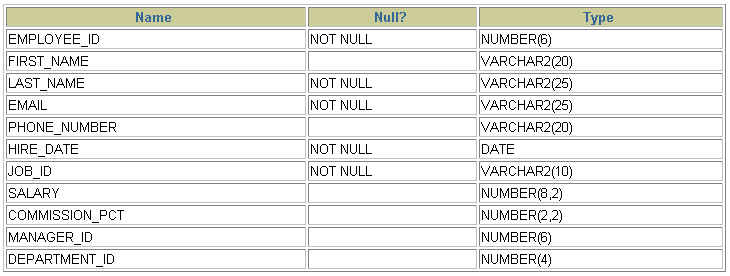
1. **管理表结构（DDL）**
   1. **显示表结构**



* 1. **数据类型**

|  |  |
| --- | --- |
| **数据类型** | **描述** |
| **VARCHAR2(*size*)** | **可变长字符数据** |
| **CHAR(*size*)** | **定长字符数据** |
| **NUMBER(*p*,*s)*** | **可变长数值数据** |
| **DATE** | **日期型数据** |
| **LONG** | **可变长字符数据，最大可达到2G** |
| **CLOB** | **字符数据，最大可达到4G** |
| **RAW and LONG RAW** | **原始的二进制数据** |
| **BLOB** | **二进制数据，最大可达到4G** |
| **BFILE** | **存储外部文件的二进制数据，最大可达到4G** |

* 1. **新建表（Create Table）**
     1. **建表规则**
* 必须以字母开头
* 必须在1-30个字符之间
* 只能包含A-Z a-Z 0-9 \_ $ 和#
* 不能和其他的数据库对象重名
* 不能是oracle的保留字
* Oracle默认存储都是大写



* + 1. **CREATE TABLE 语句语法**



* 必须指定:
  + 表名
  + 列名, 数据类型, 数据类型的大小
* 要求必须具备以个资源才可以建表:
  + CREATE TABLE权限
  + 存储空间

--创建自己的用户

*--创建自己的用户*

create user guigu identified by "guigu"

*--完成对用户的赋权*

grant connect,resource to guigu

create table student(

sid number(10),

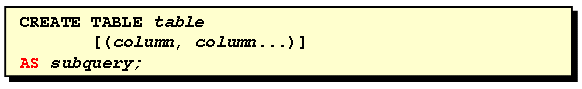
sname varchar2(50),

address varchar2(30),

bithday date default sysdate

)

* + 1. **使用子查询创建表**
* 使用 AS *subquery* 选项，**将创建表和插入数据结合起来**



* 指定的列和子查询中的列要一一对应
* 通过列名和默认值定义列
* **使用子查询创建表举例**

create table myemp

as

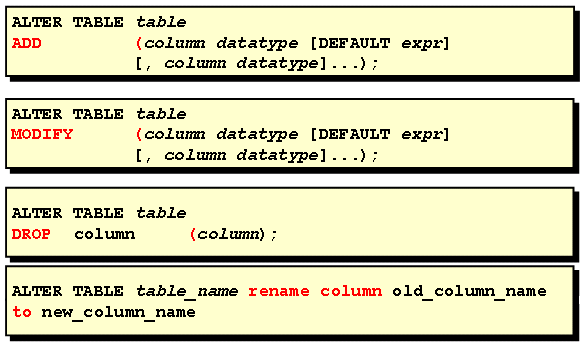
select e.employee\_id,e.first\_name ,e.salary,e.hire\_date

from employees e

where e.department\_id=50

select \* from myemp

* 1. **修改表结构（Alter Table）**
     1. **语法说明**



* + 1. **示例**
       1. **追加一个新列**

alter table student add (password varchar2(50),age number(4) default 0 )

alter table student add(last\_name number(10))

* + - 1. **修改一个列**

*--修改一个列 这里的modify 只能修改列的类型不能修改列的名字*

alter table student modify(last\_name varchar2(30))

* + - 1. **删除一个列**

alter table student drop column last\_name

* + - 1. **修改列的名字**

alter table student rename column password to pwd

* 1. **清空表（Truncate）**

清空表是指清空表中的所有数据 释放表空间

*--清空表中的所有数据，但是表结构依然保留*

select \* from myemp

truncate table myemp

* 1. **删除表（Drop Table）**

表中的数据和表结构都被删除

当前所有的事务都会被提交

所有相关的索引都被删除

*--删除表 不仅删除表中的数据 连表结构一并删除*

drop table myemp

* 1. **改变对象的名称**
* 执行RENAME语句改变表, 视图, 序列, 或同义词的名称
* 必须是对象的拥有者

*--1.7. 改变对象的名称*

rename student to stu

select \* from stu

* 1. **约束**
     1. **约束说明**

约束是表一级的约束

如果存在依赖关系，约束可以防止错误的删除数据

约束的类型分为五类

* NOT NULL 非空
* UNIQUE 唯一
* PRIMARY KEY 主键
* FOREIGN KEY 外键
* CHECK 检查
  + 1. **非空约束**

*--not null 非空约束 列级别的约束 只能定义在列级别 只能定义在单个字段上 不能同时定义多个字段*

*---每个约束都有自己的名字，如果不指定约束的名字 系统会根据自己的规则为约束进行命名*

create table teacher(

tid number(30),

tname varchar2(30) not null,

birthday date constraint teacher\_birthday\_nn not null

)

*--正常插入数据*

insert into teacher values(1,'liulaoshi','8-feb-1984')

*--非法插入数据*

insert into teacher values(1,null,'8-feb-1984')

insert into teacher values(1,'张三','')

select \* from teacher

* + 1. **唯一性约束**

*--unique 唯一性约束*

*--表级的约束 可以定义在单个字段 也可以定义在多个字段 同时可以在定义完成字段后再添加约束 而列级约束必须在建立字段的同时 建立约束条件*

create table student(

sid number (10),

sname varchar2(30) unique

)

*--插入数据*

select \* from student

drop table student

*--正常插入数据*

insert into student values (1,'aa')

*--非法插入数据*

insert into student values (2,'aa')

*--unique 是表级的约束 可以在定义完字段后 在添加约束*

create table student2(

sid number(30),

sname varchar2(30),

sage number(4),

constraint student\_sname\_un unique(sname)

)

*--正常插入*

insert into student2 values (1,'zs',11)

*--非法插入*

insert into student2 values (2,'zs',21)

*--下方的代码定义是错误的 not null 是列级约束 只能在定义字段的时候定义约束*

create table student3(

sid number(30),

sname varchar2(30),

sage number(4),

constraint student\_sname\_nn not null(sname)

)

* + 1. **主键约束**

*--primary key 主键约束*

*--主键约束是唯一标识表中的一行记录*

*--主键约束 也是表级约束 被主键约束的列 不能为空 也不能重复*

create table student3(

sid number(10) primary key,

sname varchar2(30) not null,

sage number(4)

)

*--正常插入数据*

insert into student3 values(1,'zs',16)

insert into student3 values(2,'ls',16)

select \* from student3

*--非法插入数据*

insert into student3 values(1,'ww',17)

insert into student3 values(null,'aa',18)

*--主键约束是表级约束 可以定义在多个字段上 也可以在定义完成字段之后添加约束*

create table student4(

sid number(10),

first\_name varchar2(30),

last\_name varchar2(40),

sage number(4),

*--联合主键*

constraint name\_pk primary key( first\_name, last\_name)

)

*--正常插入数据*

insert into student4 values (1,'zhang','san',14)

insert into student4 values (2,'li','si',24)

insert into student4 values(3,'zhang','wu',24)

insert into student4 values(4,'zhao','si',24)

*--联合主键 是指两个列的数据不能重复 单个字段允许重复*

insert into student4 values(5,'zhao','si',34)

select \* from student4

* + 1. **外键约束**

*--子表的外键 必须在主表中存在 否则插入报错*

*--外键必须参考主表的主键 或者唯一键*

create table depart(

did number (30) primary key,

dname varchar2(30) not null

)

create table emp(

eid number(30) primary key ,

ename varchar2(30) not null,

eage number(4),

depart\_did number(30),

constraint emp\_depart\_fk foreign key(depart\_did) references depart(did)

)

drop table depart

*--插入数据*

insert into depart values (1,'aaa');

insert into depart values (2,'bbb');

insert into depart values (3,'ccc');

insert into depart values (4,'ddd');

select \* from depart

*--插入数据到emp中*

insert into emp values (1,'zs',18,3)

insert into emp values (2,'lisi',28,2)

insert into emp values (3,'wangwu',29,2)

insert into emp values (4,'zhaoliu',23,1)

select \* from emp

*--删除数据*

*--4号部门没有员工可以直接删除*

delete from depart where did =4

*--3号部门正在被引用无法删除*

delete from depart where did =3

*--级联删除*

*--在删除父项的时候把子项 要么一并删除 要么置空*

create table depart(

did number (30) primary key,

dname varchar2(30) not null

)

create table emp(

eid number(30) primary key ,

ename varchar2(30) not null,

eage number(4),

depart\_did number(30),

constraint emp\_depart\_fk foreign key(depart\_did) references depart(did) on delete cascade *--在删除的时候 级联删除*

)

*--置空 子项*

create table depart(

did number (30) primary key,

dname varchar2(30) not null

)

create table emp(

eid number(30) primary key ,

ename varchar2(30) not null,

eage number(4),

depart\_did number(30),

constraint emp\_depart\_fk foreign key(depart\_did) references depart(did) on delete set null *--在删除父项的时候置空子项*

)

* + 1. **check约束**

create table student(

sid number (10) primary key,

sname varchar2(30) not null,

sage number(4) check(sage >=0 and sage <150),

sgender varchar2(8) check(sgender='male' or sgender='female')

)

*--正常插入数据*

insert into student values (1,'zs',120,'male')

*--非法插入数据*

insert into student values (1,'zs',160,'male')

insert into student values (2,'zs',160,'female1')

* + 1. **添加约束**

*--约束的管理*

*--查看约束*

select \* from user\_constraints

*--根据返回的结构可以查看 具体的类型 p:代表主键约束 U：代表唯一键约束 C:代表非空和检查约束*

select \* from user\_cons\_columns

*--在建标之后 添加约束*

create table stu(

sid number(10),

sname varchar2(30)

)

*--再次添加约束*

alter table stu add constraint stu\_sid\_pk primary key(sid)

select \* from user\_constraints where table\_name ='STU'

select \* from user\_cons\_columns where table\_name ='STU'

*--在创建表之后 添加外键约束*

create table depart(

did number (30) primary key,

dname varchar2(30) not null

)

create table emp(

eid number(30) primary key ,

ename varchar2(30) not null,

eage number(4),

depart\_did number(30)

)

*--添加外键*

alter table emp add constraint emp\_depart\_fk foreign key(depart\_did) references depart(did) on delete cascade

*--删除约束*

alter table emp drop constraint emp\_depart\_fk;

下面的约束只能定义在列级约束的是：

A、UNIQUE

B、NOTNULL

C、FOREIGN KEY

D、PRIMARY KEY

1. **处理数据（DML）**
   1. **数据控制语言（DML）**

可以把sql语言的操作分为四大类

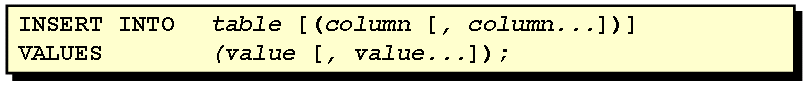
DML(Data Mainpulation Language) 数据操作语言 主要包含 insert update delete select

DDL(Data Defition Language) 数据定义语言 主要包含 create drop alter

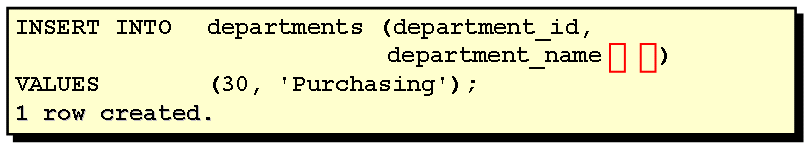
TCL(Transaction Controller Language) 事务控制语言 主要包含 commit savepoint rollback

DCL(Data Controller Language) 数据控制语言 主要包含 grant revoke

* 1. **插入数据**
     1. **INSERT 语句语法**
* 使用 INSERT 语句向表中插入数据。
* 使用这种语法一次只能向表中插入**一条**数据。



* + 1. **插入数据**
    2. **向表中插入空值**
* 隐式方式: 在列名表中省略该列的值。



* 显式方式: 在VALUES 子句中指定空值。
  + 1. **插入指定的值**
    2. **创建脚本 （使用 & 变量指定列值）**
* **在SQL 语句中使用 & 变量指定列值。**
* **& 变量放在VALUES子句中。**

--2.2.2. 插入数据

create table student(

sid number(10) primary key,

sname varchar2(30) not null,

address varchar2(30) not null,

birthday date default sysdate

)

--insert into 表名 values (具体的值 ) 具体的值 必须和表中的个数和类型相同

--insert into 表名 (列名1,列名2...) values （值1,值2,值3）

insert into student values(1,'张三','杭州西湖区','14-feb-1999');

select \* from student

insert into student values(2,'李四','北京三里屯','19-jan-1998');

insert into student(sid,sname,address) values(3,'王五','北京昌平区')

--下方代码是错误的 必须和表中的个数一直 会报值的个数不够

insert into student values (4,'赵六','山东济南')

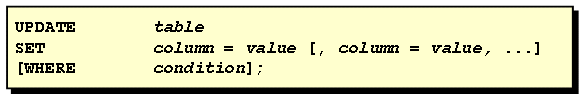
--可以指定值插入

insert into student values (4,'赵六','山东济南',sysdate)

--可以通过创建脚本的方式 进行指定变量 插入到数据库中

insert into student(sid,sname,address,birthday) values (&学号,&姓名,&住址,&出生日期)

* 1. **更新数据**
     1. **语法**



* + 1. **一般用法示例**

update student set address='济南市中区',birthday=sysdate where sid=5

* + 1. **在UPDATE语句中使用子查询**

update student

set address =(

select address from student where sid=2

),

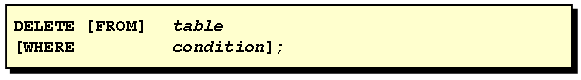
birthday=(

select birthday from student where sid=1

)

where sid=5

* 1. **删除数据**
     1. **语法**



* + 1. **一般用法示例**
    2. **在 DELETE 中使用子查询**
* 在 DELETE 中使用子查询，使删除基于另一个表中的数据。

*--删除表 中指定的数据*

delete from student where sid =5

delete from student where sid <=3

*--删除表中所有数据 类似于 truncate*

delete from student

* + 1. **Delete和Truncate**

都是删除表中的数据

Delete 可以执行rollback 进行回滚数据 可以闪回

Truncate是清空表 不可以进行或回滚数据

* 1. **数据库事务**
     1. **事务基础**

数据库事务有四大特性：ACID 原子性 一致性 隔离性 持久性

A:Atomicity 指事务逻辑上是不可分割的操作单元，所有的语句要么都执行成功，要么都失败

C:Consistentcy一致性一个状态转变为另外一个状态

I:Isolation 隔离性是针对并发而言，所谓并发是指数据库服务器如果同时处理多个事务，如果不采取专门的控制机制，并发事务之间是相互干扰的，隔离性就是要隔离并发运行的多个事务之间的相互影响。

D:Durability 持久性 事务一旦提交，对数据的修改就是持久性的， 数据的更新已经从内存中写入到磁盘中。

数据库事务可以由一下几个部分组成：

* 一个或多个DML
* 一个DDL 数据定义语言
* 一个DCL 语句

数据库事务

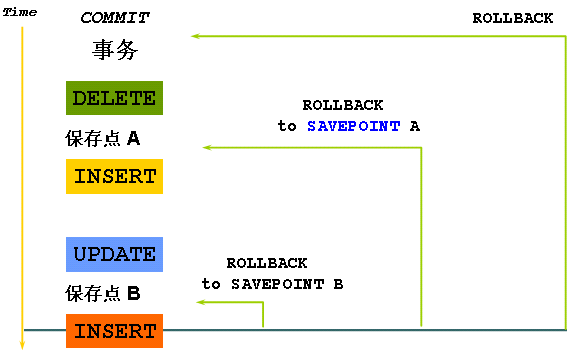
一般是以一个DML语句执行 作为开始

以下内容作为结束的标志

显示的结束 ：commit

隐式的结束： DDL语句 DCL语句 exit

* + 1. **控制事务**



insert into student values(1,'张三','杭州西湖区','14-feb-1999');

savepoint p1;

insert into student values(2,'李四','北京三里屯','19-jan-1998');

insert into student(sid,sname,address) values(3,'王五','北京昌平区');

savepoint p2;

delete from student where sid =2;

savepoint p3;

update student set sname='赵六' where sid =3;

rollback to p3;

rollback to p2;

*--提交事务*

*--显示提交事务*

commit;

update student set sname='赵六' where sid =3;

*--下面代码将隐式的提交上方所有的事务*

create table teacher(

tid number(10),

tname varchar2(30)

)

* + 1. **事务进程**
* 自动提交在哪些情况执行
* DDL语句
* DCL语句
* 不使用commit 或者rollback 语句，正常结束回话 exit 也可以提交
* 当执行到异常情况 也会自动提交
* 提交或回滚前数据的状态
* 在提交事务之前 --改变前的数据是可以恢复的 rollback
* 执行dml操作的用户可以通过select语句 查询之前修改的内容
* DML 语句 所设计到的行将 会被锁定 其他用户无法操作
* 提交后数据的状态
* 数据已经改变并且提交到了数据库
* 改变之前的值全部丢失
* 所有的用户都可以看到
* 锁都将被释放， 其他用户可以继续编辑数据
* 所有的保存点被释放
  + 1. **数据库的事务隔离级别**

对于多个运行的事务，当这些事务访问数据库中相同的数据吗，如果没有采取必要的隔离机制的手段，将会导致并发问题。

脏读：对于 两个事务 T1,T2 。T1读取了已经被T2更新**但是还没有提交的字段**之后，如果T2回滚，T1读取到的内容就是临时且无效的。

虚读(幻读)：对于两个事务T1,T2 。 T1从一个表中读取了一个字段，然后T2在该表中插入新的行，之后如果T1再次读取同一个表，就会多出几行数据。

不可重复读 对于两个事务T1，T2。 T1读取一个字段，然后T2更新了该字段之后，T1再次读取同一个字段，值就不同。

丢失更新，对于同时运行的多个事务，如果同时读取一份数据，第一个事务更改前三个字段的内容并提及， 第二个事务更改后三个字段的内容并提交。第二次更新的内容将覆盖掉第一次更新的内容。

**数据库为了解决由于隔离性引发的并发问题 定义了四种隔离级别**：

|  |  |
| --- | --- |
| 隔离级别 | 描述 |
| Read\_uncommmited（读取提交数据） | 允许事务读取未被其他事务提交的变更的内容。  脏读，不可重复读，虚度都会出现。 |
| Read\_commited | 只允许事务读取已经被其他事务提交的变更。可以避免脏读，但是不可重复读，虚度仍然会出现。 |
| Repeatable\_read | 确保事务可以多次从一个字段中读取相同 的值，在这个事务持续期间，禁止其他的事务对这个字段进行更新。避免了脏读和不可重复读 但是虚读仍然会出现。 |
| Serializable | 确保事务可以从一个表中读取相同的行，在这个事务持续期间，禁止其他的事务对该表执行插入，更新，删除操作。所有的并发问题都能解决但是效率低下。 |

悲观锁，乐观锁。

悲观锁： 就是利用数据库底层的锁机制。

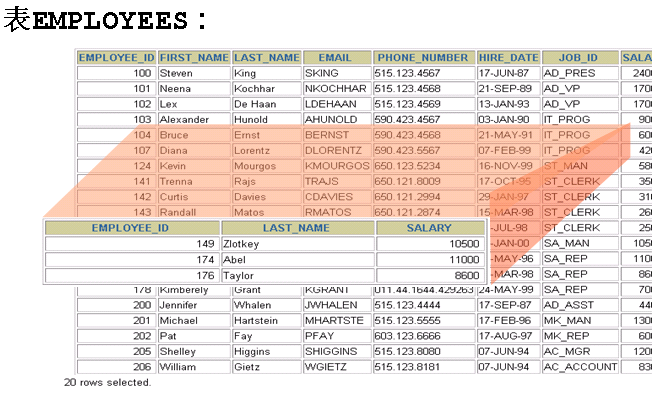
乐观锁：是和数据库无关的的，仅仅是添加一个字段，通过字段控制。效率较高。

Oracle支持两种隔离级别 ：*Read\_commited* 、Read\_commited oracle 的默认隔离级别是Read\_commited

MySql支持四种隔离级别： mysql默认的隔离级别是Repeatable\_read

1. **其他数据库对象**
   1. **视图**
      1. **视图说明**

视图 是从数据表中读取的一种虚拟的表，在视图上查询或者更新操作实际上的对基表完成的。



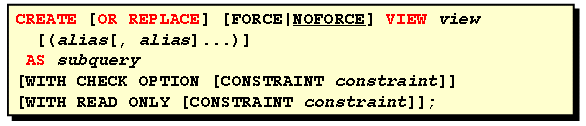
* 视图是一种虚表
* 视图是建立在已有表的基础上，视图赖以建立的这些表称为基表
* 向视图提供数据的内容为select语句，可以将视图理解为存储起来的select语句
* 视图可以屏蔽表的结构，限制数据的访问，可以简化复杂的查询，将常用的复合查询定义为视图后，每次访问不需要给出具体的查询逻辑，只需要从视图中取出数据即可
* 对视图所有是操作都是对基表直接的操作

如果用户需要对视图操作 那么需要有操作视图的权限

需要对用用户进行赋权

grant create any view to hr

* + 1. **创建视图**
* **使用下面的语法格式创建视图**



* + FORCE： 子查询不一定存在
  + NOFORCE： 子查询存在（默认）
  + WITH READ ONLY:只能做查询操作
* 子查询可以是复杂的 SELECT 语句

*--查询myemp表中工资最高的三个人 这个操作 在很多位置都需要*

*--以上方的需求创建视图*

create view salary\_top

as

select t1.first\_name ,t1.salary from

(select e.first\_name,e.salary from employees e order by e.salary desc)t1

where rownum<=3

*--可以利用这个视图进行查询 简化复杂的查询*

select \* from salary\_top

*--查看当前用户所用户的视图*

select \* from user\_views

select \* from EMP\_DETAILS\_VIEW

* + 1. **创建复杂视图**

复杂视图 是使用一个或者多个基表 使用函数和分组函数

--创建视图 查询各个部门的最低工资最高工资 平均工资

create view dept\_sum

(sname,minsal,maxsal,avgsal)

as

select d.department\_name,min(e.salary),max(e.salary),avg(e.salary)

from departments d, employees e

where d.department\_id=e.department\_id

group by d.department\_name

select \*from dept\_sum

* + 1. **查询视图**

查询其他对象语法完全相同

select \*from dept\_sum

* + 1. **修改视图**

create or replace view dept\_sum

(jobid,sname,minsal,maxsal,avgsal)

as

select e.job\_id, d.department\_name,min(e.salary),max(e.salary),avg(e.salary)

from departments d, employees e

where d.department\_id=e.department\_id

group by d.department\_name,e.job\_id

* + 1. **视图中使用DML的规定**

可以在简单的视图中执行DML操作

* 当视图定义了以下内容之一的时候不能使用update：
* 组函数
* Group by
* Distinct关键字
* Rownum伪列
* 列的定义为表达式
* 当视图定义了以下内容之一不能使用delete
* 组函数
* Group by
* Distinct关键字
* Rownum伪列
* 当视图定义了以下内容之一不能执行insert
* 组函数
* Group by
* Distinct关键字
* Rownum伪列
* 列的定义为表达式
  + 1. **屏蔽 DML 操作**

*--通过定义视图限制数据的访问 暴露指定的内容给第三方*

select \* from myemp

select \* from emp\_view

create or replace view emp\_view

as

select e.employee\_id, e.first\_name ,e.email from myemp e

update emp\_view set email ='guigu.com' where employee\_id=100

*--可以使用with read only 屏蔽对视图的DML操作*

create or replace view emp\_view

as

select e.employee\_id, e.first\_name ,e.email from myemp e

with read only

* + 1. **删除视图**

drop view view\_name

drop view dept\_sum

drop view emp\_view

drop view test

* 1. **序列**
     1. **什么是序列**

序列 可以给多个用户产生唯一的数据库对象

* 自动提供唯一的数值
* 共享对象
* 主要用于提供主键
* 将序列装入到内存中 可以提高访问效率
  + 1. **创建序列**
* **CREATE SEQUENCE 语句**

create sequence mysequ

increment by 1

start with 1

maxvalue 9999999

nocache

nocycle

* + 1. **NEXTVAL 和 CURRVAL 伪列**
    2. **序列应用举例**

select mysequ.nextval from dual;

select mysequ.currval from dual;

create table student(

sid number(20) primary key,

sname varchar2(30)

)

insert into student values (mysequ.nextval,'张三')

select \* from student

* + 1. **使用序列**

序列在以下情况下会出现裂缝

* 回滚
* 系统程序出现异常
* 多个表使用同一个序列
  + 1. **修改序列**

*--修改序列只能是修改增量 最大值 最小值 是否装入内存 无法对初始值进行改变*

alter sequence mysequ

increment by 5

maxvalue 888888

nocache

nocycle

* + 1. **删除序列**

Drop sequence seq\_name

drop sequence mysequ

* 1. **索引**

概念比较多 也会经常使用 面试题也会比较多

* + 1. **索引说明**

索引是用于 提升查询效率的数据库对象，使用索引可以快速定位数据，减少磁盘操作。

* 索引是一种独立于表的模式对象，可以存在在与表不同的磁盘或者表空间中。
* 索引被删除或者破坏，不会对表产生任何影响，其影响的仅仅是查询速度。
* 索引一旦建立，Oracle管理系统会对其自动维护，而且由oracel管理系统决定何时引用索引，用户不用再查询的时候指定使用哪个索引。
* 在删除一个表的时候，所有基于该表的索引会自动删除

索引的分类有哪些？

1）唯一性索引 和非唯一性索引

按照索引的字段是否允许出现重复

2）单字段索引 和 联合索引

按照字段基于索引的数目

3）普通索引和函数索引

按照索引是基于普通字段 还是基于复合的表达式

4） B树索引 和位图索引

按照索引的数据结构划分为B树索引(B\*Tree) 和位图(Bitmap)索引

* + 1. **创建索引**

*--创建基于ename字段的普通索引*

create index myindex\_ename on myemp(ename)

*--创建基于deptno 和sal 的联合索引*

create index myindex\_ds on myemp(deptno,sal)

*--创建基于 empno字段的唯一索引*

create unique index myindex\_empno on myemp(empno)

*--创建基于job字段的位图索引*

create bitmap index myindex\_bitmap on myemp(job)

*--创建基于函数的索引*

create index my\_fun on myemp(sal\*12)

* + 1. **什么时候创建索引**

以下情况需要创建索引：

列中的数据分布范围很广，字段中包含大量的空值

经常在where子句中 或者链接条件中出现

表中被访问的数据量很大 ，访问的数据大概占有数据总量的2%-4%之间

* + 1. **下列情况不要创建索引:**

表中的数据很小

列不经常作为链接条件 也不经常出现在where子句中

查询的数据大于4%不要建立 索引

表中的数据经常更新

* + 1. **查询索引**

*--查询所有的索引*

select \* from user\_indexes

*--查询这个索引在哪个字段上*

select \* from user\_ind\_columns

* + 1. **删除索引**

Drop index index\_name

drop index MYINDEX\_BITMAP

drop index MYINDEX\_DS

* 1. **同义词**
     1. **同义词说明**

使用同义词简化对象的名称

* + 1. **创建和删除同义词**

grant create synonym to hr

create synonym emp for employees

select \* from emp

drop synonym emp

数据字典

*--查询当前用户的权限*

*--查询当前系统索引用户 用用户id 用户创建时间*

select \* from all\_users

*--查询当前用户的详细信息*

select \* from user\_users

*--查询系统可以赋予哪些权限 208*

select \* from system\_privilege\_map