Oracle学习笔记

大型数据库：DB2、Oracle

中小型数据库：SQL Server、MySQL、Access

古老级别的数据库：Infomix、Sybase

Oracle 8/ Oracle 8i:其中i表示Internet,代表Oracle向网络发展。

Oracle 9i：现在使用最广泛的版本。

Oracle10g：过度产品，其中g代表grid网格。

Oracle11g：一个完整性产品。

Oracle数据库主要使用以下四个用户：

超级管理员：sys password:change\_on\_install

普通管理员：system password:manager

普通用户：scott password:tiger(默认是锁定的)

大数据用户：sh password:sh

对于win7 64位系统安装PL/SQL，需要使用了一个Oracle Instant Client ，且需要如下的设置。

PATH-需要包含Instant Client的根目录（oci.dll所在位置）

TNS ADMIN-需要指出tnsnames.ora所在位置的根目录

NLS LANG-为客户设定语言，区域和特征。

cmd 命令：sqlplus / as sysdba //作为sysdba登陆

alter user scott account unlock //用户名解锁

alter user scott identified by tiger //用户名更改密码为tiger

show user //显示登陆用户名

conn scott/tiger //登陆scott账户

conn sys/change\_on\_install as sysd //登陆sys账户

select \* from tab //显示所有的表

对于每一张表，都有其属于的用户，所以一张表的完整名称是“用户名.表名”或者说“模式名.表名称”，所以不同的用户想要访问其它用户的表，则应该添加上用户名，例如sys需要访问scott的表，需要使用如下的命令：

select \* from scott.emp;

如果使用超级管理员登陆，管理员可以控制数据库的实例的打开和关闭，关闭数据库实例命令为：

shutdown immediate; //启动数据库

startup //启动数据库

oracle也可以调用本机的操作系统命令，使用host命令作为前缀：

host copy test.sql test1.sql //copy为 操作系统的命令，通过host前缀调用操作系统命令

**sys和system用户的区别**

【system】用户只能用normal身份登陆em。

【sys】用户具有“SYSDBA”或者“SYSOPER”权限，登陆em也只能用这两个身份，不能用normal。

“SYSOPER”权限，即数据库操作员权限，权限包括：

打开数据库服务器 关闭数据库服务器

备份数据库 恢复数据库

日志归档 会话限制

“SYSDBA”权限，即数据库管理员权限，权限包括：

打开数据库服务器 关闭数据库服务器

备份数据库 恢复数据库

日志归档 会话限制

管理功能 创建数据库

normal 、sysdba、 sysoper有什么区别

normal 是普通用户

另外两个，你考察他们所具有的权限就知道了

sysdba拥有最高的系统权限

sysoper主要用来启动、关闭数据库，sysoper 登陆后用户是 public

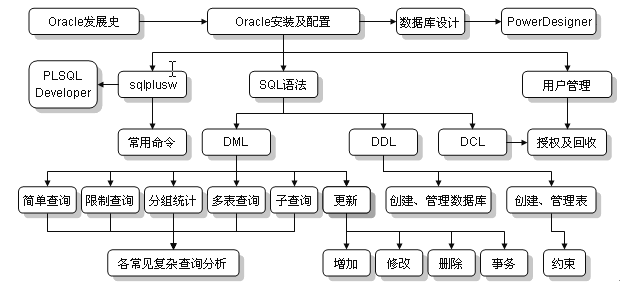
sysdba登陆后是 sys

**在Oracle数据库中，sql语句是不区分大小写，但所有数据库中的数据都是区分大小写的。**

安装Oracle 11g之后最重要的两个服务：

OracleOraDb11g\_home1TNSListenerALANCE：监听服务

OracleServiceALANCE：保存数据库的具体信息



数据库中除了用户使用的数据外，还有另外一种数据，它们是有关数据的定义信息，如数据库的名称、数据表的定义、数据库账户、权限等。这些数据不会经常性的使用，但是对数据库来说非常重要，这些数据通常存放在一个“数据字典“中，数据字典是数据库管理系统的工作依据。

**scott用户表结构：**

**部门表：DEPT**

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 名称 | 类型 | 描述 |
| 1 | DEPTNO | NUMBER（2） | 部门编号,由两位数字组成 |
| 2 | DNAME | VARCHAR2（14） | 部门名称，最多有14个字符组成 |
| 3 | LOC | VARCHAR（13） | 部门所在位置 |

**雇员表：EMP**

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 名称 | 类型 | 描述 |
| 1 | EMPNO | NUMBER（4） | 雇员的编号，由4为数字促成 |
| 2 | ENAME | VARCHAR2（10） | 雇员的姓名 |
| 3 | JOB | VARCHAR2（9） | 雇员的职位 |
| 4 | MGR | NUMBER（4） | 雇员对应的领导编号，领导也是雇员 |
| 5 | HIREDATE | DATE | 雇员的雇佣日期 |
| 6 | SAL | NUMBER（7,2） | 基本工资，其中2位小数，5位整数，一共7位 |
| 7 | COMM | NUMBER（7,2） | 奖金， |
| 8 | DEPTNO | NUMBER（2） | 雇员所在部门编号 |

**工资等级表：SALGRADE**

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 名称 | 类型 | 描述 |
| 1 | GRADE | NUMBER | 工资等级 |
| 2 | LOSAL | NUMBER | 此等级的最低工资 |
| 3 | HISAL | NUMBER | 此等级的最高工资 |

**工资表：BONUS**

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 名称 | 类型 | 描述 |
| 1 | ENAME | VARCHAR2（10） | 雇员姓名 |
| 2 | JOB | VARCHAR2（10） | 雇员职位 |
| 3 | SAL | NUMBER | 雇员的工资 |
| 4 | COMM | NUMBER | 雇员的奖金 |

**sql分类：**

dml—数据操纵语言（select, delete, update, insert）

ddl—数据定义语句（create, alter, drop, declare）

dcl—数据控制语句（grant, revoke, commit, rollback）

* **DML：**

1. **distinct**

select distinct job from emp; // distinct删除所有的行字段都相同的记录

1. **别名**

select distinct job jobname from emp; //为job取个别名为jobname，如果select多个字段，字段之间使用都好隔开，别名不需要使用逗号隔开。

1. **拼接**

select empno || ',' ||ename from emp; //将empno和ename拼接起来，单引号中的逗号表示原样输出，在sql中使用**单引号**表示字符串。

select empno, ename from emp; //**逗号的作用为将查询的数据分成多个不同的列，如果删除逗号，可以使用||将多列的数据拼接在一起。**

1. **等于**

select \* from emp where not job = 'CLERK'//不等于的三种写法，not可以用来求反操作，sql中的等号为一个等于号。

select \* from emp where job != 'CLERK' //不等于的三种写法

select \* from emp where not <> 'CLERK' //不等于的三种写法

1. **between and**

select \* from tablename where field between value1 and value2//between and 不仅适应于数字，而且适应于日期。

1. **not null**

is (not) null//判断字符内容是否为null

1. **in查询**

select \* from emp where empno in (7369,7566) //in指定查询范围，**in中不能出现null**

1. **万能的like**（模糊查询数字日期都适用）

**下划线匹配单个字符，%匹配多个字符，如果like后面跟上%%，可以查询所有的记录，这一个特点可以帮助用户节约很多代码。**

select \* from table1 where field1 like “%value1”

select \* from table1 where field1 like “a\_z”

select \* from table1 where field1 like “%az”

select \* from table1 where field1 like “%az%”

1. **排序（order语句排在所有的语句之后）**

select \* from order by field1, field2

select \* from store order by fileld1 desc, field2 desc(如果栏目一相同时，则依靠栏目二进行排序，asc表示升序，desc表示降序，默认为升序)

* **函数**

不同数据库都支持SQL，但是不同的数据库都有自己所支持的函数，这些就是单行函数，单行函数主要分为以下5类：**字符函数、数字函数、日期函数、转换函数、通用函数。**

1. **字符函数：**

UPPER:转换为大写,通常用来指定用户的输入，用户不会在意输入的大小写，但是Oracle中的所有数据都区分大小写。

select \* from emp where ename= UPPER('&str') //&用来读取用户的输入

LOWER：转换为小写

INITCAP：首字母大写

LENGTH：字符串长度

REPLACE：替换操作

SUBSTR：字符串截取，从0或者1开始截取都可以

Oracle中为了用户查询方便，专门提供了一个”**dual**”虚拟表。

即: select upper('hello') from dual

1. **数字函数**

ROUND（数字[,保留小数的位数]）：四舍五入

ROUND（903.4，-1）//从整数开始截取

ROUND（903.4，2）//保留两位小数

TRUNC：舍弃指定位置的内容，与ROUND区别在于不进行进位操作

MOD：取模

1. **日期函数**

通过SYSDATE取得当前日期。日期+数字=日期 日期-日期=天数

LAST\_DAY(日期)：本月最后一天

SELECT LAST\_DAY(SYSDATE) FROM DUAL

NEXT\_DAY(日期)：下一个星期一的日期

SELECT NEXT\_DAY(SYSDATE, '星期一') FROM DUAL

ADD\_MONTHS(日期):加上一个时间后的日期

MONTHS\_BETWEEN(日期):两个日期之间间隔时间

1. **转换函数**

转换函数完成数字、字符串、日期之间的转换。

TO\_CHAR(字符串，格式化字符串)：将日期、数字转换为字符串

SELECT TO\_CHAR(SYSDATE,'YYYY-MM-DD HH24:MI:SS') FROM DUAL;

SELECT TO\_CHAR(SYSDATE,'YYYY') FROM DUAL;

SELECT TO\_CHAR(SYSDATE,'FM') FROM DUAL;//用来消除月份之前的0

SELECT TO\_CHAR(123456789,'L999,999,999,999') FROM DUAL; //其中L表示LOCAL，即转换为本地语言使用的货币单位。

TO\_DATE(字符串，格式化字符串)：将字符串转换成DATE型数据，一般此函数在更新数据库的时候使用的较多。

SELECT TO\_DATE('2014-08-31','YYYY-MM-DD') FROM DUAL;

TO\_NUMBER(字符串，格式化字符串)：基本不用

1. **通用函数**

nvl(),decode()这两个函数为Oracle特有函数。

nvl():将空转换为0;

select (sal+ nvl(comm,0))\*12 from emp;//同样nvl(comm,100)表示将comm为空替换为100.

**decode()**:类似于程序中的if else语句，decode为oracle中最有特色的函数。

select decode(job,'CLERK', '销售') from emp;//将job列中的 CLERK转换为销售

* **多表查询**

select count(\*) from sales, costs where sales.prod\_id= costs. prod\_id

虽然上述语句避免了笛卡尔积的问题，但是查询结果依旧很缓慢，甚至不显示，**多表查询的性能很差**，当然前提是：数据量大。

在多表查询时，通常会给表起个别名

select e.ename, f.ename from emp e, emp f

where e.mgr=f.empno //自己关联自己

* **左右连接**

select \*

from emp e, deptno d

where e.empno(+)=d.empno

也可以使用如下的写法：

SELECT \*

FROM Emp e RIGHT JOIN Dept d

ON e.Deptno = d.Deptno

//右连接（可以理解为连接（+）右面的数据，所以为右连接），不必记住到底是左链接还是右连接，可以通过尝试来确定左右连接。这种符号是Oracle数据库独有的，left join 和right join为SQL语法通用的语法。

* **SQL1999**

除了以上的表连接操作：

**cross join**:交叉连接，产生笛卡尔积

select \* from emp cross join dept

**natural join**：自然连接，自动找到匹配的关联字段，消除笛卡尔积

select \* from emp natural join dept

**join using**：用户指定**关联字段**消除笛卡尔积

select \* from emp join dept using（deptno）

**join on**：用户自己指定一个消除笛卡尔积的关联条件

* **内连接：**指连接结果仅包含符合连接条件的行，参与连接的两个表都应该符合连接条件。inner join可以简写成join
* **外连接：**连接结果不仅包含符合连接条件的行同时也包含自身不符合条件的行。包括左外连接、右外连接和全外连接。

**连接方向的改变：**

**左（外）连接：**left outer join on//左边表数据行全部保留，右边表保留符合连接条件的行

**右（外）连接：**right outer join on//右边表数据行全部保留，左边表保留符合连接条件的行

**全（外）连接：**full outer join on //把两张表中没有的数据全部显示即左外连接 union 右外连接。

**外连接的outer都可以省略。left、right和full都标注出该连接为外连接，同样如果join前面没有标注出任何关键字则表示该连接为inner join，所以inner关键字也是可以省略的。**

**select \* from table1 join table2 on table1.id = table2.id**

**select \* from table1,table2 where table1.id = table2.id**

**其实这两种方式的执行结果一样，具体区别如下：**

**WHERE子句中使用的连接语句，在数据库语言中，被称为隐性连接。INNER JOIN……ON子句产生的连接称为显性连接。（其他JOIN参数也是显性连接）WHERE和INNER JOIN产生的连接关系，没有本质区别，结果也一样。但是！隐性连接随着数据库语言的规范和发展，已经逐渐被淘汰，比较新的数据库语言基本上已经抛弃了隐性连接，全部采用显性连接了。**

在Oracle之外的数据都使用以上的SQL1999语法操作。**多表查询的性能可定不高，而且性能在数据量比较大的情况下一定会受到影响。**

* **函数**

对于count（）函数，如果一张表中没有统计记录，则count返回0，而其它函数不返回值，注意区分。

**分组：**

**进行分组之后，select之后只能出现分组的字段和统计函数，其它的字段不能出现。也就是说select之后必须跟上分组的字段。 使用group by进行分组，group by需要放置在order by之前。**

**如果需要对分组之后的数据进行过滤，则不能再使用where进行过滤，应该是用having进行过滤。**

**where与having的区别如下：**

**where：在执行group by操作之前进行过滤，表示从全部数据中筛选出部分数据。不能在where之中使用统计函数。**

**having：是在group by分组之后的再次过滤，可以在having子句中使用同济函数。**

**注意:group by可以对多个关键字进行分组，对于多字段分组，select后面可以跟上多个字段。**

* 子查询

子查询=简单查询+限定查询+多表查询+统计查询。子查询出现在where和from之后较多。

* where之后：一般只返回单行单列、多行多列、单行多列的数据。

1. 出现较少的单行多列：

select \*

from emp

**where (job, sal) = (select job, sal from emp where ename = 'ALLEN')**

多行单列：需要使用三种判断符：IN、ANY、ALL

1. 在使用not in的时候一定需要注意一个问题，如果使用not in 操作，在子查询之中，如果有一个内容是null，则不会查询出任何结果。
2. =ANY:效果等同于IN
3. >ANY：比最小的大
4. <ANY：比最大的小
5. >ALL：比最大的还要大
6. <ALL：比最小的还要小

* from之后：多行多列，当做一张临时表出现。

**大部分情况下，如果最终查询结果之中需要出现select自居，但是又不能直接使用统计函数的时候，就在子查询的信息中统计，即：有复杂查询的地方大部分都需要子查询。**

* **数据的更新操作**

create table myemp as select \* from emp//oracle中支持的数据库，其它数据库不支持。

* **数据增加**

insert into myemp (empno, ename, hiredate, sal, comm) values(8888, '张三', to\_date('1960-08-17','yyyy-mm-dd'), 8000, 0)

* **数据更新**

update myemp set ename = 'lisi' **where** myemp.ename = 'zhangsan'

更新的时候如果不加上更新条件，则意味着更新全部的数据，但是这种做法绝对是不可取的，如果表中数据量较大的话，这种更新耗时相当的长，且性能会明显的下降。

* **数据删除**

delete from myemp where myemp.ename = 'lisi'

同样不加where条件时，删除整张表格里面的内容，但是该表格还是存在，如果连表格都需要删除需要使用drop table myemp。

* **事务处理**

所有的事务处理都是针对每一个session进行的，在oracle数据库中，每一个链接到数据库上的而用户都为一个session，每一个session彼此之间独立，不会有任何通信，每一个session独享自己的事务控制，而事务控制之中主要使用两个命令：

**事务的回滚：**rollback，更新操作回到原点。

**事务的提交：**commit，真正的发出更新操作，一旦提交便无法回滚。

**某一个session在更新数据表的时候还没有提交事务，其它的session是无法更新的，必须等待之前的session提交才可以。**

**这种问题从大的方面上讲称之为死锁。**开发者不用管死锁的事情，通常这些事情都是由DBA进行处理。

* **数据伪列**

数据伪列是指用户不需要处理的列，而是由Oracle自行维护的数据列，在Oracle之中有两个数据伪列：rownum、rowid。

* **rownum**

rownum系统自动生成的行号，且该行号是每次动态生成的。

rownum的优势在于可以在查询的过程中，不显示所有的行号，例如：select \* from emp where rownum < 5

注意，rownum不能使用between and 进行限制范围，如果需要限制范围，可以使用子查询，先查询前10条数据，然后再显示5~10条数据。通过子查询可以实现分页查询。实现代码如下:

select \* from

(select rownum m, empno, ename from emp

where rownum <=15)tmp

where tmp.m>10

* **rowid**

rowid与rownum的区别在于rowid不会随着select范围不同而随机变化，rowid是固定不变的。rowid表示每一行数据保存的物理地址编号。

* **表的创建**

对于数据库而言，实际上没一张表都表示一个数据库的对象，对象的操作包括：表、视图、索引、序列、约束等等，对于对象的操作主要分为以下三类语法：

* 创建对象：creat 对象名称；

create table member(name varchar2(500) default '无名氏',

age number(3))

insert into member(name, age) values('zhangsan', 20)

select \* from member

* 表的复制：

create table myemp as select \* from emp where 1=2//写一个一定不成立的等式，**可以实现表结构的复制**。

* 为表重命名:

在Oracle中，所有的数据实际上都是通过数据字典保存的，Oracle中提供了三种类型的数据字典，最常用的是：dba\_、user\_,下面代码为查询一个user\_tables数据字典：

select \* from user\_tables

因此修改表名称就相当于修改一条数据而已。

rename member to person//修改数据库表名称，该语法为Oracle特有。

* 截断表

truncate 表名称

truncate、delete与drop区别

truncate table 比 delete 速度快，且使用的系统和事务日志资源少。delete 语句每次删除一行，并在事务日志中为所删除的每行记录一项。

truncate table：删除内容、释放空间但不删除定义。

delete table:删除内容不删除定义，不释放空间。

drop table：删除内容和定义，释放空间。

drop table 表名称 purge//该表格不进入回收站，直接删除。

* 删除对象：drop 对象名称；

在Oracle之后，为了预防用户的误删除表操作，专门提供了回收站功能，用户所删除的表默认情况下会在一个回收站中保存，而用户也可以通过回收站，进行表的恢复，所以此技术称为闪回（flashback）。

查看回收站命令：

select \* from recyclebin; //显示回收站

flashback table 表名称 to before drop//恢复回收站中的表格

puger table 表名称//清空回收站中的一个表

purge recyclebin//清空回收站里面的所有表

* 修改对象：alter 对象名称；

alter table member add

alter table member modify

虽然在Oracle中给出了修改表结构操作，但是尽量不要使用这种表结构的修改，DB2中就不允许修改表结构。

常用数据类型有如下几种：

|  |  |  |
| --- | --- | --- |
| 数据类型 | 关键字 | 描述 |
| 字符串 | VARCHAR2(n) | n表示字符串最大长度 |
| 整数 | NUMBER(n) | 表示最多为n位的整数 |
| 小数 | NUMBER(n,m) | 其中n为整数位，m为小数位数 |
| 日期 | DATE | 存放日期、时间 |
| 大文本 | CLOB | 存储海量文字（4G） |
| 大对象 | BLOB | 存放二进制数据，例如：电影、图片、文字 |

一般在开发过程中使用最多的：VARCHAR2()，NUMBER，DATE(),CLOB，而对于BLOB字段一般使用较少。

* **约束：**
* 非空约束：建表的时候约束not null
* 唯一约束：每一列上的数据不允许重复，

create table member(name varchar2(500) unique,

age number(3))

create table member(name varchar2(500),age number(3) constraint UK\_email unique(email))//可以用来提示用户约束条件。

* 主键约束：主键约束=唯一约束+**非空**约束，主键一般作为唯一的一个标记出现。 从正常的开发而言，一张表只能有一个主键，但是SQL的语法规定而言，一张表却可以设置多个主键，此种做法称为复合主键，参考如下：

create table member(name varchar2(500),

age number(3), constraint PK\_name\_num primary key(name, age))//复合主键提示

create table member(name varchar2(500) primary key,

age number(3) primary key)//这种方式会报错。

但在实际开发过程中，正常人都不使用复合主键，只要是数据表，永远都只设置一个主键。

* 检查约束：check，为表中的数据增加一些过滤条件。

constraint ck\_sex check(sex in('男','女'))

constraint ck\_age check(age between 0 and 200)

* 主外键约束（核心难点）：主外键是在两张表中进行约束的，即：子表中某个字段的取值范围由父表所决定。使用主外键最大好处为控制了字表中某些数据的取值范围，见视频42。

constraint fk\_mid FOREIGN KEY(mid) REFERENCES member(mid)

constraint fk\_mid FOREIGN KEY(mid) REFERENCES member(mid) on delete cascade //级联删除

constraint fk\_mid FOREIGN KEY(mid) REFERENCES member(mid) on delete NULL //删除之后变为空。

drop table member cascade constraint purge//不关系子表是否存在，直接强制性的删除父表。

* 修改约束：alter table 表名称 add constraint 约束名称 约束类型

alter table 表名称 drop constraint 约束名称 约束类型

需要注意的是在5种约束中，非空约束是不可以后期添加的。

跟表结构一样，约束最好也不要修改。

* 查询约束:在oracle之中，所有的对象都在数据字典之中保存，可以直接查询，select \* from user\_constraints

purge recyclebin //清除垃圾箱

select \* from tab //查询用户名下的所有表

* **集合操作：**

**UNION:**

**UNION ALL:**

**INTERSECT:**

**MINUS:**

* 序列
* **视图**

creat view myview as

SQL语句

replace view myview as

SQL语句

select view from myview//相当于给所有的sql语句起了一个别名。

视图本身不是具体的真是数据，而是一些查询语句，所以这样的更新并不合理，那么在创建视图的时候建议将其设置为只读视图：

creat view myview as

SQL语句

**with read only**

* **同义词**

通常，不同的用户访问其它用户的表，则需要写上“用户.表名称”，那么为什么此时scott用户访问的时候直接使用dual即可，而不需要使用“sys.dual”，这个实际上就是同义词的应用，dual表示的是sys.dual的同义词，而同义词在Oracle之中称为SYNONYM。语法如下:

CREATE [public] SYNONYM myemp for scott.emp

同义词只是Oracle的概念。

* **索引**

所以的主要功能就是用户提升数据库的操作性能。索引通过二叉树（Oracle十几种索引中的一种，也是最简单的一种索引，称为B树索引）来时间优化搜索速度，因此内存中必须一致保持一颗这样的索引树，如果这棵树上的数据频繁的修改，则代码性能可定会降低。在Oracle之中创建索引有以下两种方式：

主键约束：如果一张表中的列上存在了主键约束的话自动创建索引。

手工创建:在某一个操作列上指定一个索引。

create index emp\_sal\_ind on emp(sal)

* **用户管理**

DCL：控制权限。如果把硬盘表示称一个图书馆的话，那么每个表空间表示每一个书柜，每一张表就表示柜子上的一本书。

创建用户：

create user 用户名 identified by 密码

alter user Alance identified by 密码;

alter user Alance password expire //密码失效，首次登陆时需要重新设置密码

alter user account lock //用户锁定

alter user account unlock //用户解锁

GRANT create session to 用户名

GRANT create table to 用户名 //此时只是将数据表的创建权限给了用户，但并没有把表空间的操作权限给用户，所以用户仍无法创建表，因此表也没地方保存。**为了解决用户的授权操作，在Oracle 之中为用户提供了许多的角色，每一个角色会包含多个权限，而角色主要有两个：CONNECT、RESOURCE**

GRANT CONNECT、RESOURCE to 用户名 //角色有多个权限，简化操作，不比对每一个用户赋权限。

使用Alance用户名无法访问scott用户权限下的表，此时粗腰将scott用户的权限授予Alance用户，主要的权限有：insert、delete、update、select

grant select on scott.emp to Alance //授予权限

revoke select on scott.emp to Alance //收回权限

* **数据库备份：**数据库备份是整个项目中最重要的工作之一，分为数据库的导出和导入。

**数据的导出：**使用exp命令

**数据的导入：**使用imp命令

**数据表的冷备份：**在关闭数据库实例的情况下进行数据库备份的操作，如果需要进行冷备份，需要备份一下几个核心内容（这几个文件缺失也是Oracle启动不起来的常见原因）：

**控制文件：**整个Oracle数据的实例服务的核心文件，直接通过v$controlfile找到

**重做日志文件：**可以进行数据的灾难恢复，通过v$logfile找到

**数据文件，表空间文件：**通过v$datafile和v$tablespace找到

**核心操作的配置文件：**通过show parameter pfile找到

* **数据库设计范式**

数据库的设计范式可以让数据表更好的进行数据的保存，数据库设计的永恒宝典:尽量避免多表关联。

1. **第一范式：**数据表中的数据列不可以再分。

create table member(

mid number primary key,

name varchar2(200),

contract varchar2(200)

)

上述不符合第一范式，contract可能包含电话、邮编、地址。所谓的不可分割指的是所有的数据类型都使用数据提供好的各个数据类型。

1. **第一范式：**不存在函数依赖，第二范式分为两种方式理解。（多对多）

方式一：列之间不应该存在函数关系

1. **第三范式（一对多）：**学生与徐晓的关系，第三范式是使用最多的一种范式。

* **数据库设计工具：powerdesigner**

insert into table1(field1, field2) values(value1, value2)

delete from table where 范围

update from tablename set field1= value1 where 范围

select count \* as totalcount from table1

select sum(field) as sumvalue from table1

select sum(sales) from store

select avg(field) as avgvalue from table1

select max(field) as maxvalue from table1

select min(field) as minvalue from table1

select count(store) from store where store is not null

select count(distinct store) from store where store is not null (剔重后的数字)

select store, sum(sales) from store\_info group by store

select store, sum(sales) from store\_info group by store having sum(sales)>199

通常在sql环境下，命令式不区分大小写，且每句命令不用；结尾。

创建数据库：

create database database-name

删除数据库：

drop database database-name

备份数据库：

backup database base1 to base2

创建新的数据表：

create table tablename(col1 type1 [not null] [primary key], col2 type2 [not null] ])

create table table\_new like table\_old

删除新表：

drop table tablename

增加一个列：

alter table tablename add column col type

增加主键：

alter table tablename add primary key(col)

删除主键：

alter table tablename drop primary key(col)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*alter drop 与truncate区别\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

当你不再需要该表时， 用 drop;

当你仍要保留该表，但要删除所有记录时， 用 truncate;

当你要删除部分记录时（always with a where clause), 用 delete.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*alter drop 与truncate区别\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

创建与删除索引：

create [unique] index indexname on tablename(col)

drop index indexname

创建与删除视图：

create view viewname as select statename

drop view viewname

常见的sql语句：

别名的使用：

select a1.store store, sum(a1.sales) “total sales”

from store\_info a1

group by a1.store

select a1.region region, sum(a2.sales) sales

from mytable a1, mytable2 a2

where a1.store = a2.store

group by a1.region

串接字符串：

select concat(region, store)

from mytable1

where store =’boston’

抓取栏目一部分：

使用substr()函数

合并表格：

合并两个列，并删除重复的数据(两个表格的合计)：

select field from table1

union

select field from table1

合并两个列，允许重复的数据：

select field from table1

union all

select field from table1

删除表格：

从表格a中筛选出不在表格b中的数据，并且删除重复数据(两个表格不重复部分)：

select field from table1

except

select field from table2

如果剔重，则为：

select field from table1

except all

select field from table2

找出两个表格的交集(剔重)：

select field from table1

intersect

select field from table1

不剔重

select field from table1

intersect all

select field from table2

--lesson about select

use scott --切换到scott数据库下编辑

--as的使用,别名最好使用双引号,因为Oracle中别名只支持双引号

select ename, sal, sal\*12 as "年薪" from emp

--distinct 过滤重复项

select distinct comm from emp

select distinct comm, deptno from emp --对comm和deptno的组合进行唯一性过滤

select comm, distinct deptno from emp --程序逻辑错误

--between的使用（数值在某个区间）

select \* from emp where sal > 0 and sal < 1500

select \* from emp where sal between 0 and 1500

select \* from emp where sal not between 0 and 1500

--in的使用（在选定的值内进行查找,注意在数据库中不等于使用!=个<>来表示,但是推荐后者来使用）

select \* from emp where sal in (800, 1500, 3000)

select \* from emp where sal not in (800, 1500, 3000)

select \* from emp where sal <> 800 and sal <> 1500 and sal <> 3000

--top的使用(注意top的使用位置)

select top 20 \* from emp

select top 50 percent \* from emp

select top 4 \*

from emp a1

where sal between 1500 and 3000

order by a1.sal asc

--not null用法(!!!!!注意:null 不可以参与<>, !=, =, +运算,但是可以参与is, is not运算,且任何数据类型都可以设置为null类型)

select \*

from emp a1

where comm is not null

--isnull的作用为,判断记录是否为null,如果is not null,则返回记录本身,如果is null返回

select ename, sal\*12 + isnull(comm,0) as "年薪" from emp

--order by的使用,默认使用升序进行排序

select \* from emp order by deptno asc, sal asc --两个记录都使用升序进行排序,注意联想和distinct的区别

--like的使用

select \* from emp where ename like 'M%'

select \* from emp where ename like '%M\_L%'

select \* from emp where ename like '[^a-a][a-z]%'--可以对查询的查询的记录的每个字母进行设定范围,且不区分大小写字母

select \* from emp where ename like '%\\_%' escape '\' --查询ename中含有\_的记录

-函数的使用

select upper(ename) from emp

select MAX(sal) from emp

select MIN(sal) from emp

select AVG(sal) from emp

select COUNT(sal) from emp

select COUNT(deptno) from emp

select COUNT(distinct deptno) from emp

select COUNT(comm) from emp --记录中的null不计数

--group by的使用

select job, deptno as "部门", avg(sal) as "部门平均薪水" --!!!稽核函数所针对的对象都是group by内部的最小单元,即先处理group by

from emp

group by deptno, job --可以联想order by的使用规则

order by deptno

--having 对分组之后的信息进行过滤

select deptno as "部门", AVG(sal) as "部门平均薪水"

from emp as A1

group by deptno

having COUNT(\*) > 3

select deptno as "部门", AVG(sal) as "部门平均薪水", COUNT(ename) as "部门员工数"

from emp as A1

where ename not like 'A%'

group by deptno --!group by 或者having 后面的约束必须使用原名,不可以使用别名

having avg(sal)>2000 and COUNT(\*) > 0 --where和having的顺序不可以颠倒

--内链接的使用(只返回满足条件的内容)

select \*

from emp as A1, dept as B1--输入两个表格的笛卡尔积

select sal

from emp as A1, dept as B1

where A1.deptno = B1.deptno and sal > 1000--如果对emptno进行过滤,则报错,where为sql92标准

select A1.ename as "员工姓名", B1.dname as "员工部门", A1.sal as "员工薪水"

from emp as A1

join dept as B1 --join 为sql99标准,推荐使用sql99

on A1.deptno = B1.deptno --on后面可以为任意条件,例如:1=1

where sal > 1000

select A1.ename as "员工姓名", B1.dname as "员工部门", C1.GRADE as "员工薪水等级"

from emp as A1

join dept as B1 --join 为sql99标准,推荐使用sql99

on A1.deptno = B1.deptno --on后面可以为任意条件,例如:1=1

join SALGRADE as C1

on A1.sal >= C1.LOSAL and A1.sal <= C1.HISAL

select A1.ename as "员工姓名", B1.dname as "员工部门", C1.GRADE as "员工薪水等级"

from emp as A1, dept as B1, SALGRADE as C1

where A1.deptno = B1.deptno and A1.sal >= C1.LOSAL and A1.sal <= C1.HISAL

select \* from emp, dept

where dept.deptno = 10

select \* from emp

join dept

on emp.deptno = dept.deptno

select A1.ename as "员工姓名", A1.deptno as "部门编号", C1.grade as "薪水等级"

from emp as A1

join SALGRADE as C1

on A1.sal>=C1.LOSAL and A1.sal<=C1.HISAL

select D1.deptno as "部门编号", B1.dname as "部门名称", D1.avgsal as "部门平均薪水", C1.grade as "部门薪水等级"

from (

select A1.deptno, AVG(A1.sal) as avgsal

from emp A1

group by A1.deptno

) as D1

join dept as B1

on D1.deptno = B1.deptno

join SALGRADE as C1

on D1.avgsal between C1.LOSAL and C1.HISAL

--on D1.avgsal >= C1.LOSAL and D1.avgsal <= C1.HISAL

select A1.ename as "领导姓名"

from emp as A1

where A1.job != 'manager'

select \* from emp

where EMPNO in(select mgr from emp)

select top 1 C1.avgsal as "部门平均薪水", C1.deptno as "部门编号", B1.dname as "部门名称"

from(

select AVG(A1.sal) as avgsal, A1.deptno

from emp as A1

group by A1.deptno

) as C1 --!!!!!C1也可以取名A1,这是因为A1的作用范围和C1不用,不会发生冲突

join dept as B1

on C1.deptno = B1.deptno

select top (COUNT(A1.sal)-1) \*

from emp as A1

order by A1.sal desc

--外链接的使用(不满足条件的值也会返回)

--内链接默认使用join,实际为inner join的缩写,外连接有左链接和右链接只分

select \*

from dept as A1

left join emp as B1

on A1.deptno = B1.deptno

--全链接full join,相当于左链接和右链接的集合

--交叉链接cross join

select \*

from emp

cross join dept

--等价于

select \*

from emp, dept

--等价于

select \*

from emp

join dept

on 1 = 1

--自链接,用于自己和自己链接,下面例子在不适用聚合函数的情况下,查询员工薪水最高的员工信息

select \*

from emp

where emp.EMPNO not in

(

select distinct A1.EMPNO

from emp as A1

join emp as A2

on A1.sal < A2.sal

)

--联合的使用

select ename, sal

from emp

where emp.sal >= 2000

union

select ename, sal

from emp

where emp.sal < 2000

--identity的使用,通常用户主键表示自动增长,一般不认为地对主键进行赋值

create table student

(

student\_id int primary key identity(100, 5),

student\_name nvarchar(200) not null

)

insert into student values('张三')--默认情况下主键不可以进行赋值操作

select \* from student

--视图的使用,视图逻辑上为表的别名,视图的优势在于便于查询,而且可以隐藏表内的内容

--不建议通过视图更改原始表中的数据,视图中的所有数据必须都有别名,不可以为avg(sal)

create view V\_emp

as

select deptno, AVG(sal) as "avgsal"

from emp

group by deptno

select \* from V\_emp

--事务:避免数据出于不合理的中间状态,避免数据并发性给数据带来的影响

--事务一般通过第三方插件来实现事务的处理