**数据库学习笔记**

**sql分类：**

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

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

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

**通常在sql环境下，命令式不区分大小写，且每句命令不用一；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：

select [distinct] from tablename where fields>100 or

(sales <500 and sales>275) (distinct用来剔重)

select \* from tablename where field in (‘’,’’) 包含哪些数据的那些列

select \* from tablename where field between value1 and value2

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

delete from table where 范围

update from tablename set field1= value1 where 范围

万能的like

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%”

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)

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

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