# **关系数据库标准语言sql**

## 修改基本表语句

alter table <表名>

[add <新列名><数据类型>] —给某表添加新列名

eg: alter table student add sname char(5);

[drop <列名> [cascade|restrict]] —删除某表列名

eg: alter table student drop sname cascade

[alter column <列名><数据类型>] —更新某表列名字段类型

eg: alter table student alter column sname varchar(5)

## 删除基本表语句

drop table <表名>[cascade|restrict(默认)]

eg:

create view is\_student //创建视图

as

select sno,sname,sage from student where sdept=’is’

drop table student restrict //会报错因为该表下有视图

drop table student cascade//会直接删除包括表上视图

注意：

drop、truncate、 delete区别

drop直接删掉表。

truncate删除表中数据，再插入时自增长id又从1开始。

delete删除表中数据，可以加where字句。

## 索引语句

create [unique][cluster] index <索引名> —建立索引

on <表名>(<列名>[<次序>],<列名>[<次序>]…………)

eg: create unique index stusno on student(sno);

alter index <旧索引名> rename to <新索引名> —修改索引

eg: alter index stusno rename to mystu;

drop index <索引名> —删除索引

eg: drop index mystu;

## 数据查询操作

1.select [all|distinct] <目标列表达式> (as 别名) from <表名或者视图名>

[where <条件表达式>]

[group by <列名> [having <条件表达式>]]

[order by <列名> [asc(默认升序)|desc]]

2.补充where条件所需关键字：

确定范围：between and, not between and （包括上下限）

确定集合：in, not in

字符匹配：like, not like

空值：is null, is not null

3.字符匹配

[not] like‘<匹配串>’[escape ‘<换码字符>’]

eg: select cno from course where cname like ‘db\\_lis’escape ‘\’;

escape ‘\’表示为换码字符之后的‘\_’就不会被认为是通配符

注：where子句是不能用聚集函数作为条件表达式，聚集函数只能用于group by中having子句

eg:select sno from sc group by sno having count(\*)>3//查询选修课大于3

4.带有any或者all关键字的子查询

eg: select sname form student sage<any(select sage from student where sdept=’cs’) and sdept!=’cs’//查询比计算机任意年龄都小的学生

注：=any等价于in，<any等价于<max，<>all等价于not in ，<all等价于<min

5.带有exists和not exists的子查询

//查询选修所有课程的学生姓名->反向思维查询没有一门课程是他不选修的

eg：select sname from student where not exists(select \* from course where not exists (select \* from sc where son=student.sno and con=course.cno))

//查询至少选修了学生20128989选修课的全部课程学生号码。

select distinct sno from sc sc1 where not exists

(select \* from sc sc2 where sno=’20128989’and not exists

(select \* from sc sc3 sc1.sno= sc3.sno and sc2.cno=sc3.cno));

注：exists: 子查询有数据则返回外查询为真否则为假

not exists: 子查询为空时则返回外查询为真否则为假

说明：通过from子句生成派生表时，as关键字可以省略，但必须为派生表指定别名，如果子查询中有聚合函数，则派生表需指定属性列。

6.集合查询：集合操作主要包括并操作 union ,交操作intersect和差操作except

//查询计算机专业年龄小于19岁的学生姓名

eg: select sname from student where sdept=’cs’union

select sname from student where sage<19;

//找出每个学生超过他自己选修课程平均成绩的课程号

select sname from sc,(select sno,avg(grade) from sc group by sno)

as avg\_sc(sno,grade) where sc.sno=avg\_sc.sno and sc.grade>avg\_sc.grade

## 数据更新操作

1.插入数据

insert into <表名>[<属性列>,<属性列>] values (<常量>，<常量>)

注：属性列可省略，valuse不可省略，常量需用null代替

2.插入子查询结果（可省略values）

Insert into<表名>[<属性列>，<属性列>] 子查询；

Eg: insert into dep\_age(sno,sage) select sno,avg(sage) from student group by sno;

3.修改数据

update <表名> set <列名>=<表达式> where <条件>

4.删除数据

delete from <表名> [where <条件>]

## 视图

视图：是一种虚拟的表，具有和物理表相同的功能，它可以对试图进行增删改查操作，试图通常是一个表或者多个表的行或者列的子集，对视图的修改会不影响基本表。它使得我们获取数据更容易，相比多表查询。

1.建立视图

create view <视图名>[<列名>，<列名>] as <子查询> [with check option];

注：with check option的作用表示对视图进行update,delete,insert,操作时，满足where条件表达式。

eg:create view stu\_vw(sno,sage,sname) as select sno,sage,sname from student where sage>16

说明：视图不仅可以建立在一个或者多个基础表上，也可以建立在一个或者多个视图上。

eg:create view is\_s2 as select sno,sname,sage from is\_s1 where sage<19

//此处不可省略了视图的列名

eg:create view s\_c (sno,gavg) as select sno,avg(grade) from sc group by sno;

2.删除视图

drop view <视图名> [cascade]

cascade的作用是将删除该视图以及由该视图导出的所有视图数据

3.查询视图

select sno,sname,sage from is\_s2 where sname=’小花’

4.更新视图

update is\_s1 set sname=’李婷’ where sno=’20210208’

->update student set sname=’李婷’ where sno=’20210208’and sage<19

注：若定义视图时加with check opion 则使用该视图时会保留where条件

并不是所有视图都可以更新：

1>若视图是由两个以上基本表导出的，则此视图不允许更新

2>若视图的字段来自字段表达式或常数，则不允许对此视图执行insert和update操作，但可执行delete

3>若视图的字段来自聚集函数，则此视图不允许更新

4>若视图定义中含有group by子句，则此视图不允许更新

5>若视图定义中含有distinct短语，则此视图不允许更新

6>若视图定义中有嵌套查询，并且内查询的from字句中涉及的表也是导出该视图的基础表，则此视图不允许更新。

# **数据库安全性控制**

## 授全—grant

grant<权限> on <对象类型> <对象名> to <用户> [with grant option]

eg：grant select on table student to yyk

赋予全部权限

eg: grant all privileges on table student,course to yyk

## 收回—revoke

收回语句已经授予用户的权限

revoke <权限> on <对象类型><对象名> from <用户>[cascade|restrict]

eg:revoke select on table student from yyk

## 创建数据库模式权限

create user <username> [with] [dba|resource|connect(默认)]

# **数据库完整性**

## 创建表

create table student

(

sno char(9) not null,

cno char(4) not null,

grade smallint,

primary key(sno,sno),

froeign key (sno) references student(sno),

foreign key (cno) references course(cno)

);

## 完整性约束命名子句

1.创建表中完整性约束

constraint <完整性约束条件名> <完整性约束条件>

完整性约束条件包括 not null,unique,primary key, foreign key,check等。

eg:建立学生登记表student，要求学号在9000和9999之间，姓名不能为空，年龄不能小于30，性别只能是“男”或者“女”

create table student(

sno numeric(6) constraint c1 check(sno between 9000 and 9999),

sname char(5) constraint c2 not null,

sage int constraint c3 check(sage<30),

sex char(2) conctraint c4 check(sex in (‘男’，‘女’))

default sex=’男’，

constraint stu\_key primary key(sno)

);

2.修改表中的完整性约束

eg：去掉student表中对性别限制

alter table student drop constraint c4;

eg：添加对student表中性别限制

alter table student add constraint c4 check(sex in(‘男’，‘女’))；

## 创建触发器

create trigger <触发器名> /\*每当触发事件发生时，该触发器被激活\*/

{before|after}<触发事件> on <表名> /\*指明触发器激活的时间是在执行触发器事件前或后\*/

referencing new|old row|table as <变量>/\*referencing指出引用的变量\*/

for each {row|statement} /\*定义触发器的类型，指明动作体执行的频率\*/

[when<触发条件>]<触发动作体> /\*仅当触发条件为真时才执行触发动作体\*/

注：（each row:触发动作体多次 each statement：触发动作体一次）

eg:当对表sc中的grade属性修改时，若分数增加到10%,则将此次操作记录到另一个表中sc\_u(sno,cno,oldgrade,newgrade)中，其中oldgrade是修改前的分数，newgrade是修改之后的分数

create trigger sc\_tr

after update of grade on sc

referencing old row as oldtable,new row as newtable

for each row

when(newtable.grade>=1.1\*oldtable.grade)

insert into sc\_u(sno,cno,oldgrade,newgrade)

values(oldtable.sno,oldtable.cno,oldtable.grade,newtable.grade);

eg:将每次对表student执行插入操作所增加的学生个数记录到student\_insertlog中

create trigger student\_cout

after insert on student

referencing new table as deleta

for each statement

insert into student\_insertlog(number)

select count(\*) from deleta

eg:定义一个before行级触发器，为教师表teacher定义完整性规则“教授的工资不低于4000元，如果低于4000元，自动改为4000元”

create trigger insert\_or\_update\_sal

before insert or update on teacher

referencing new row as newtable

for each row

begin

if(newtable.job=’教授’)and(newtable.sal<4000)

then newtable.sal:=4000;

end if;

end;

注：在更新操作表前就按照触发器规则调整教授工资

## 删除触发器

drop trigger <触发器名> on <表名>

## 创建存储过程

create or replace procedure <过程名>[参数1,参数2...]

as <过程化sql块>

## 执行存储过程

call/perform procedure <过程名>[参数1,参数2...]

## 修改存储过程

alter procedure <过程名1> rename to <过程名2>

## 删除存储过程

drop procedure <过程名1>();

# **关系数据理论**

第一范式（1NF）：指数据库表的每一列都是不可分割的基本数据项，同一列中不能有多个值，（例如一个部门实体还包括部门成员属性字段）即实体中的某个属性不能有多个值或者不能有重复的属性。

第二范式（2NF）：要求实体的属性完全依赖于主关键字。（例如员工信息表中加上了员工编号（emp\_id）列，因为每个员工的员工编号是惟一的，因此每个员工可以被惟一区分。这个惟一属性列被称为主关键字或主键、主码）

第三范式（3NF）：要求一个数据库表中不包含已在其它表中已包含的非主关键字信息。（例如，存在一个部门信息表，其中每个部门有部门编号（dept\_id）、部门名称、部门简介等信息。那么在的员工信息表中列出部门编号后就不能再将部门名称、部门简介等与部门有关的信息再加入员工信息表中。如果不存在部门信息表，则根据第三范式（3NF）也应该构建它，否则就会有大量的数据冗余。）

第四范式（4NF）：就是限制关系模式的属性之间不允许有非平凡且非函数依赖的多值依赖。

注：在任何一个关系数据库中，第一范式（1NF）是对关系模式的基本要求，不满足第一范式（1NF）的数据库就不是关系数据库。

1NF：消除非主属性对码的部分函数依赖

2NF：消除非主属性对码的传递函数依赖

3NF：消除非主属性对码的部分和传递函数依赖

BCNF：消除主属性对码的部分和传递函数依赖

4NF：消除非平凡非函数依赖的多值依赖

# **数据库编程**

## 流程控制语句

1.if语句

if <condition> then

sequence\_of\_statement;

end if;

eg: //创建sql执行块

begin

if(job=’教授’) and (sal<4000)

then sal:=4000;

end if;

end;

if()

2.if-then语句

if <condition> then

sequence\_of\_statement;

else

sequence\_of\_statement;

end if;

## 循环控制语句

1. 最简单的loop循环语句

loop

sequence\_of\_statement;

and loop;

1. while-loop循环语句

while <condition> loop

sequence\_of\_statement;

end loop;

1. for-loop循环语句

for count in[reverse] bound1..bound2 loop

sequence\_of\_statement;

end loop;

说明：将count设置为循环下界bound1，检查是否小于上界bound2;当指定reverse时则将count设置为上界bound2，检查count是否大于下界bound1。

# **事务的ACID特性**

事务：用户定义的一个操作序列，这些操作要么全都执行要么全部不执行，是一个不可分割的工作单位。

事务具有四个特性：原子性，隔离性，一致性，持续性。

原子性：要么都做，要么都不做；

隔离性：事物之间互不干扰；

一致性：执行结果是从一个一致性状态变化到另一个一致性状态；

持久性：事务执行的结果对于数据库是永久性；

关系完整性：实体完整性，参照完整性，用户自定义的完整性；

# **数据库相关简答题**

1.数据库故障

事务故障，介质故障（磁盘丢失），系统故障（因os，dbms故障或断电故障）。

2.恢复策略（通过备份和日志进行恢复）

1>.定期转储整个数据库；

2>.建立事务日志，要求事务的每一个修改操作都要写入日志文件，在修改数据库时，要求先将记录写入日志磁盘文件中之后，才能正式开始。undo撤销，redo重新做。undo比redo需要更多io操作。redo延迟更新。

3.数据库的并发访问会带来哪些数据不一致的问题？

丢失修改，不能重复读和读脏数据。

4.解决并发访问不一致问题方案：

通过并发控制来保证隔离性。并发控制可以通过封锁（排他锁，共享锁）来实现，但是封锁操作需要用户自己控制，相当复杂。数据库管理系统提供了事务的隔离级别，让用户以一种更轻松的方式处理并发一致性问题。

5.叙述等值连接与自然连接的区别和联系

自然连接是除去重复属性的等值连接。自然连接一定是等值连接，反之不一定。等值连接要求相等的属性分量，但不一定是相同的属性，而自然连接必需是相同的属性。等值连接不把相同的属性去掉，而自然连接去掉。

6. 由全码组成的关系模式，最高可以达到的模式为（BCNF）

7. 对关系模式进行分解时，要求保持函数依赖，最高可以达到（3NF）

8. 关系模式中的属性全部是主属性，则的最低范式必定是（3NF）

9. 关系模式R中的属性全部是主属性，则R的最高范式是（5NF）

10. 如果一个关系模式R是1NF，并且关系的每个决定因子都是候选码，那么R至少应该是(BCNF)范式

11.一个关系模式如果满足了(BCNF)范式，那么在函数范围内它以实现彻底分离

12.能够消除多值依赖引起的冗余的是4NF.

7.事务遵守（两段锁协议）是可串行化调度的充分条件。

8. 数据抽象的内容是（分类、概括、聚集）。

9.数据：描述事物的符号记录称为数据

10.数据库：是长期存储在计算机内的有组织的可共享的数据集合

11.数据库系统：包括数据，数据库，数据库管理系统，应用系统以及数据库管理员组成

12.数据库管理系统：指位于用户和操作系统之间的一层数据管理软件（系统软件）

19. 数据库管理系统的工作不包括（为定义的数据库提供操作系统）

13.在关系代数中，五种基本运算是指（并、差、笛卡儿积、投影、选择）

14.在关系代数中，传统的集合运算是指（并、差、、笛卡儿积）

14. 视图是DBS采用的(完整性措施)

15.sql语言支持关系数据库的三级模式结构，其中外模式对应于(视图)和基本表，模式对应(基本表)，内模式对应（存储文件）

16.关系模型中可以有三类完整性约束（实体完整性，参照完整性，用户自定义完整性）

19. 实体完整性是针对(主码)而言的，参照完整性是针对(外码)而言的。

17. 外模式/模式映像——数据的逻辑独立性。

18. 模式/内模式映像——数据的物理独立性。

**11>关系数据库都有那些操作,特点是什么?**

◇查询:选择、投影、连接、除、并、交、差

◇数据更新:插入、删除、修改

◇关系操作的特点:集合操作方式,即操作的对象和结果都是集合。

**12>死锁是什么?**

指多个有关进程由于争夺资源而造成的一种僵局,在无外力的情况下这些进程都将无法再向前推进的状态

**13**. 关系数据库用**（表格数据）**来表示实体之间的联系。

**14**. 数据库系统的特性不包括以下（**数据加工**），支持某种数据模型

15. 数据库系统不仅包括数据库本身，还要包括相应的硬件、软件和**（各类相关人员）**

**16.** 数据库管理系统能实现对数据库中数据表、视图等对象的定义、修改和删除，这类语言称为**（数据定义语言（DDL））**

**18.**用来对数据库中的数据进行查询、插入、删除和修改的语句，这类语言称为**（数据操纵语言(DML)）**

**17.** 在数据库系统中，通常用三级模式来描述数据库，其中**（外模式）**是用户与数据库的接口，是应用程序可见到的数据描述，**（模式）**是对数据整体的**（逻辑结构）**的描述，而**（内模式）**描述了数据的**（物理结构）**

**19.**SQL语言中的“视图（VIEW）”对应于数据库系统三级模式结构中的**（外模式）**

**18.** 数据库的网状模型应满足的条件是**（允许一个以上的结点无双亲，也允许一个结点有多个双亲）**

**19.**sql语言是一种高度**（非过程化）**语言。

**21**. 在数据库设计中数据流图（DFD）和数据字典(DD)主要用来描述结构化方

法中的**（需求分析）**阶段的工具。

**22.** SQL的集合处理方式与宿主语言单记录的处理方式之间用**（游标）**来协调。

**23.** 下面关于关系数据库范式的描述中，错误的是**（关系满足的范式越低，则查询操作的代价就越高）**

**24.** 关系模型和网状模型的数据结构以及层次数据结构分别是**（二维表）和(有向图)和（有向树）**

**25.** 数据模型是由**（数据结构）、（数据操作）、和（完整性约束）**三部分组成的

**26.** 关系数据语言可以分为三类：**（sql语言）**、**（关系代数语言）**和关系演算语言

**27.** 在关系数据库的规范化理论中，在执行“分解”时，必须遵守规范化原则：保持原有的函数依赖和**（无损链接）。**

**28.** 以下是信息世界的模型，且实际上是现实世界到机器世界的一个中间层次的是**(概念模型)**

**29.** 关系代数是一种关系操纵语言，它的操作对象和操作结果均为**（关系）。**

**30.** 关系数据库中，实现表与表之间的联系是通过**（参照完整性规则）。**

**31.** 采用数据库镜像技术，主要是为了有效解决**（介质故障）**的问题。

**32**．在数据库概念结构设计中，各分E-R模型之间的冲突主要有：属性冲突、**（命名冲突）**和结构冲突。

**33**. SQL中，表有三种：（**基本表-实表**）、（**视图-虚表**）和（**导出表-临时表**）

**34.** 二级封锁协议除了可以防止丢失修改，还可以防止**(读脏数据)**

**35.**数据库管理系统是通过**（日志和锁机制）**保持ACID特性

**36.**关系规范化中插入操作异常是指**（应该插入的数据未被插入）**

**37.**在关系数据库系统中，设计关系模式是**（逻辑结构）**任务

**38.**关系数据库中可命名的最小数据单位是**（属性名）**

**39.**在优化策略中，正确的策略是**（尽可能早的执行选择操作）**

**40.**物理优化就是要选择高效合理的操作算法或**（存取路径）**以获取优化查询

**40.**聚合函数中不忽略null的是**（count(\*)）**

**41.**rdmbs查询处理可分为**（查询分析，查询检查，查询优化，查询执行）**四个阶段。

**42.简述两段锁协议并说明两段锁协议和事务可串行化调度的关系。**

答：所谓两段锁协议是指所有事务必须分两个阶段对数据项加锁和解锁。

1、在对任何数据进行读、写操作之前，首先要申请并获得对该数据的封锁；

2、在释放一个封锁之后，事务不再申请和获得任何其他封锁。如果事务中所有的加锁操作都在事务的第一个解锁操作之前进行，那么这个事务是遵循两段锁协议的。如果一个调度中的每个事务都遵循两段锁协议，那么该调度必然是冲突可串行化的。事务遵守两段锁协议是可串行化调度的充分条件，而不是必要条件。

**43．简述可串行化调度和正确调度。**

答：多个事务的并发执行是正确的，当且仅当其结果与按某一次序串行执行它们的结果相同，我们称这种调度策略为可串行化的调度。可串行性是并发事务正确性的准则，按这个准则规定，一个给定的并发调度，当且仅当它是可串行化的，才认为是正确调度。

**44.** 数据库逻辑设计的主要任务是（**创建数据库说明**）

**45.** 数据库的概念模型也称为**（全局性逻辑数据组织）**

**46.** 数据库管理系统中用于定义和描述数据库逻辑结构的语言称为**（数据库模式描述语言）**

**47.** 缓冲区是由控制信息和若干**（定长页）**组成的

**48.**