 字段类型 字段长度 字段约束

什么是约束：是加在表上面的（实际是加在字段上面的）检测条件，只有符合条件的数 据才能被操作（插入操作、删除、修改）

约束分类

1. 列级约束：如果一个约束只能作用在单独某列字段上面，这就是一个列级约束
2. 如果一个约束可以同时作用在多列字段上面，这就是表级约束

创建约束的时间

1. 我们可以在创建表的时候一起创建约束
2. 也可以在表创建好后，使用ALTER语句，单独给表添加约束

约束的分类

1. 非空约束（NOT NULL）
2. 唯一约束（UNIQUE）
3. 主键约束
4. 外键约束

在一个表中可以同时有多个非空约束，多个唯一约束，多个外键约束；但是在一个表中 只能有唯一的一个主键约束

非空约束是唯一的一个列级约束

一个唯一约束可以作用在单列字段上面，也可以作用在多列字段上面，所以这是一个表 级约束

唯一约束保证字段的数据或字段的组合数据不能重复，但是可以为空，唯一约束中null 可以重复

创建约束的时候也可以使用标准方式，也就是给约束命名

创建约束的标准方式：约束关键字 + 约束名称 + 约束类型 + 作用字段

CREATE TABLE STUS(

SID INT(4),

SNAME CHAR(4),

TEL CHAR(11),

CONSTRAINT TEL\_UNIQUE UNIQUE (TEL)

)

数据库约束和数据库表一样，都是一个独立的数据库对象，保存在数据库的系统表中

数据库中所有的表都保存在INFORMATION\_SCHEMA数据库的TABLES表中

数据库中所有的约束对象都保存在INFORMATION\_SCHEMA数据库的 TABLE\_CONSTRAINTS表中

一个唯约束可以同时作用在多列字段上面，保证字段组合的数据不能重复，但是可以为 null这种方式只 能用标注方式创建

CREATE TABLE STUS(

SID INT(4),

SNAME CHAR(4),

TEL CHAR(11),

CONSTRAINT TEL\_NAME\_UNIQUE UNIQUE (TEL,SNAME)

)

一个字段上面可以有多个约束，如非空且唯一约束

CREATE TABLE STUS(

SID INT(4),

SNAME CHAR(4),

TEL CHAR(11) NOT NULL,

CONSTRAINT TEL\_UNIQUE UNIQUE (TEL)

)

主键约束：从功能上说，相当于非空且唯一约束

主键约束和手动的非空且唯一约束的区别：

1. 在一个表中，可以同时有多个非空且唯一的约束；但是在一个表中只能有一个 唯 一的一个主键约束
2. 数据库会自动为主键约束创建索引；但是数据库不会为手动的非空且唯一创建 索引

主键约束的相关名字：主键约束、主键字段、主键值

主键约束的作用

1. 主键约束保证数据不能重复且不能为NULL
2. 通过主键值，可以确定唯一的一条记录

主键约束可以作用在单列字段上面，也可以同时作用在多列字段上面，称为联合主 键

思考：表中什么样的字段应该用来做主键？

1. 我们不能用和数据有紧密联系的业务数据当做主键，因为业务数据会发生 变化
2. 应该用和数据没有联系的非业务数据当做主键，例如一个整型的数值或一个定长的字符串

创建主键约束的方式：

1. 简化方式

DROP TABLE IF EXISTS STUS;

CREATE TABLE STUS(

SID INT(4) PRIMARY KEY,

SNAME CHAR(4),

AGE INT(2)

)

1. 标准方式，给主键命名

DROP TABLE IF EXISTS STUS;

CREATE TABLE STUS(

SID INT(4),

SNAME CHAR(4),

AGE INT(2),

CONSTRAINT STUS\_PK PRIMARY KEY (SID)

)

1. 联合主键，保证字段组合的数据不能重复且都不能为空，需使用标准方式 创建

DROP TABLE IF EXISTS STUS;

CREATE TABLE STUS(

SID INT(4),

SNAME CHAR(4),

AGE INT(2),

CONSTRAINT STUS\_PK PRIMARY KEY (SID,SNAME)

)

1. 在MYSQL/SqlServer中，为了维护主键值，有一种特殊的整型数据，也就 是自增字段

DROP TABLE IF EXISTS STUS;

CREATE TABLE STUS(

SID INT(4) AUTO\_INCREMENT,

SNAME CHAR(4),

AGE INT(2),

CONSTRAINT STUS\_PK PRIMARY KEY (SID)

)

把SID指定为自增字段，默认的从1开始，按1递增，插入数据的时候不 需要插入主键的值，数据库会使用自增字段来维护主键值

插入数据的时候，自增字段会查询表中最大的主键值，然后在最大的纪录 的基础上加1，自增字段是不会回退的

外键约束：

实体和实体之间有一对一，多对一的关系，在数据库中通过外键来体现实体和实体 之间的关系，这种数据库就称为关系型数据库

父表中的主键字段或唯一字段，才能被子表引用为外键；

外键约束相关名称：外键约束、外键字段、外键值

外键约束的作用：

1. 子表的违建值必须在父表的主键值范围之内
2. 外键约束的值可以重复，可以为NULL

在一个表中，可以同时有多个外键约束，一个外键约束可以作用在单列字段上，也 可以同时作用在多列字段上面

创建外键约束时，先创建父表，再创建子表

--创建父表

CREATE TABLE DEPT(

DEPTNO INT(4) PRIMARY KEY,

DNAME VARCHAR(10),

LOC, VARCHAR(20)

);

--创建子表

CREATE TABLE EMP(

EMPNO INT(4) PRIMARY KEY,

ENAME VARCHAR(10),

JOB VARCHAR(10),

DEPTID INT(4),

CONSTRAINT EMP\_DEPT\_FK FOREIGN KEY (DEPTID) REFERENCES DEPT (DEPTNO)

);

创建外键约束的时候，只能使用标准方式

子表的外键字段必须要引用父表的主键字段或唯一字段

在外键约束的作用下，在插入数据的时候，子表的外键值必须要在父表的主键 范围之内

注意1：在外键约束的作用下，我们不能直接更新父表的主键值，如需更新步 骤 如下

1. 必须先把子表的相关数据的外键值设为NULL
2. 更新父表的主键值
3. 把子表的相关的外键值设为新的外键值

注意2：在外键约束的作用下，在删除数据的时候，必须先要删除子表数据， 然后才能删除父表数据；或者把子表相关数据的外键设置为NULL,然后才能删 除父表相关的数据

注意3：同理，在外键约束作用下，删除表的时候，必须先删除子表，后删除 父表

为保障数据安全，在开发中对外键约束的不同处理方式

1. 在数据库中设置外键，通过外键约束保障数据的安全性
2. 有人认为外键会影响数据库性能，故在数据库中不设置外键，而是在java 程序中通过逻辑判断保障数据安全

级联操作：包括级联更新和级联删除

定义：在外键约束的作用下，我们不能直接操作父表数据（如更新父表的主键或删 除父表数据），但是我们可以设置级联操作，也就是在操作父表数据的时候，会连 同子表的相关数据一起操作

注：级联操作必须建立在外键约束的基础之上，先有外键约束才能做级联操作

1. 级联更新 ON UPDATE CASCADE

在更新父表数据的时候，会连同子表的相关数据一起更新

--创建父表

CREATE TABLE DEPT(

DEPTNO INT(4) PRIMARY KEY,

DNAME VARCHAR(10),

LOC, VARCHAR(20)

);

--创建子表

CREATE TABLE EMP(

EMPNO INT(4) PRIMARY KEY,

ENAME VARCHAR(10),

JOB VARCHAR(10),

DEPTID INT(4),

CONSTRAINT EMP\_DEPT\_FK FOREIGN KEY (DEPTID) REFERENCES DEPT (DEPTNO) ON UPDATE CASCADE

--首先指定外键约束，然后才能指定级联更新

);

1. 级联删除 ON DELETE CASCADE

在删除父表数据的时候，会连同子表的相关数据一起删除

--创建父表

CREATE TABLE DEPT(

DEPTNO INT(4) PRIMARY KEY,

DNAME VARCHAR(10),

LOC, VARCHAR(20)

);

--创建子表

CREATE TABLE EMP(

EMPNO INT(4) PRIMARY KEY,

ENAME VARCHAR(10),

JOB VARCHAR(10),

DEPTID INT(4),

CONSTRAINT EMP\_DEPT\_FK FOREIGN KEY (DEPTID) REFERENCES DEPT (DEPTNO) ON DELETE CASCADE

--首先指定外键约束，然后才能指定级联删除

);

切记：级联操作使用的时候一定要慎重！级联操作通常只会出现在项目测试阶段； 项目正常运行之后，是不会设置级联操作的

删除约束：

数据库约束和数据库表一样，是一个独立的数据库对象，因为约束是建立在表上面 的，所以我们删除表的时候回连同约束一起删除掉，但是删除约束的时候，并不会 影响表的结构和表中的数据

1. 删除主键约束

在一个表中只能有唯一的一个主键约束，所以在删除主键约束的时候，并不需 要指明主键约束的名称

ALTER TABLE DEPT **DROP PRIMARY KEY**

1. 删除外键约束

在一个表中可以有多个外键约束，所以删除的时候，需指明外键约束名称

ALTER TABEL EMP **DROP FOREIGN KEY** EMP\_DEPT\_FY(创建表的时候指定的名称)

1. 删除唯一约束

同理需指定唯一约束名称，也就是创建唯一约束的时候，需要使用标准方式指 定唯一约束名称

ALTER TABLE DEPT **DROP INDEX** DNAME\_UNIQUE

1. 删除非空约束

要用MODIFY方式，把DNAME字段修改为可以为NULL，也就是删除了非空约 束

ALTER TABLE DEPT MODIFY DNAME VARCHAR(10) NULL;

Mysql存储引擎

1. 存储引擎是MySql特有的，其他数据库没有
2. 存储引擎的本质：不同的存储引擎，底层采用不同的方式来存储数据，MySql数据 管理系统支持很多种引擎，可以使用show engines\G查看当前服务器有多少存储引 擎
3. Mysql支持9中存储引擎，默认的存储引擎是InnoDB，InnoDB的最大特点是支持 事物的完整处理，支持外键及引用的完整性，包括级联更新和级联删除

事务：

定义：把多个相关的操作捆绑成一个整体，要么全部都成功，要么全部都失败

特点：

1. 一个最小的不可再分的工作单元
2. 通常一个事务对应一个完整的业务，如银行转账业务
3. 一个完整的业务需要批量的DML（insert、update、delete）语句完成
4. 事务只和DML语句有关系，或者说只有DML语句才有事务
5. 批量DML语句的多少，和业务逻辑有关系，业务逻辑不通，DML语句个数不 通

事务的4个特性：

1. 原子性（atomicity）

事务是最小的单元，不可再分；

1. 一致性（Consistency）

事务要求所有的DML语句操作的时候，必须保证同时成功或同时失败；

1. 隔离性（Isolation）

一个事物不会影响其他事物的运行；

1. 持久性（Durability）

在事务完成之后，该事务对数据库所做的修改将持久的保存在数据库中，并不 会被回滚

事务中的概念

开启事务 start transaction

结束事务 end transaction

提交事务 commit transaction

回滚事务 rollback transaction

事务开始的标志

任何一条DML语句执行，标志事务的开启

事务结束的标志：提交或者回滚

提交：成功的结束，将所有DML语句操作的纪录和硬盘文件中的数据进行一次同 步；

回滚：失败的结束，将所有DML语句操作的纪录全部清空；

注意：在事务进行的过程中，未结束之前，DML语句是不会修改底层数据库文件中的 数据，只是将历史操作纪录一下，在内存中完成纪录。只有在事务结束时，而 且是成功结束时才会修改底层硬盘文件中的数据。

Mysql管理事务的方式是：自动提交

Mysql默认认为，每个DML语句都会引起一个单独的事物，这个DML语句在执行 结束时，mysql会自动提交，提交之后，事物中的对数据库的修改被永久的保存到 数据库中，其他所有的会话都可以查询到修改的数据

会话（SESSION）：客户端与服务器之间一个连续的完整的交互过程

关闭mysql的自动提交：

1. 执行语句START TRANSACTION;关闭自动提交；这是关闭本次的自动提交，事 务结束之后，新的事务又是默认的自动提交
2. 执行语句SET AUTOCOMMIT=OFF;这是关闭会话过程中的自动提交；在整个会 话中都是有效的，在新的会话中，事务又是默认的自动提交

关闭自动提交后，我们执行的第一个DML语句就会引起一个事物

在事务开始之后，只要事务没有结束，我们执行的所有DML操作，都是隶属于同一个 事务，事务中对数据的修改被保存在内存中，只有当前事物可以查询到这些修改

SET AUTOCOMMIT=OFF;默认就是关闭会话中的自动提交，等效于SET SESSION AUTOCOMMIT=OFF

事务隔离性的四个级别，从低到高，用来解决数据库的读并发问题，所谓读并发问题， 也就是多个事务同时修改相同的数据

1. 读未提交
2. 读已提交
3. 可重复读
4. 串行化

隔离级别的范围

1. 设置会话级别的隔离级别：只对当前会话有效
2. 设置全局级别的隔离级别：对所有的会话都是有效的

设置全局级别的隔离级别之后，必须要开启新的会话，新的隔离级别在新的会 话中生效

设置全局级别的隔离级别的相关语句

1. 设置全局级别的隔离级别为：读未提交

SET GLOBAL TRANSACTION ISOLATION LEVEL READ UNCOMMITED

1. 设置全局级别的隔离级别为：读已提交

SET GLOBAL TRANSACTION ISOLATION LEVEL READ COMMITTED

1. 设置全局级别的隔离级别为：可重复读

SET GLOBAL TRANSACTION ISOLATION LEVEL REPEATABLE READ

1. 设置全局级别的隔离级别为：串行化

SET GLOBAL TRANSACTION ISOLATION LEVEL SERIALIZABLE

查看全局的当前隔离级别的SQL语句

SELECT @@GLOBAL.TX\_ISOLATION 查询出来默认为REPEATABLE-READ 可重复读

隔离级别解析：

1. 读未提交：一个事物读取到了另一个事物尚未提交的数据；这种隔离级别不高： 这里读到的数据称为脏数据，这种方式也称为脏读！

脏读的后果比较严重，这个隔离级别只在理论上存在，实际中没有数据库会使 用这种隔离级别

1. 读已提交：一个事务读取到另一个事务已经提交的数据；读已提交可以避免脏 读，但是会导致另一个问题，也就是不可重复读；不可重复读即在同一个事务 中，每次读取数据的条数都是不同的，（如有三个事务，第一个事务向空表中 插入一条数据并提交，此时第二个事务读取出来一条数据，此时第三个事务也 向数据库表中插入一条数据，此刻事务二再去读就有两条数据，这就是不可重 复读），读已提交是orcale数据库的默认隔离级别
2. 可重复读：一个事物提交之后的数据，另一个事务仍然读取不到

可重复读可以避免不可重复读，在同一个事务中，每次读取到的数据的条数都 是一致的，但是会导致新的问题即幻读，幻读即读取到的数据的条数和数据库 中实际的条数已经不一致，读取到的是一幻象（如数据库中有5条数据，事务 1读取到了5条，此时事务2插入了新的1条记录并提交，事务1再去读还是 读取到5条记录，和实际数据库表中的数据条数不一致），可重复读是mysql 默认的隔离级别

1. 串行化：一个事务操作一个表的时候，会把这个表锁住（这种锁称为表级锁）， 其他的事物只能等待，只有当第一个事务提交之后，表级锁才释放，串行化可 以避免脏读，避免不可重复读，避免幻读，这是最安全的方式，但是串行化会 极大的降低数据库的处理能力，所以在实际开发中没有数据库会设置这种隔离 级别（如公司中一个程序员开启一个事物锁住了一个表，其他的程序员都得等 待第一个程序员释放锁后才能操作该表）

以上四种隔离级别都有小问题，不能很好的解决数据的读并发问题，此时可以使用行级 锁解决数据库的读并发问题

行级锁的使用步骤：

1. 关闭自动提交
2. 执行语句SELECT \* FROM EMP WHERE EMPNO IN(7369,7499) FOR UPDATE; 开启事务，并且启动行级锁，锁住7369,7499这两条记录

注意：启动行级锁的语句必须要使用主键作为查询条件，启动的才是行级 锁，否则依旧是表级所

1. 启动行级锁的目的是只有当前事务可以修改7369,7499这两条记录
2. 结束事务，释放行级锁

数据库索引：

含义：数据库索引也是一个独立的数据库对象，类似于书中的目录，可以加快数据 库的查询速度

数据库检索数据的方式有两种：

1. 全盘扫描，效率低
2. 通过索引查询数据，效率高

数据库中创建索引的方式有两种

1. 数据库自动的创建索引：数据库会自动为主键创建索引
2. 我们手动的创建索引

什么样的字段应该创建索引呢？

1. 该字段存储的数据量比较大
2. 该字段的数据很少执行DML操作
3. 该字段经常出现在查询条件中（也就是where子句）

如给EMP表中的ENAME字段创建索引

CREATE INDEX ENAME\_INDEX ON EMP (ENAME)

数据库索引也是一个独立的数据库对象，和数据库表是分别保存的，可以用一 下语句查询EMP表的索引

SHOW INDEX FROM EMP;

给ENAME字段创建索引之后，我们再使用ENAME作为查询条件，数据库会 自动的使用索引进行查询，提高查询效率

删除索引语句

DROP INDEX ENAME\_INDEX ON EMP;

删除索引，并不会影响表的结构和表中的数据

视图的使用：

含义：视图也是一个数据库对象，实际上视图就是一个命名的查询；也就是把一个查询 语句的查询结果做成一个临时表，然后从这个临时表中查询相关的数据，视图都是 和一个有效的查询语句绑定在一起的，视图的出现在from后面的子查询类似。 视图可以看做是一个表，可以像操作表一样来操作视图。

创建视图的语句：

CREATE VIEW 视图名称

AS

有效的查询语句；

删除视图：DROP VIEW IF EXISTS V\_EMP\_INFO

注意：视图实际上是不能保存数据的，对视图的DML操作实际上是对源表的操作，我 们从视图中查询数据的时候，实际上是先执行视图绑定的查询语句，然后把查询语句的 结果做成一个临时表，然后从这个临时表中查询相关数据，类似出现在from后面的子

查询

视图的作用：

1. 可以加快开发的速度，但是并不能提高查询的速度；如果一个查询结果经常被 使用到，我们就应该把这个查询语句做成一个视图，加快开发速度
2. 可以隐藏数据的真实来源（也可以对单表的列级访问做限制，如有表user，针 对user前四个字段创建一个视图a，后四个四段创建一个视图b，A用户就只 能查a，B用户只能查b）

注意：创建视图的时候，视图绑定的查询语句不能有出现在from后面的子查询

数据库设计的三范式

1. 第一范式：要求表有主键，数据库中表不能出现重复的纪录，每一个字段是原子性 不能再分；
2. 第二范式：建立在第一范式的基础之上，要求数据库中所有非主键字段完全依赖主 键，不能产生部分依赖；（严格意义上说即尽量不要使用联合主键）
3. 第三范式：建立在第二范式的基础之上，要求非主键字段不能产生传递依赖于主键 字段；

不同的实体关系在数据库中的实现

1. 多对一（1对多）：分成父子表，一的一方是父表，多的一方是字表，把父表的主键引到子表中作为外键
2. 多对多：必须要引入关系表，把一个多对多关系拆分成两个一对多关系，如有student和teacher是多对多关系，设计的时候引入一张学生老师的关系表stu\_tea\_info,此时student和stu\_tea\_info是一对多关系，teacher和stu\_tea\_info是一对多关系；
3. 一对一关系，在数据库中有两种实现方式，如妻子和丈夫表是一对一关系
   1. 把妻子表的主键作为外键引入丈夫表，在外键的基础上加一个UNIQUE约束（因为外键可重复，一个丈夫不能有多个妻子）,FK+UNIQUE
   2. 公用主键：把丈夫表的主键引到妻子表中作为外键，同时也作为妻子表的主键，PK+FK