**数据库的范式**

第一范式：一张表，确定主键，以及需要记录的每个字段无法再分。

第二范式：首先是1NF,当一张表里的部分字段不完全依赖主键，通过非主键来确定。比如分数表：学号、系名、宿舍楼、课程、分数。系名、宿舍楼可以通过学校确定。则可以分成两张表。

第三范式：首先是 2NF，另外非主键列必须直接依赖于主键，不能存在传递依赖，一个订单表【Order】（OrderID，OrderDate，CustomerID，CustomerName，CustomerAddr，CustomerCity）主键是（OrderID）。 其中 OrderDate，CustomerID，CustomerName，CustomerAddr，CustomerCity 等非主键列都完全依赖于主键（OrderID），所以符合 2NF。不过问题是 CustomerName，CustomerAddr，CustomerCity 直接依赖的是 CustomerID（非主键列），而不是直接依赖于主键，它是通过传递才依赖于主键，所以不符合 3NF。 通过拆分【Order】为【Order】（OrderID，OrderDate，CustomerID）和【Customer】（CustomerID，CustomerName，CustomerAddr，CustomerCity）从而达到 3NF。

第二范式（2NF）和第三范式（3NF）的概念很容易混淆，区分它们的关键点在于，2NF：非主键列是否完全依赖于主键，还是依赖于主键的一部分；3NF：非主键列是直接依赖于主键，还是直接依赖于非主键列。第一范式确定表的主键，需要掌握的所有字段。第二范式将只依赖于主键一部分的字段剔除，第三范式把依赖于非主键字段的字段剔除。比如台账库，剔除指标属性是符合第二范式，保留报表iD剔除报表属性是为了符合第三范式。

**sql 查询**

**Select , ,**

**From**

表1 , 表2跟子查询缩小范围

sql 1999语法： 表1 cross（笛卡尔积） natrue（自动选择字段连接） left join 表2 on（条件语句） 或者 using（相同名称字段）

**Where e.deptno(+) = d.deptno**

**Group by**

**Having avg(sal)>0**

**Order by**

多表查询的内交只需要写where a.字段 =b.字段 AND 。。。。即可。

oracle里日期可以直接用符合日期格式的字符串来表示。不用转化为日期。

sql 取反用NOT

like \_表示一个字符，%表示多个字符，字符、数字都可以

如何查询字段中是否含有中文字符：

--方法一：使用asciistr函数得出是否字段里面包含“\”，但是需要注意一个特殊字符“\”，当它出现的时候转换后的码为“\005C”

select t.\* from work\_events t where asciistr(t.descr) like '%\%'

--方法二：使用length和lengthb，如果中文的话length是一个字符，但是lengthb是两个字节。但是这个方法不适合数据库字符集非gbk的情况。

select \* from work\_events t where length(t.descr) != lengthb(t.descr);

**Mysql 表分大小写**

**Oracle 表名不分大小写**

**插入数据表，并判断主键约束。**

Mysql：

INSERT USER(id,username) VALUES(1,'张三丰') ON DUPLICATE KEY UPDATE username='王老五'

INSERT INTO table (id,a,b,c) select id,a,b,c from xxx

ON DUPLICATE KEY UPDATE a=VALUES(a),b=VALUES(b),c=VALUES(c)

Oracle：

merge into ZLS\_TJFX.TJ\_QUOTA\_DATA d

using ZLS\_TJFX.TJ\_DATA\_TMP2 t

on (d.QUOTA\_DATE=t.QUOTA\_DATE and d.QUOTA\_DEPT\_CODE=t.QUOTA\_DEPT\_CODE and d.QUOTA\_CODE=t.QUOTA\_CODE and d.RECORD\_TYPE=t.RECORD\_TYPE)

when matched then UPDATE set d.QUOTA\_VALUE=t.QUOTA\_VALUE

WHEN NOT MATCHED THEN INSERT VALUES(t.quota\_code,t.mon,t.quota\_date,t.quota\_value,t.report\_flag,t.quota\_dept\_code,t.import\_flow\_no,t.warning\_code,t.record\_type);

**备份数据表**

从一个表中选取数据，然后把数据插入另一个表中，

SELECT INTO 在SQLsever里可用,

配合 into字句，使用identity(int,1,1)来产生行号，必须把数据插入到一张表中。

--(1)

SELECT IDENTITY(int, 1,1) AS ID\_Num

INTO NewTable

FROM OldTable

--(2)

SELECT ID\_Num = IDENTITY(int, 1, 1)

INTO NewTable

FROM OldTable

--(3)

select identity(int,1,1) kk,\* into #t from T

为所有T表的数据自动加自增序列，并放入临时表#t

ORACLE里可以用

create table NEW\_TEMPTABLE as select \* from TABLE1

-- 使用insert into select 拷贝数据（注意红色部分，可以自动生成id序列值）

insert into test2(id,testname,createtime,falg)

select seq\_test.nextval,t1.testname,t1.createtime,t1.falg from test1 t1;

-- 使用 create table select 创建被拷贝数据（注意要删除test2表先）

create table test2 as select t1.id,t1.testname,t1.createtime,t1.falg from test1 t1;

mysql 里用

create table dust select \* from student;//用于复制前未创建新表dust的情况下

Create table new\_table\_name (Select \* from old\_table\_name);

insert into dust select \* from student;//已经创建了新表dust的情况下

**union用处**

当在查询两种情况，比如按照佣金字段统计是否领取佣金的人数时候，不能用groupby因为会统计每个佣金的人数。可以用分开where佣金是否有来分组汇总count，最后用union对结果合并。当然也可以decode函数计算新列，作为子查询。

**Sql的判断符**

not in （） 注意不能有空

in（） 与 =ANY() 是一样的。还有>any() 比返回的最小值要大 <any()比返回的最大值要小 >all() 比最大值都大 <all()比最小值要小。

**子查询中尽量加别名**

复杂查询 先列号查询格式，然后根据查询字段确定需要什么表，然后找到表与表之间的关联关系

建立约束 constract PK\_字段名 peri(),

**Oracle 的提交回滚与死锁**

提交事务：COMMIT

回滚：Roolback

DDL操作是隐性提交的！不能rollback

用SQL命令间接完成的提交为隐式提交。这些命令是：

ALTER，AUDIT，COMMENT，CONNECT，CREATE，DISCONNECT，DROP，

EXIT，GRANT，NOAUDIT，QUIT，REVOKE，RENAME。

在 ORACLE 中对每一个连接到数据库中的窗口，都会与数据库建立一个 Session。一个

Session 对数据库所做得修改不会马上反应到数据库的真实数据之上。是允许回滚的，当一

个 Session 提交所有操作之后，数据库才真正做出修改。

进行同样的删除操作

DELETE FROM EMP10 WHERE SAL=2450;

结果它停住了，等待第一个终端操作结束再动。----》这就是 ORACLE 死锁

**Oracle建立约束一定要单独设置别名**

在字段后面设置格式为CONSTRAINT person\_pid\_PK PRIMARY KEY(pid)

关于约束类型的命名一定要统一:

1) PRIMARY KEY:主键字段\_PK

2) UNIQUE：字段\_UK

3) CHECK：字段\_CK

4) FOREIGN KEY：父字段\_子字段\_FK

主外键关联要注意以下两点：

子表的外键在父表必须是主键

删除时应该先删除子表再删除父表

如果希望删除一个表时自动删除其子表记录则加上级联删除设定即下面红色部分。



增加/删除约束为 ALTER TABLE person ADD/DROP CONSTRAINT，非空约束直接修改表结构。ALTER TABLE person MODIFY（name 数据格式 NOT NULL）；

**Oracle的伪列。**

**rownum再数据库并不存在，再查询的时候才赋予。**

Rownum = 1 查第一条 <=10 查前十条。

分页显示查询。

Select \*

From (SELECT ROWNUM rn,empno,ename,job,sal,hiredate FROM emp WHERE ROWNUM<=10) tmp

Where tmp.rn>5

**Rowid**

Rowid虽然默认不显示但是 是实际存在的行编号，在处理重复数据的时候有用。Groupby 需要判断重复的字段，然后选择min（rowid），则可以筛选出非重复记录。然后delete操作删除not in （rowid）的记录。

**ORACLE的序列**

CREATE SEQUENCE 序列名

[INCREMENT BY n]

[START WITH n]

[{MAXVALUE/ MINVALUE n|NOMAXVALUE}]

[{CYCLE|NOCYCLE}]

[{CACHE n|NOCACHE}];

范例: 创建一个 myseq 的序列, 验证自动增长的操作

CREATE SEQUENCE myseq;

创建的序列对象在user\_sequences表中。

范例: 使用序列

INSERT INTO testseq(next,curr) VALUES( myseq.nextval, myseq.currval);

myseq.nextval 调用序列增长，myseq.currval显示目前序列数值。先执行nextval才能使用currval。

**Oracle的别名**

比如：dual是sys.dual的别名

CREATE PUBLIC SYNONYM mydept FOR scott.dept

**Oracle的视图**

CREATE [OR REPLACE] VIEW 视图名称 AS 子查询

WITH READ ONLY;控制视图有关数据不被修改。

Select \* from myview;即可调用视图。

**Oracle的物化视图**

普通视图只是一个查询方式，物化视图是存储了数据的实体表，可以提高查询数据，物化视图又三种更新方式。ON DEMAND视图“需要”被刷新了，才进行刷新(REFRESH)，ON COMMIT是说，一旦基表有了COMMIT，则立刻刷新

刷新的模式有两种：ON DEMAND和ON COMMIT。（如上所述）

刷新的方法有四种：FAST、COMPLETE、FORCE和NEVER。

FAST刷新采用增量刷新，只刷新自上次刷新以后进行的修改。

COMPLETE刷新对整个物化视图进行完全的刷新。

如果选择FORCE方式，则Oracle在刷新时会去判断是否可以进行快速刷新，如果可以则采用FAST方式，否则采用COMPLETE的方式。

NEVER指物化视图不进行任何刷新。

**变量：**

在mysql文档中，mysql变量可分为两大类，即系统变量和自定义变量。

但根据实际应用又被细化为四种类型，自定义变量分为：局部变量、用户变量。系统变量分为会话变量和全局变量。

局部变量

一、mysql局部变量，只能用在begin/end语句块中，比如存储过程中的begin/end语句块。

其作用域仅限于该语句块。

-- declare语句专门用于定义局部变量，可以使用default来说明默认值

declare age int default 0;

-- 局部变量的赋值方式一

set age=18;

-- 局部变量的赋值方式二

select StuAge

into age

from demo.student

where StuNo='A001';

二、mysql用户变量，mysql中用户变量不用提前申明，在用的时候直接用“@变量名”使用就可以了。

其作用域为当前连接。

-- 第一种用法，使用set时可以用“=”或“:=”两种赋值符号赋值

set @age=19;

set @age:=20;

-- 第二种用法，使用select时必须用“:=”赋值符号赋值

select @age:=22;

select @age:=StuAge

from demo.student

where StuNo='A001';

三、mysql会话变量，服务器为每个连接的客户端维护一系列会话变量。

其作用域仅限于当前连接，即每个连接中的会话变量是独立的。

-- 显示所有的会话变量

show session variables;

-- 设置会话变量的值的三种方式

set session auto\_increment\_increment=1;

set @@session.auto\_increment\_increment=2;

set auto\_increment\_increment=3; -- 当省略session关键字时，默认缺省为session，即设置会话变量的值

-- 查询会话变量的值的三种方式

select @@auto\_increment\_increment;

select @@session.auto\_increment\_increment;

show session variables like '%auto\_increment\_increment%'; -- session关键字可省略

-- 关键字session也可用关键字local替代

set @@local.auto\_increment\_increment=1;

select @@local.auto\_increment\_increment;

四、mysql全局变量，全局变量影响服务器整体操作，当服务启动时，它将所有全局变量初始化为默认值。要想更改全局变量，必须具有super权限。其作用域为server的整个生命周期。

-- 显示所有的全局变量

show global variables;

-- 设置全局变量的值的两种方式

set global sql\_warnings=ON; -- global不能省略

set @@global.sql\_warnings=OFF;

-- 查询全局变量的值的两种方式

select @@global.sql\_warnings;

show global variables like '%sql\_warnings%';

**游标、触发器、存储过程和存储函数**

**Mysql:**

delimiter $$　　#将语句的结束符号从分号;临时改为两个$$(可以是自定义)

默认的MySQL语句分隔符为分号 ; 。Mysql命令行实用程序也是 ; 作为语句分隔符。如果命令行实用程序要解释存储过程自身的 ; 字符，则他们最终不会成为存储过程的成分，这会使存储过程中的SQL出现句法错误

解决方法是临时更改命令实用程序的语句分隔符

**触发器：**

分析：

监视谁：ord

监视动作：insert

触发之间：之后（之前）

触发事件：update

创建语法：

delimiter //

Ceate trigger 触发器名称

Before / after （触发时间）

Insert / update /delete （监视动作）

On 表名 （监视对象）

For each row --行级触发器，mysql必须写。此处的含义是监视动作修改了多少行，后面的触发动作执行多少次。

Begin

Declare -- 声明变量

rnum int [default 18] ; 变量名 类型 可选的默认值 与建表的定义一样（python不用只需要赋值）。set 变量，给变量赋值；@变量名 ，再begin和end里面@可以不加。

Select num into rnum from goods where gid = new.gid;

If rnum < new.much then

Set new.much = rnum

End if;

Update goods set num = num –new.much where gid = new.gid; -- new表示插入的新行

……

sqlN

End//

查看触发器：

Show tiggers /G

删除触发器：

Drop trigger triggerName

**自定义函数：有且仅有1 个返回，适合做处理数据后返回一个结果。**

--创建函数，

CREATE FUNCTION 函数名(参数列表) RETURNS 返回类型

BEGIN

函数体

END

/\*

注意：

1、参数列表 包含两部分：参数名 参数类型

2、函数体：肯定会有return语句，如果没有会报错

如果return语句没有放在函数体的最后也不报错，但不建议

3、函数体中仅有一句话，则可以省略begin end

4、使用 delimiter语句设置结束标记

\*/

# 案例：根据商品名称返回商品价格

DELIMITER $

DROP FUNCTION IF EXISTS myf2$

CREATE FUNCTION myf2(proName VARCHAR(20)) RETURNS DOUBLE

BEGIN

SET @sal=0;#定义用户变量

SELECT cash\_cost INTO @sal #赋值

FROM mall\_pro

WHERE pro\_name = proName;

RETURN @sal;

END $

SELECT myf2('手表') $

调用函数：SELECT 函数名(参数列表)

查看函数：SHOW FUNCTION STATUS;

删除函数：DROP FUNCTION [IF EXISTS] genPerson$$

**存储过程：** 类似于 python 中的函数。封装一段复杂sql代码,存储再数据库中，方便重复调用。

基本结构

DELIMITER // //定义新的语句分隔符为//

CREATE PROCEDURE productpricing([in/out/inout] 参数名 参数类型)

BEGIN

SELECT Avg(prod\_price) AS priceaverage FROM products;

END //

DELIMITER ; //改回原来的语句分隔符为 ;

除\符号外，任何字符都可以作为语句分隔符

create procedure outtest(in a int, in b int, out c int)

begin

select a+b into c;

-- select c;

end //

out参数是和以上两种参数的赋值方法不一样，被out的参数不需要声明和定义，只需要用out指定就可以了，在最后输出的就是该out参数。

使用productpricing存储过程：CALL productpricing(18，@name); 输出变量位置需要加一个变量来获取输出。如果是inout型变量，则先set @currage = 18，然后在把@currage传进存储过程。

查看已有的procedure： show procedure status;

存储函数：

不能有out参数，调用不用call，必须有return。

**游标：**

**用于一次取一行，定义在begin end之间**

--定义变量

declare testrangeid BIGINT;

declare versionid BIGINT;

declare done int;

--创建游标

declare cur\_test CURSOR for

select id as testrangeid,version\_id as versionid from tp\_testrange;

--申明一个句柄事件，当不能fetch不到数据的时候，设置done变量=1，可以设置done=1时结束循环。

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done=1; 踩空设置done=1，但是继续fetch

DECLARE EXIT HANDLER FOR NOT FOUND SET done=1;--如果没取到数据设置done=1，并且不会再fetch。

--打开游标

open cur\_test;

--取游标中的值

FETCH cur\_test into testrangeid,versionid;

--释放游标

CLOSE cur\_test;

**索引：**

Oracle：

索引就是对字段建立二叉树，以加快查询效率

CREATE INDEX emp\_sal\_ind ON scott.emp(sal);

**Sprk连接在DBI中用hive语句查询，hive函数可以用在sparklyr里面**

sparR中的dataframe的格式：

int代表整数

dbl代表浮点数或者实数

chr代表字符向量或者字符串

dttm代表日期-时间

lgl代表逻辑向量，只含TRUE和FALSE

fctr代表因子，R用它来代表含固定可能值的分类变量

date代表日期

R

n\_distinct(...,na.rm=FALSE)