# 1.查询语句

--单行注解

/\*

多行注解

\*/

--查询语法

--select 字段名字,字段名,字段名,字段名,字段名...,字段名 from 表名

--select \* from 表名 查询所有字段

--查询员工表中的id ,名字, 邮箱,工资

SELECT employee\_id,last\_name,email,salary FROM employees;

--查询员工表中的id,名字,经理号,部门,工资

SELECT employee\_id,last\_name,manager\_id,department\_id,salary FROM employees;

--查询所有员工信息

SELECT \* FROM employees;

1.运算符 + - \* /(正常除法运算)

--员工工资减半

SELECT employee\_id/2,salary\*2 FROM employees;

--2+5

SELECT 2+5 FROM employees;

--虚拟表dual

SELECT 2+5 FROM dual;

SELECT 5-2 FROM dual;

SELECT 2-5 FROM dual;

SELECT 5\*2 FROM dual;

--SELECT 5%2 FROM dual; 语法错误没有%操作

--查询员工id,名字,工资 ,增长10000

SELECT employee\_id,last\_name,salary,salary+10000 FROM employees;

--查询员工id,名字,工资 ,上浮0.5

SELECT employee\_id,last\_name,salary,salary+salary\*0.5 FROM employees;

SELECT employee\_id,last\_name,salary,salary\*1.5 FROM employees;

SELECT employee\_id,last\_name,salary,salary\*(1+0.5) FROM employees;

2.起别名 as 名字 起名字默认大写

-- 空格 名字

SELECT employee\_id,last\_name,salary,salary+salary\*0.5 AS newsal FROM employees;

SELECT employee\_id,last\_name,salary,salary\*(1+0.5) newsalary FROM employees;

--定义的名字是小写的 用""强制

SELECT employee\_id,last\_name,salary,salary\*1.5 "newsal" FROM employees;

3.字符串 单引号括起来

SELECT 'helloworld' FROM dual;

--' 转义字符 转义'

SELECT 'hello''world' "aa" FROM dual;

--查询id ,名字 ,部门,工作,工资 , 年薪(工资\*12)

SELECT employee\_id,last\_name,department\_id,job\_id,salary,salary\*12 yearsal FROM employees;

4.null 等价于'' 空 与任何值做运算结果都是空

SELECT null+5 FROM dual;--null

SELECT ''+5 FROM dual;--null

SELECT null\*5 FROM dual;--null

SELECT ''/5 FROM dual;--null

SELECT null-5 FROM dual;--null

--\* /高于+-

--查询id ,名字 ,部门,工作,工资 , 年薪(工资\*12) 年终奖100000

SELECT employee\_id,last\_name,department\_id,job\_id,salary,salary\*12+100000 yearsal FROM employees;

# 2.where过滤语句02

--oracle 语句中没有所谓的true 或者false

--条件查询where 放在查询语句后面

--select 字段,字段,字段,字段,字段,字段...,字段 from 表名 where 条件

1.关系运算符 > < >= <=

--查询工资大于17000 员工信息

SELECT \* FROM employees WHERE salary >17000;

--查询工资小于17000 员工信息

SELECT \* FROM employees WHERE salary <17000;

--查询工资小于等于17000 员工信息

SELECT \* FROM employees WHERE salary <= 17000;

--查询工资小于等于17000 员工信息

SELECT \* FROM employees WHERE salary >= 17000;

2. = 赋值 判断是否相等 != <> ^=

--查询工资等于17000 员工信息

SELECT \* FROM employees WHERE salary = 17000;

--查询工资不等于17000 员工信息

SELECT \* FROM employees WHERE salary != 17000;

SELECT \* FROM employees WHERE salary <> 17000;

SELECT \* FROM employees WHERE salary ^= 17000;

-- from 1 where 2 select 3

--查询员工信息年薪在70000以上的员工id 工资 年薪

SELECT employee\_id,salary, salary\*12 FROM employees WHERE salary\*12>70000;

--别名不能做where语句条件判断,表达式,字段可以的

--SELECT employee\_id,salary, salary\*12 yearsal FROM employees WHERE yearsal>70000;--错误

--查询员工的编号 ,名字last\_name,工资,年薪(工资+奖金)

--空 null '' 与任何值做计算结果都是空

SELECT employee\_id,last\_name,salary,commission\_pct,(salary+salary\*commission\_pct)\*12 FROM employees;

3.连接方式 ||

SELECT 'hello'||'world' FROM dual;

--SELECT 'hello'+'world' FROM dual; 不合法

SELECT '1'+'5' FROM dual; --6 自动转成数

SELECT '1'||'world' FROM dual;--1world

SELECT 1||'world' FROM dual;--1world 自动转成字符串

--查询员工信息 id 姓.名 姓.职务id 部门号 部门经理号

SELECT employee\_id,last\_name||'.'||first\_name AS myname, last\_name||'.'||job\_id myjob ,department\_id,manager\_id FROM employees;

--查询部门是90号员工信息

SELECT \* FROM employees WHERE department\_id=90;

--查询部门大于50号员工信息

SELECT \* FROM employees WHERE department\_id>50;

--查询部门不是90号员工信息

SELECT \* FROM employees WHERE department\_id<>90;

--空 与任何值做判断结果都是假

--字母比较按照自然顺序

SELECT \* FROM employees WHERE last\_name>'King';

--关键字不区分大小写,列名,列值,表名区分大小写

--列名与表名默认大写

SELECT \* FROM employees WHERE last\_name>'king';

--查询没有部门的员工信息 is null is not null

SELECT \* FROM employees WHERE department\_id IS NULL;

--查询有部门的员工信息

SELECT \* FROM employees WHERE department\_id IS NOT NULL;

--标识符 表名 列名 别名等 命名规范 数字 /字母 /\_ # $可以 数字不能开头的

--如果名字不合法,可以通过""强制写

SELECT employee\_id,last\_name,salary,commission\_pct,(salary+salary\*commission\_pct)\*12 AS "1year$sal" FROM employees;

SELECT employee\_id,last\_name,salary,commission\_pct,(salary+salary\*commission\_pct)\*12 AS year$sal FROM employees;

SELECT employee\_id,last\_name,salary,commission\_pct,(salary+salary\*commission\_pct)\*12 AS "year$sal" FROM employees;

4.默认的日期格式是 DD-MON-RR 日 月 年 17-Oct-18

--查询1987年6月17之后入职的员工

--SELECT \* FROM employees WHERE hire\_date> '1987年6月17日'

SELECT \* FROM employees WHERE hire\_date> '17-6月-87';

SELECT \* FROM employees WHERE hire\_date> '17-6月-1987';

5.逻辑优先级 not 不是 and 与 or 或者

--查询50号部门的员工工资大于3500的信息

SELECT \* FROM employees WHERE department\_id=50 AND salary>3500;

--查询在50号部门和90号部门的员工信息

SELECT \* FROM employees WHERE department\_id=50 OR department\_id=90;

--查询50号部门员工的工资在3500以上,90号部门的工资在17000以上的信息

SELECT \* FROM employees WHERE department\_id= 50 AND salary>3500 OR department\_id=90 AND salary>17000;

--查询员工工资在3500~17000之间的信息

SELECT \* FROM employees WHERE salary >=3500 AND salary<=17000;

SELECT \* FROM employees WHERE salary between 3500 AND 17000;--意义同上

--查询部门在60~90之间的信息

SELECT \* FROM employees WHERE department\_id between 60 AND 90;

--查询部门号是 50 60 80 90 的员工信息

SELECT \* FROM employees WHERE department\_id = 50 OR department\_id = 60 OR department\_id = 80 OR department\_id = 90;

--in() 在这个里面的任意一个 not in()不在这里面

SELECT \* FROM employees WHERE department\_id in(50,60,80,90);

--查询部门号不是 50 60 80 90 的员工信息

SELECT \* FROM employees WHERE department\_id != 50 AND department\_id != 60 AND department\_id != 80 AND department\_id != 90;

SELECT \* FROM employees WHERE department\_id NOT in(50,60,80,90);

--查询名字last\_name K开头的

6.like 模糊查询(instr也是)

--%0~n个 \_ 1个

SELECT \* FROM employees WHERE last\_name LIKE 'K%';

--查询名字last\_name 是K开头的 4个字母

SELECT \* FROM employees WHERE last\_name LIKE 'K\_\_\_';

----查询名字last\_name g结尾的

SELECT \* FROM employees WHERE last\_name LIKE '%g';

--查询last\_name中含有i的

SELECT \* FROM employees WHERE last\_name LIKE '%i%';

--查询last\_name中含有i或g

SELECT \* FROM employees WHERE last\_name LIKE '%i%' OR last\_name LIKE '%g%';

--查询last\_name中含有i和g

SELECT \* FROM employees WHERE last\_name LIKE '%i%' AND last\_name LIKE '%g%';

--like 等价于 =

SELECT \* FROM employees WHERE employee\_id LIKE '100';

SELECT \* FROM employees WHERE employee\_id LIKE 100;

--模糊查询

SELECT \* FROM employees WHERE employee\_id LIKE '1%';

--查询email中包含EN的员工

SELECT \* FROM employees WHERE email LIKE '%EN%';

--查询last\_name第三个字母是a，并且a的后面包含en的员工

SELECT \* FROM employees WHERE LAST\_NAME LIKE '\_\_a%en%';

--查询email中包含R，且R的后面有四个字符的员工

SELECT \* FROM employees WHERE email LIKE '%R\_\_\_\_';

--查询员工编号以6结尾的员工

SELECT \* FROM employees WHERE employee\_id LIKE '%6';

--查询工资大于12000的员工

SELECT \* FROM employees WHERE salary > 12000;

--查询员工编号为176的员工

SELECT \* FROM employees WHERE employee\_id LIKE 176;

SELECT \* FROM employees WHERE employee\_id = 176;

SELECT \* FROM employees WHERE employee\_id IN( 176);

--SELECT \* FROM employees WHERE employee\_id is 176; is null is not null

7.Between and范围 in具体值

--查询工资不在5000~6000范围的员工

SELECT \* FROM employees WHERE salary NOT BETWEEN 5000 AND 6000;

--查询入职日期在1998-2-1到1998-5-1之间的员工

SELECT \* FROM employees WHERE hire\_date BETWEEN '1-2月-1998' AND '1-5月-1998';

SELECT \* FROM employees WHERE hire\_date >= '1-2月-1998' AND hire\_date<= '1-5月-1998';

--查询在20号或50号部门工作的员工

SELECT \* FROM employees WHERE department\_id in(20,50);

--between and 范围

--in 具体值

--查询在50~70部门的员工信息

SELECT \* FROM employees WHERE department\_id BETWEEN 50 AND 70;-->= 50 <=70

SELECT \* FROM employees WHERE department\_id IN( 50 , 70);--是50部门 或是70部门

SELECT \* FROM employees WHERE department\_id IN( 50 ,60 , 70);--是50部门或是60/70部门

8.去除重复行，&值，伪列rownum

--去除重复行DISTINCT

SELECT DISTINCT department\_id FROM employees

--输入一个值&

SELECT \* FROM employees WHERE last\_name = '&myname';

SELECT \* FROM employees WHERE employee\_id = &myid;

--伪列 rownum序号 rowid物理地址唯一值

SELECT ROWNUM,ROWID ,employee\_id,last\_name FROM employees;

--查询前10条的信息

--rownum 可以做<= < =1 比较 其他均为假

SELECT ROWNUM,e.\* FROM employees e WHERE ROWNUM <=10;

SELECT ROWNUM,e.\* FROM employees e WHERE ROWNUM <10;

SELECT ROWNUM,e.\* FROM employees e WHERE ROWNUM =1;

SELECT ROWNUM,e.\* FROM employees e WHERE ROWNUM =2;

--查询10,20,90号部门的员工

SELECT \* FROM employees WHERE department\_id IN (10,20,90);

SELECT \* FROM employees WHERE department\_id=10 OR department\_id = 20 OR department\_id=90;

--查询不是10,20,90号部门的员工

SELECT \* FROM employees WHERE department\_id NOT IN (10,20,90);

SELECT \* FROM employees WHERE department\_id!=10 AND department\_id != 20 AND department\_id!=90;

--查询工资大于5000而且last\_name以s结尾(and)

SELECT \* FROM employees WHERE salary>5000 AND LAST\_name LIKE '%s';

--查询部门编号是10,20，或者没有部门的员工信息

SELECT \* FROM employees WHERE department\_id IN (10,20) OR department\_id IS Null ;

--查询部门编号不为空的员工信息

SELECT \* FROM employees WHERE department\_id IS NOT NULL;

--查询员工工资加1000之后大于5000的员工+信息

SELECT employee\_id,salary,salary+1000 sal FROM employees WHERE salary+1000>5000;

--查询部门编号不为空并且工资加上1000之后小于5000的员工

SELECT \* FROM employees WHERE department\_id IS NOT NULL AND (salary+1000)<5000;

9.排序 order by 字段名字,字段名字... 最后面--asc 升序 默认升序 desc 降序

# 3.函数

--函数 实现某一个功能

--单行函数:输入一行,结果一行

--多行函数:输入多行,结果一行

--数据类型

--字符类型

--字符型:char(字节数) 2000字节 不可变的 char(8)最大只能放8个字节 tom 占3 但是8个处理的

--字符型:varchar2(字节数) 4000字节 可变的 varchar2(8)最大只能放8个字节 tom 占3 但是3个处理的

--字符型:clob 4G 大文本格式

--数值类型

--数值:number 存储整型或者浮点型，最大38位精度

--数值:number(4) 4位 最大值9999

--数值:number(5,2) 5位 2位小数位 整数位3位

--日期类型:

--date:存储年月日时分秒，精确到秒

--timestamp(n):时间戳 精确到纳秒

--二进制数据类型

--blob : 大二进制数据 最大值4G

1.大写小写concat,substr,length

--字符函数

--lower 变小写

SELECT lower('ABC') FROM dual;

-- king.steven

SELECT lower(last\_name) || '.'|| lower(first\_name) FROM employees;

--upper 变大写

SELECT upper('abc') FROM dual;

--将last\_name 变成大写 KING.steven

SELECT upper(last\_name) || '.'|| lower(first\_name) FROM employees;

--initcap 首字母大写,其余小写

SELECT initcap('abc') FROM dual;

--将last\_name变成小写之后再变成大写

SELECT upper(lower(last\_name)) FROM employees

--函数与函数之间是可以嵌套的

--concat(参数1,参数2) 连接

SELECT last\_name || first\_name FROM employees;

SELECT concat(last\_name,first\_name) FROM employees;

SELECT concat('hello','world','123') FROM dual;--错误

SELECT concat(concat('hello','world'),'123') FROM dual;

--last\_name变成大写 first\_name变成小写 KING.steven

SELECT concat(concat(upper(last\_name),'.'),lower(first\_name)) FROM employees;

--substr(字符串，起始位置[，截取的个数]) 截取一个字符串的子串

SELECT substr('12345678',2) FROM dual;--2345678 从左向右 第二个位置开始 截取到最右侧

SELECT substr('12345678',-2) FROM dual;--78 从右向左 第二个位置开始 截取到最右侧

SELECT substr('12345678',2,2) FROM dual;--23 从左向右 第二个位置开始 截取2个

SELECT substr('12345678',-2,1) FROM dual;--7 从右向左 第二个位置开始 截取1个

--查询员工信息 名字以 k.steven 形式呈现

SELECT concat(concat(substr(lower(last\_name),1,1),'.'),lower(first\_name)) FROM employees;

SELECT lower(substr(last\_name,1,1))||'.'||lower(first\_name) FROM employees;

--查询K开头的员工信息

SELECT \* FROM employees WHERE substr(last\_name,1,1)='K';

--查询g结尾的员工信息

SELECT \* FROM employees WHERE substr(last\_name,-1,1)='g';

--length 长度

SELECT length('123') FROM dual;

--查询first\_name 的字符数等于5的员工信息

SELECT \* FROM employees WHERE length(first\_name)=5;

--查询g结尾的员工信息

SELECT \* FROM employees WHERE substr(last\_name,length(last\_name),1)='g';

--查询K开头的员工信息

SELECT \* FROM employees WHERE substr(last\_name,-length(last\_name),1)='K';

2.instr(字符串，查找子串[，起始位置[，第几次出现]])查找字符串中子串的起始位置，如果找不到返回0

SELECT instr('12345678','3') FROM dual;--3 从第一位开始找第一次出现的位置

SELECT instr('12345678','34') FROM dual;--3 从第一位开始找第一次出现的位置

SELECT instr('12345678123123123','3',3) FROM dual;--3 从第3位开始找第1次出现的位置

SELECT instr('12345678123123123','3',4) FROM dual;--11 从第4位开始找第一次出现的位置

SELECT instr('12345678123123123','3',-1) FROM dual;--17 从右向左第一个位置开始找第一次出现的位置

SELECT instr('12345678123123123','3',3,3) FROM dual;--14 从第3位开始找第3次出现的位置

SELECT instr('12345678123123123','12',3,2) FROM dual;--12 从第3位开始找第一次出现的位置

SELECT instr('12345678123123123','3',-3,1) FROM dual;--14 从右往左第3位开始找从右往左第一次出现的位置

SELECT instr('12345678123123123','3',-5,2) FROM dual;--3 从右往左第5位开始找从右往左第2次出现的位置

--查询last\_name中g结尾的员工信息

SELECT \* FROM employees WHERE instr(last\_name,'g',-1,1)=length(last\_name);

SELECT \* FROM employees WHERE instr(last\_name,'g',length(last\_name))!=0;

--查询last\_name中K开头的员工信息

SELECT \* FROM employees WHERE instr(last\_name,'K')=1;

SELECT \* FROM employees WHERE instr(last\_name,'K',-length(last\_name))>0;

--查询last\_name中包含i的员工信息

SELECT \* FROM employees WHERE instr(last\_name,'i') !=0;

--last\_name倒数第三个字母是i的信息

SELECT \* FROM employees WHERE instr(last\_name,'i',-3,1)=length(last\_name)-2;

SELECT \* FROM employees WHERE instr(last\_name,'g',-5,1)=length(last\_name)-4 AND instr(last\_name,'g',-5,1) !=0;

3.lpad | rpad (字符串，固定长度，填充字符)填充字符串

--lpad 左侧填充

SELECT lpad('abc',7,'\*') FROM dual;--\*\*\*\*abc

SELECT lpad('abcabcabcabc',7,'\*') FROM dual;--abcabca 从左侧开始截取7位

--rpad 右侧填充

SELECT rpad('abc',7,'\*') FROM dual;--abc\*\*\*\*

SELECT rpad('abcabcabcabc',7,'\*') FROM dual;--abcabca 从左侧开始截取7位

--trim | ltrim | rtrim(关键字 from 字符串)|(字符串)|(字符串,关键字）修剪字符串

4.trim 去两端的 (关键字 from 字符串)|(字符串)

SELECT TRIM(' nn ') ||'123' FROM dual;--去空格

SELECT TRIM(' n n ') ||'123' FROM dual;

SELECT TRIM('a' FROM 'aaaaaabcn nabcaaaaa' )||'123' FROM dual;--从aaaaaabcn nabcaaaaa 截取a

--SELECT TRIM('aaaaaabcn nabcaaaaa','a' ) FROM dual;--不合法

--ltrim 左侧去字符 (字符串)|(字符串,关键字）

SELECT '123'||ltrim(' nn ')||'123' FROM dual;--去空格

SELECT '123'||ltrim(' n n ')||'123' FROM dual;

SELECT '123'||ltrim('aaaaannaaaaaa','a')||'123' FROM dual;

--rtrim 右侧去字符 (字符串)|(字符串,关键字）

SELECT '123'||rtrim(' nn ')||'123' FROM dual;--去空格

SELECT '123'||rtrim(' n n ')||'123' FROM dual;

SELECT '123'||rtrim('aaaaannaaaaaa','a')||'123' FROM dual;

--13456789876 134\*\*\*\*9876

SELECT substr('13456789876',1,3)||'\*\*\*\*'|| substr('13456789876',-4,4) FROM dual;

SELECT rpad(substr('13456789876',1,3),7,'\*')|| substr('13456789876',-4,4) FROM dual;

SELECT rpad(rpad('13456789876',3,'\*'),7,'\*')|| substr('13456789876',-4,4) FROM dual;

5.replace(字符串，查找字符串[，替换字符串])替换字符串中的子串，默认是替换为空字符串

SELECT REPLACE('abcacbsdhkajfhjkfhabcabacabc','a') FROM dual;--去掉a

SELECT REPLACE('abcacbsdhkajfhjkfhabcabacabc','a','\*') FROM dual;--去掉a

SELECT REPLACE(' abcacb sdhkajf hjkfhabca baca bc ',' ') FROM dual;--去掉空格

--CHR把编码转换为字符

SELECT chr(97) FROM dual;

--ASCII把字符转换为编码

SELECT ascii('a') FROM dual;

SELECT ascii('好') FROM dual;

SELECT chr(15050173) FROM dual;

6.round:(数字[,小数位数])四舍五入

SELECT round(3.1415) FROM dual;--3

SELECT round(3.6415) FROM dual;--4

SELECT round(-3.1415) FROM dual;-- -3

SELECT round(-3.6415) FROM dual;-- -4

SELECT round(3.1415,2) FROM dual;--3.14

SELECT round(3.6465,2) FROM dual;--3.65

SELECT round(-3.1415,2) FROM dual;-- -3.14

SELECT round(-3.6465,2) FROM dual;-- -3.65

7.trunc:(数字[,小数位数]) 截断

SELECT trunc(3.1415) FROM dual;--3

SELECT trunc(3.6415) FROM dual;--3

SELECT trunc(3.1415,2) FROM dual;--3.14

SELECT trunc(3.6465,2) FROM dual;--3.64

8.mod,ceil,floor,sysdate

--mod :(被除数，除数) 取余

SELECT mod(5,2) FROM dual;--1

SELECT mod(2,5) FROM dual;--2

--ceil：进位取整，返回不小于本身最小整数===>大于等于它的最小整数

SELECT ceil(3.6) FROM dual;-- 4

SELECT ceil(-3.6) FROM dual;-- -3

SELECT ceil(-3.3) FROM dual;-- -3

--floor：返回不大于本身最大整数===>小于它的最大整数

SELECT floor(3.6) FROM dual;-- 3

SELECT floor(-3.6) FROM dual;-- -4

SELECT floor(-3.3) FROM dual;-- -4

--sysdate 获得当前时间

SELECT '当前时间:'||SYSDATE FROM dual;

SELECT '当前时间:'||(SYSDATE+1) FROM dual;--下一天

SELECT '当前时间:'||(SYSDATE-1) FROM dual;--上一天

--一个月之后

SELECT '当前时间:'||(SYSDATE+31) FROM dual;

--一年之后

SELECT '当前时间:'||(SYSDATE+365) FROM dual;

--入职多少个月

SELECT '入职天数:'|| trunc(sysdate-hire\_date) FROM employees;

SELECT '入职月数:'|| round((sysdate-hire\_date)/30) FROM employees;

SELECT '入职年数:'|| round((sysdate-hire\_date)/30/12) FROM employees;

--months\_between:两个日期之间间隔多少个月

SELECT months\_between(sysdate,'1-1月-2000') FROM dual;

SELECT '入职月数:'|| round(months\_between(SYSDATE,hire\_date),2) FROM employees;

SELECT hire\_date-1 FROM employees;

--查询员工入职日期提前1个月

SELECT hire\_date-30 FROM employees;

--查询员工入职日期提前1年

SELECT hire\_date-365 FROM employees;

--入职时间大于20个月的员工信息

SELECT \* FROM employees WHERE months\_between(sysdate,hire\_date)>20;

--add\_months给一个日期加减若干个月，返回一个新日期，N为整数表示加，负数表示减

SELECT '下一个月:'||add\_months(sysdate,1) FROM dual;

SELECT '上一个月:'||add\_months(sysdate,-1) FROM dual;

----入职时间大于20个月的员工信息

SELECT \* FROM employees WHERE add\_months(hire\_date,20)<SYSDATE;

SELECT \* FROM employees WHERE add\_months(SYSDATE,-20)>hire\_date;

--next\_day (日期，星期几)返回以指定日期为准，下一个最近的星期几的日期

SELECT next\_day(SYSDATE,'星期一') FROM dual;

SELECT next\_day(SYSDATE,2) FROM dual;-- 周日第一天

--last\_day返回指定日期的月最后一天的日期

SELECT last\_day(SYSDATE) FROM dual;

SELECT last\_day('1-1月-2018') FROM dual;

--round (日期[,日期单位])对日期进行四舍五入

SELECT round(SYSDATE) FROM dual;--默认以日为单位

SELECT round(SYSDATE,'month') FROM dual;--默认以月为单位

SELECT round(SYSDATE,'year') FROM dual;--默认以年为单位

SELECT round(SYSDATE,'day') FROM dual;--默认以星期为单位

--trunc (日期[,日期单位])对日期进行截断

SELECT trunc(SYSDATE) FROM dual;--默认以日为单位

SELECT trunc(SYSDATE,'month') FROM dual;--默认以月为单位

SELECT trunc(SYSDATE,'year') FROM dual;--默认以年为单位

SELECT trunc(SYSDATE,'day') FROM dual;--默认以星期为单位

--数字和日期不能做任何转换，无论是显式还是隐式

--自动类型转换

SELECT '1'+'2' FROM dual;--字符串自动转成数

SELECT 'hello'||1 FROM dual;--数自动转成字符串

--查询入职员工在1987-1-1之前的

SELECT \* FROM employees WHERE hire\_date<'1-1月-1997';--字符串自动转成日期

--to\_char(日期|数字,'模式')：把一个日期或者数字按照指定模式转换为字符串

--日期===>字符串

SELECT to\_char(SYSDATE,'yyyy-mm-dd') FROM dual;

SELECT to\_char(SYSDATE,'yyyy-mm-dd hh:mi:ss') FROM dual;

SELECT to\_char(SYSDATE,'yyyy-mm-dd hh24:mi:ss') FROM dual;

SELECT to\_char(SYSDATE,'yyyy"年"mm"月"dd"日"') FROM dual;--""双引号转义

--获得年份

SELECT to\_char(SYSDATE,'yyyy') FROM dual;

--获得月份

SELECT to\_char(SYSDATE,'mm') FROM dual;

--获得日期

SELECT to\_char(SYSDATE,'dd') FROM dual;

--获得星期

SELECT to\_char(SYSDATE,'d') FROM dual;--获得是数

SELECT to\_char(SYSDATE,'day') FROM dual;--获得是中文星期几

--查询员工入职日期在7号之前,是周3的员工信息

SELECT \* FROM employees WHERE to\_char(hire\_date,'dd')<7 AND to\_char(hire\_date,'d')=4;

SELECT employee\_id,to\_char(hire\_date,'fmyyyy-mm-dd') FROM employees;--自动去除无效的0

--数===>字符串 0 必有 9 有无均可

SELECT to\_char(999999.9,'99990.00') FROM dual;--转换失败 整数位超过5位 99990.00 代表是必有1位整数 整数位最大5位,两位小数

SELECT to\_char(999.9,'99990.00') FROM dual;-- 999.90

SELECT to\_char(999.99999,'99990.00') FROM dual;-- 1000.00

SELECT to\_char(999.99499,'99990.00') FROM dual;-- 999.99

SELECT to\_char(999.11111,'99990.00') FROM dual;-- 999.11

SELECT to\_char(999.11111,'L99990.00') FROM dual;-- ￥999.11 L 本地货币符

SELECT to\_char(00999.11111,'99990.00') FROM dual;-- 999.11

--to\_date(日期字符串,'模式')： 把一个日期字符串按照指定模式解析为一个日期型

SELECT \* FROM employees WHERE hire\_date<'1997-1-1';--dd-mon-rr

SELECT \* FROM employees WHERE hire\_date<to\_date('1997-1-1','yyyy-mm-dd');

--to\_number(数字字符串,'模式')：把一个字符串解析为一个数字型

SELECT to\_number('12.33367','99.99999') FROM dual;

SELECT to\_number('￥12.33367','L99.99999') FROM dual;

9.--通用函数nvl/nvl2

--nvl (参数1，参数2)：如果参数1不为空，返回参数1，如果为空，返回参数2

SELECT nvl(NULL,1) FROM dual;

SELECT nvl(2,1) FROM dual;

SELECT nvl('',1) FROM dual;

--查询员工的年薪 (工资+奖金) \*12

SELECT e.\*,e.salary\*(1+ nvl(e.commission\_pct,0))\*12 "年薪" FROM employees e;

--nvl2 (参数1，参数2，参数3)：如果参数1不为空，返回参数2，如果为空，返回参数3

SELECT nvl2(null,1,2) FROM dual;

SELECT nvl2(3,1,2) FROM dual;

--查询员工的年薪 (工资+奖金) \*12

SELECT e.\*, e.salary\*12\*(nvl2(e.commission\_pct,e.commission\_pct,0)+1) FROM employees e;

--nullif (参数1，参数2)：参数1不等于参数2，返回参数1，如果相等，返回空

SELECT nullif(1,2) FROM dual;

SELECT nullif(1,1) FROM dual;

--last\_name 与 first\_name 字符数一致的员工信息

SELECT \* FROM employees WHERE nullif(length(last\_name),length(first\_name)) IS NULL ;--is null 判断是否为null

--coalesce (参数1，参数2，……，参数N)：返回第一个非空值，如果都为空，则返回空

SELECT coalesce(null,'','1','2','3','4','5','6','7') FROM dual;

SELECT coalesce(NULL,1,2,3,4,56) FROM dual;

--查询员工的年薪 (工资+奖金) \*12

SELECT salary\*(COALESCE(commission\_pct,0)+1)\*12 FROM employees;

--条件判断

10.CASE 表达式

--等值判断和非等值判断

--等值判断

/\*case 值

when 条件表达式1 then 返回值1

when 条件表达式2 then 返回值2

...

[else 默认返回值]

end

\*/

-- 1 查询余额 2 业务办理 3.取消业务 4.人工服务 其他 按错了

SELECT

CASE &num WHEN 1 THEN '查询余额'

WHEN 2 THEN '业务办理'

WHEN 3 THEN '取消业务'

WHEN 4 THEN '人工服务'

ELSE '按错了'

END

FROM dual;

--显示员工的部门信息 50 开发部 60 人事部 70 销售部 80 后勤部 其他显示空

SELECT e.\*,

CASE e.department\_id

WHEN 50 THEN '开发部'

WHEN 60 THEN '人事部'

WHEN 70 THEN '销售部'

WHEN 80 THEN '后勤部'

ELSE NULL

END

FROM employees e ;

--范围判断

/\*case

when 条件表达式1 then 返回值1

when 条件表达式2 then 返回值2

...

[else 默认返回值]

end

\*/

--60 以下不及格 60~70 及格 70~80 中等 80~90 良好 90 ~100 优秀

SELECT

CASE

WHEN &score <60 THEN '不及格'

WHEN &score<70 THEN '及格'

WHEN &score <80 THEN '中等'

WHEN &score <90 THEN '良好'

WHEN &score<=100 THEN '优秀'

ELSE '成绩有误'

END

FROM dual;

--显示工资 10000 优秀 5000 以上 将就活 5000 以下 活不起

SELECT salary,

CASE

WHEN salary>10000 THEN '优秀'

WHEN salary>=5000 THEN '将就活'

WHEN salary<5000 THEN '活不起'

END

FROM employees;

--DECODE 函数等值判断decode(表达式，值1，返回值1，值2，返回值2，...[,默认返回值])

-- 1 查询余额 2 业务办理 3.取消业务 4.人工服务 其他 按错了

SELECT decode(&num,1,'查询余额',2,'业务办理',3,'取消业务',4,'人工服务','按错了') FROM dual;

--显示员工的部门信息 50 开发部 60 人事部 70 销售部 80 后勤部 其他显示空

SELECT e.\*,decode(e.department\_id,50,'开发部',60,'人事部',70,'销售部',80,'后勤部',NULL) FROM employees e

# 4.函数练习

--练习

--1 找出每个月倒数第三天受雇的员工（如：2009-5-29）

SELECT \*FROM employees WHERE hire\_date-last\_day(hire\_date)=-2;

SELECT \*FROM employees WHERE hire\_date=last\_day(hire\_date)-2;

--2 找出25年前雇的员工

SELECT \* FROM employees WHERE SYSDATE-hire\_date>25\*365;

SELECT \* FROM employees WHERE months\_between(SYSDATE,hire\_date) > 25 \* 12;

SELECT \* FROM employees WHERE add\_months(hire\_date,12\*25) < SYSDATE;

--3 所有员工名字前加上Dear ,并且名字首字母大写

SELECT e.\*,'Dear'||initcap(last\_name) FROM employees e;

--4 找出姓名为5个字母的员工

SELECT \* FROM employees WHERE length(last\_name)=5;

SELECT \* FROM employees WHERE last\_name LIKE '\_\_\_\_\_';

--5 找出姓名中不带R这个字母的员工

SELECT \* FROM employees WHERE first\_name NOT LIKE '%R%' AND last\_name NOT LIKE '%R%' ;

SELECT \* FROM employees WHERE instr(lower(last\_name), 'r') <= 0;

--6 显示所有员工的姓名的第一个字

SELECT substr(last\_name,1,1) T,substr(first\_name,1,1)K FROM employees;

--7 显示所有员工，按名字降序排列，若相同，则按工资升序排序

SELECT employee\_id, last\_name,salary FROM employees e ORDER BY 2 DESC,3;

--8 假设一个月为30天，找出所有员工的日薪，不计小数

SELECT trunc(salary/30) FROM employees;

--9 找到2月份受雇的员工

SELECT \* FROM employees WHERE to\_char(hire\_date,'mm' )=2;

--10 列出员工加入公司的天数(四舍五入）

SELECT round(SYSDATE-hire\_date) FROM employees;

SELECT months\_between(SYSDATE,hire\_date)\*30 FROM employees;

--11 用1997年1月1日代替空的雇SYSDATE-hire\_date佣日期。

SELECT nvl(hire\_date,to\_date('1997年1月1日','yyyy"年"mm"月"dd"日"'))FROM employees;

--12 用“无”代替空的职务。

SELECT nvl(job\_id,'无') FROM employees;

--13 分别用case和decode函数列出员工所在的部门，

/\* deptno=10显示'部门10',

deptno=20显示'部门20'

deptno=30显示'部门30'

deptno=40显示'部门40'

否则为'其他部门'

\*/

SELECT CASE department\_id

WHEN 10 THEN '部门10'

WHEN 20 THEN '部门20'

WHEN 30 THEN '部门30'

WHEN 40 THEN '部门40'

ELSE '其他部门'

END

FROM employees

SELECT e.\*,decode(department\_id,10,'部门10',20,'部门20',30,'部门30',40,'部门40','其他部门') FROM employees e;

# 5.多表连接

--RDBMS：关系型数据库管理系统

--主键:表中唯一的标识 叫做 主键 例:用户编号 商品编号 订单编号

--外键:表中引入其他表中的主键叫做 外键 例:订单表中的 用户编号 商品编号

--主表:被引用的表 例:用户表 商品表

--从表:引用其它表 例:订单表

--表和表三种关系(从表:学生表:主键 :学号 外键:班级号)(主表 :班级表 主键:班级号)

--一对多:主表中一条信息对应从表中多条信息

--一对一:主表中一条信息对应从表中一条信息

--多对多:表中多条信息对应另一个表中多条信息===>必须借用第三张表

--三范式

/\*第一范式（1NF）

任何列不可分割(列值必须明确,不能表示多个信息)

第二范式（2NF）

非主属性非部分依赖于主关键字:===>必须有主键

第三范式（3NF）

属性不依赖于其它非主属性:===>不能引用其他表的非主键

\*/

--92语法

--99语法

1.92语法

--多表连接

**--笛卡尔集(叉集)**

SELECT employees.\*, departments.\* FROM employees,departments;

内连接：返回满足条件的数据

外连接：左外连接：左侧返回全部数据

右外链接：右侧返回全部数据

满外链接：返回所有数据

--表名.

SELECT employees.\*, departments.\*

FROM employees,departments

WHERE employees.department\_id=departments.department\_id;

--别名.

--显示员工的id,姓.名,工资,部门id,部门名称

SELECT employee\_id,last\_name ||'.'||first\_name ,salary,e.department\_id,department\_name

FROM employees e,departments d

WHERE e.department\_id=d.department\_id;

--显示员工的id,姓.名,工资,部门id,部门名称,职务id,职务名称,职务最小工资,最大工资

SELECT e.employee\_id,e.last\_name||'.'||e.first\_name,e.salary,d.department\_id,d.department\_name,e.job\_id,j.job\_title,j.min\_salary,j.max\_salary

FROM employees e, departments d,jobs j

WHERE e.department\_id=d.department\_id AND e.job\_id=j.job\_id;

--显示员工的id,姓.名,工资,部门id,部门名称,部门街道地址,职务id,职务名称,职务最小工资,最大工资

SELECT e.employee\_id,

e.last\_name || '.' || e.first\_name,

e.salary,

d.department\_id,

d.department\_name,

l.street\_address,

e.job\_id,

j.job\_title,

j.min\_salary,

j.max\_salary

FROM employees e,

departments d,

jobs j,

locations l

WHERE e.department\_id = d.department\_id

AND e.job\_id = j.job\_id

AND d.location\_id = l.location\_id;

--显示员工的id,姓.名,工资,部门id,部门名称,洲 ,国家,城市 ,部门街道地址 ,职务id,职务名称,职务最小工资,最大工资

SELECT e.employee\_id,

e.last\_name || '.' || e.first\_name,

d.department\_id,

d.department\_name,

r.region\_name,

c.country\_name,

l.city,

l.street\_address,

j.job\_id,

j.job\_title,

j.min\_salary,

j.max\_salary

FROM employees e,

departments d,

jobs j,

locations l,

countries c,

regions r

WHERE e.department\_id = d.department\_id

AND e.job\_id = j.job\_id

AND d.location\_id = l.location\_id

AND l.country\_id = c.country\_id

AND c.region\_id = r.region\_id;

--连接 n个表,至少需要 n-1个连接条件

--非等值连接

--查询员工的id, 名字,工资,工资的等级 ,等级范围

SELECT e.employee\_id,e.last\_name,e.salary,jg.grade\_level,'等级范围是'||jg.lowest\_sal||'=>'||jg.highest\_sal

FROM employees e,job\_grades jg

WHERE e.salary BETWEEN jg.lowest\_sal AND jg.highest\_sal;

--查询部门id,部门名称,部门经理id,部门经理姓名,部门经理的工资

SELECT d.department\_id,

d.department\_name,

d.manager\_id,

m.last\_name || m.first\_name,

m.salary

FROM departments d,

employees m

WHERE d.manager\_id = m.employee\_id;

--查询部门id,部门名称,部门所在的城市,街道,部门经理id,部门经理姓名,部门经理的工资,部门经理工资的等级

SELECT d.department\_id,

d.department\_name,

l.city,

l.street\_address,

m.employee\_id,

m.last\_name || m.first\_name,

m.salary,

jg.grade\_level

FROM departments d,

locations l,

job\_grades jg,

employees m

WHERE d.location\_id = l.location\_id

AND d.manager\_id = m.employee\_id

AND m.salary BETWEEN jg.lowest\_sal AND jg.highest\_sal;

--自连接

--查询员工的id ,姓名,员工的工资,员工的工资的等级,员工的管理者id,管理者姓名,管理者工资,管理者工资等级

SELECT e.employee\_id "员工id",

e.last\_name || e.first\_name "员工姓名",

e.salary "员工工资",

jg1.grade\_level "员工工资等级",

e.manager\_id "管理者id",

m.last\_name || m.first\_name "管理者名字",

m.salary "管理者工资",

jg.grade\_level "管理者工资等级"

FROM employees e,

employees m,

job\_grades jg,

job\_grades jg1

WHERE e.manager\_id = m.employee\_id

AND m.salary BETWEEN jg.lowest\_sal AND jg.highest\_sal

AND e.salary BETWEEN jg1.lowest\_sal AND jg1.highest\_sal;

--左连接:左侧的表数据全部显示,(+)放在右侧

--员工信息,部门名称

SELECT e.\*,

d.department\_name

FROM employees e,

departments d

WHERE e.department\_id= d.department\_id(+);

--部门信息,及部门经理的名字

SELECT d.\*,m.last\_name||m.first\_name

FROM departments d,employees m

WHERE d.manager\_id=m.employee\_id(+);

--右连接:右侧的表数据全部显示,(+)放在左侧

--所有员工信息,部门名称

SELECT e.\*,

d.department\_name

FROM employees e,

departments d

WHERE d.department\_id(+)=e.department\_id;

--所有部门信息,及部门经理的名字

SELECT d.\*,m.last\_name||m.first\_name

FROM departments d,employees m

WHERE m.employee\_id(+)=d.manager\_id;

--查询所有员工的id,员工的名字,及员工的管理者名字

SELECT e.employee\_id,

e.last\_name,

m.last\_name

FROM employees e,

employees m

WHERE e.manager\_id = m.employee\_id(+);

---1.显示所有员工的姓名，部门号和部门名称。

SELECT e.last\_name||e.first\_name,e.department\_id,d.department\_name

FROM employees e,departments d

WHERE e.department\_id=d.department\_id(+);

--2.查询90号部门员工的job\_id和90号部门的location\_id

SELECT e.job\_id,d.location\_id

FROM employees e,departments d

WHERE e.department\_id=d.department\_id AND d.department\_id=90;

--3.选择所有有奖金的员工的last\_name , department\_name , location\_id , city

SELECT e.last\_name,d.department\_name,l.location\_id,l.city

FROM employees e,departments d,locations l

WHERE e.department\_id=d.department\_id(+) AND e.commission\_pct IS NOT NULL AND d.location\_id=l.location\_id(+);

--4.选择在Toronto工作的员工的last\_name , job\_id , department\_id , department\_name

SELECT e.last\_name,e.job\_id,e.department\_id,d.department\_name

FROM employees e,departments d,locations l

WHERE e.department\_id=d.department\_id AND d.location\_id=l.location\_id AND initcap(l.city)='Toronto';

--5.选择所有员工的姓名，员工号，以及他的管理者的姓名和员工号

SELECT e.last\_name,e.employee\_id,e.manager\_id,m.last\_name

FROM employees e,employees m

WHERE e.manager\_id=m.employee\_id(+);

/\*--1.显示所有员工的姓名，所有部门号和部门名称。

SELECT e.last\_name||e.first\_name,e.department\_id,d.department\_name

FROM employees e,departments d

WHERE e.department\_id(+)=d.department\_id(+);

\*/

--显示所有员工的姓名，员工的工资,部门号和部门名称,部门升序排列 。

SELECT e.last\_name||e.first\_name,e.salary,e.department\_id,d.department\_name

FROM employees e,departments d

WHERE e.department\_id=d.department\_id(+)

ORDER BY e.department\_id;

--查询名中有\_的信息 ESCAPE 后面定义的是转义字符 1个

SELECT \* FROM employees WHERE first\_name LIKE '%\\_%' ESCAPE '\';

--查询名中有%的信息

SELECT \* FROM employees WHERE first\_name LIKE '%a%%' ESCAPE 'a';

2.99语法

--笛卡尔集

SELECT e.\*,d.\*

FROM employees e

CROSS JOIN departments d

--内连接: inner 可以省略的

SELECT e.\*,d.\*

FROM employees e

INNER JOIN departments d ON e.department\_id=d.department\_id;

--查询员工工资大于5000的员工编号,名称,工资,部门名称

SELECT e.employee\_id,e.last\_name,e.salary,d.department\_name

FROM employees e

INNER JOIN departments d ON e.department\_id=d.department\_id

WHERE e.salary>5000;

--外连接 OUTER 可省略

--查询所有的员工信息

SELECT e.\*,d.\*

FROM employees e

LEFT OUTER JOIN departments d ON e.department\_id=d.department\_id;

--查询所有的部门信息

SELECT e.\*,d.\*

FROM employees e

RIGHT OUTER JOIN departments d ON e.department\_id=d.department\_id;

---1.显示所有员工的姓名，部门号和部门名称。

SELECT e.last\_name||e.first\_name,e.department\_id,d.department\_name

FROM employees e

LEFT JOIN departments d ON e.department\_id=d.department\_id;

--2.查询90号部门员工的job\_id和90号部门的location\_id

SELECT e.job\_id,d.location\_id

FROM employees e

JOIN departments d ON e.department\_id=d.department\_id

WHERE e.department\_id=90;

--3.选择所有有奖金的员工的last\_name , department\_name , location\_id , city

SELECT e.last\_name,d.department\_name,l.location\_id,l.city

FROM employees e

LEFT JOIN departments d ON e.department\_id=d.department\_id

LEFT JOIN Locations l ON d.location\_id =l.location\_id

WHERE e.commission\_pct IS NOT NULL;

--4.选择在Toronto工作的员工的last\_name , job\_id , department\_id , department\_name

SELECT e.last\_name,e.job\_id,e.department\_id,d.department\_name

FROM employees e

JOIN departments d ON e.department\_id=d.department\_id

JOIN locations l ON d.location\_id=l.location\_id

WHERE initcap(l.city)='Toronto';

--5.选择所有员工的姓名，员工号，以及他的管理者的姓名和员工号

SELECT e.last\_name,e.employee\_id,e.manager\_id,m.last\_name

FROM employees e

LEFT JOIN employees m ON e.manager\_id=m.employee\_id;

--满外连接两边显示全部数据

--查询所有的部门,员工信息

SELECT e.\*,d.\*

FROM employees e

FULL OUTER JOIN departments d ON e.department\_id=d.department\_id;

--显示员工的id,姓.名,工资,部门id,部门名称

SELECT e.employee\_id,e.last\_name||e.first\_name,e.salary,d.department\_id,d.department\_name

FROM employees e

JOIN departments d ON e.department\_id=d.department\_id

--显示员工的id,姓.名,工资,部门id,部门名称,职务id,职务名称,职务最小工资,最大工资

SELECT e.employee\_id,e.last\_name||e.first\_name,e.salary,e.department\_id,d.department\_name,e.job\_id,j.job\_title,j.min\_salary,j.max\_salary

FROM employees e

JOIN departments d ON e.department\_id=d.department\_id

JOIN jobs j ON e.job\_id=j.job\_id;

--显示员工的id,姓.名,工资,部门id,部门名称,部门街道地址,职务id,职务名称,职务最小工资,最大工资

SELECT e.employee\_id,e.last\_name||e.first\_name,e.salary,e.department\_id,d.department\_name,l.street\_address , e.job\_id,j.job\_title,j.min\_salary,j.max\_salary

FROM employees e

JOIN departments d ON e.department\_id=d.department\_id

JOIN jobs j ON e.job\_id=j.job\_id

JOIN locations l ON d.location\_id=l.location\_id;

--显示员工的id,姓.名,工资,部门id,部门名称,洲 ,国家,城市 ,部门街道地址 ,职务id,职务名称,职务最小工资,

SELECT e.employee\_id,

e.last\_name || e.first\_name,

e.salary,

e.department\_id,

d.department\_name,

r.region\_name,

c.country\_name,

l.city,

l.street\_address,

e.job\_id,

j.job\_title,

j.min\_salary,

j.max\_salary

FROM employees e

JOIN departments d ON e.department\_id = d.department\_id

JOIN jobs j ON e.job\_id = j.job\_id

JOIN locations l ON d.location\_id = l.location\_id

JOIN countries c ON l.country\_id = c.country\_id

JOIN regions r ON c.region\_id = r.region\_id

--查询员工的id, 名字,工资,工资的等级 ,

SELECT e.employee\_id,e.last\_name,e.salary,jg.grade\_level

FROM employees e

JOIN job\_grades jg ON e.salary BETWEEN jg.Lowest\_Sal AND jg.highest\_sal

--查询部门id,部门名称,部门经理id,部门经理姓名,

SELECT d.department\_id,d.department\_name,d.manager\_id,m.last\_name

FROM departments d

JOIN employees m ON d.manager\_id=m.employee\_id;

--查询部门id,部门名称,部门所在的城市,街道,部门经理id,部门经理姓名,部门经理的工资,

SELECT d.department\_id,d.department\_name,l.city,l.street\_address,d.manager\_id,m.last\_name,m.salary

FROM departments d

JOIN locations l ON d.location\_id = l.LOCATION\_id

JOIN employees m ON d.manager\_id=m.employee\_id;

--查询员工的id ,姓名,员工的工资,员工的工资的等级,员工的管理者id,管理者姓名,管理者工资,管理者工资等级

SELECT e.employee\_id,e.last\_name,e.salary,j.grade\_level,e.manager\_id,m.Last\_Name,m.salary,jg.grade\_level

FROM employees e

JOIN job\_grades j ON e.salary BETWEEN j.lowest\_sal AND j.highest\_sal

JOIN employees m ON e.manager\_id=m.employee\_id

JOIN job\_grades jg ON m.salary BETWEEN jg.lowest\_sal AND jg.highest\_sal;

--员工信息,部门名称

SELECT e.\*,d.department\_name

FROM employees e

JOIN departments d ON e.department\_id=d.department\_id;

--部门信息,及部门经理的名字

SELECT d.\*,m.last\_name

FROM departments d

JOIN employees m ON d.manager\_id = m.employee\_id;

--所有员工信息,部门名称

SELECT e.\*,d.department\_name

FROM employees e

LEFT JOIN departments d ON e.department\_id=d.department\_id;

--所有部门信息,及部门经理的名字

SELECT d.\*,m.last\_name

FROM departments d

LEFT JOIN employees m ON d.manager\_id = m.employee\_id;

--查询所有员工的id,员工的名字,及员工的管理者名字

SELECT e.employee\_id,e.last\_name,m.last\_name

FROM employees e

LEFT JOIN employees m ON e.manager\_id = m.employee\_id;

--2.查询90号部门员工的job\_id和90号部门的location\_id

SELECT e.job\_id,d.location\_id

FROM employees e

JOIN departments d ON e.department\_id = d.department\_id

WHERE d.department\_id=90;

--3.选择所有有奖金的员工的last\_name , department\_name , location\_id , city

SELECT e.last\_name,d.department\_name,d.location\_id,l.city

FROM employees e

LEFT JOIN departments d ON e.department\_id=d.department\_id

LEFT JOIN locations l ON d.location\_id = l.location\_id

WHERE e.commission\_pct IS NOT NULL;

--4.选择在Toronto工作的员工的last\_name , job\_id , department\_id , department\_name

SELECT e.last\_name,e.job\_id,e.department\_id,d.department\_name

FROM employees e

JOIN departments d ON e.department\_id=d.department\_id

JOIN locations l ON d.location\_id = l.location\_id

WHERE l.city = 'Toronto';

--5.选择所有员工的姓名，员工号，以及他的管理者的姓名和员工号

SELECT e.last\_name,e.employee\_id,m.last\_name,m.employee\_id

FROM employees e

LEFT JOIN employees m ON e.manager\_id = m.employee\_id;

--6.显示所有员工的姓名，员工的工资,部门号和部门名称,部门升序排列

SELECT e.last\_name,e.employee\_id,d.department\_id,d.department\_name

FROM employees e

LEFT JOIN departments d ON e.department\_id=d.department\_id

ORDER BY d.department\_id;

--2.查询部门编号，部门名称，部门经理编号，部门经理姓名，部门经理的职务编号，部门经理职务名称（显示所有部门 99语法）

SELECT d.department\_id,d.department\_name,d.manager\_id,m.last\_name,m.job\_id,j.job\_title

FROM departments d

LEFT JOIN employees m ON d.manager\_id = m.employee\_id

LEFT JOIN jobs j ON m.job\_id = j.job\_id;

--4.查询各部门员工姓名和他们的同事姓名

SELECT e.last\_name,t.last\_name,e.department\_id

FROM employees e

JOIN employees t ON e.department\_id=t.department\_id

WHERE t.employee\_id<>e.employee\_id;

# 6.作业

--查询员工姓和名字数相等的员工

SELECT \* FROM employees WHERE length(first\_name)=length(last\_name);

--查询last\_name以s结尾的员工（不用like）

SELECT \* FROM employees WHERE substr(last\_name,-1,1)='s';

--查询所有的员工姓和名，输出以下格式S.King

SELECT substr(first\_name,1,1)||'.'||last\_name FROM employees;

SELECT concat(concat(substr(first\_name,1,1),'.'),last\_name) FROM employees;

--查询所有的电话号码，把分隔符“点”换为“-”之后再输出

SELECT replace(phone\_number,'.','-') FROM employees;

--用户输入一个任意编号，查询此编号的员工

SELECT \* FROM employees WHERE employee\_id = &myid;

--用户输入一个关键字，查询last\_name包含此关键字的员工（不用like）

SELECT \* FROM employees WHERE instr(last\_name,'&myname')<>0;

--查询员工的编号，姓名，工资，部门编号，要求按照工资的升序进行排列

SELECT employee\_id,last\_name,salary,department\_id FROM EMPLOYEES ORDER BY salary ASC;

--工资降序排列

SELECT employee\_id,last\_name,salary,department\_id FROM EMPLOYEES ORDER BY salary DESC;

--查询员工工资大于5000的员工，以工资的降序进行排列

SELECT employee\_id,salary FROM employees WHERE salary>5000 ORDER BY salary DESC;

--查询员工信息，要求按照工资降序和job\_id,入职日期升序进行查询（多行排序）

SELECT \* FROM employees ORDER BY salary DESC,job\_id ASC ,hire\_date ASC;

--查询工龄超过20年员工的员工信息

SELECT \* FROM employees WHERE sysdate-hire\_date>20\*365;

SELECT \* FROM employees WHERE months\_between(sysdate,hire\_date)>20\*12;

SELECT \* FROM employees WHERE add\_months(hire\_date,20\*12)<SYSDATE;

SELECT \* FROM employees WHERE add\_months(SYSDATE,-20\*12)>hire\_date;

--查询员工编号，姓名，入职日期（yyyy-mm-dd）

SELECT employee\_id,last\_name,hire\_date FROM employees;

SELECT employee\_id,last\_name,to\_char(hire\_date,'fmyyyy-mm-dd') FROM employees;

--查询1993年12月31日之前入职的员工

SELECT \* FROM employees WHERE hire\_date <to\_date('1993年12月31日','yyyy"年"mm"月"dd"日"');

SELECT \* FROM employees WHERE to\_char(hire\_date,'yyyy')<=1993;

--查询17号之前入职的员工

SELECT \* FROM employees WHERE substr(hire\_date,1,2) <17;

SELECT \* FROM employees WHERE to\_char(hire\_date,'dd')<17;

--查询查询7月或者8月入职的员工

SELECT \* FROM employees WHERE to\_char(hire\_date,'mm') IN(7,8);

--查询1995年以前入职的员工

SELECT \* FROM employees WHERE hire\_date<'1-1月-1995';

SELECT \* FROM employees WHERE to\_char(hire\_date,'yyyy')<1995;

# 7.分组

--单行函数 输入一行 结果一行

--多行函数 输入多行 结果一行

--组函数是多行函数的一种

--查询员工表中的平均工资

SELECT AVG(salary) "平均工资" FROM employees;

--总和sum

--支出工资

SELECT SUM(salary) "总工资" FROM employees;

--计数

SELECT SUM(1) "总工资" FROM employees;

--count 计数

--有多少个员工

SELECT COUNT(employee\_id) FROM employees;

SELECT COUNT(\*) FROM employees;

--显示有奖金的员工数量

SELECT COUNT(commission\_pct) FROM employees;

SELECT SUM(nvl2(commission\_pct,1,0)) FROM employees;

--显示没有奖金的员工数量

SELECT COUNT(employee\_id)-COUNT(commission\_pct) FROM employees;

SELECT SUM(nvl2(commission\_pct,0,1)) FROM employees;

--max 最大值

--显示最高工资

SELECT MAX(salary) FROM employees;

--min 最小值

--显示最低工资

SELECT MIN (salary) FROM employees;

--count avg sum max min 都忽略空值

--查询员工数,最大工资 ,最小工资,平均工资,工资总和

SELECT COUNT(employee\_id) "人数",

MAX(salary) "最大工资",

MIN(salary) "最小工资",

AVG(salary) "平均工资",

SUM(salary) "工资总和"

FROM employees;

--自然顺序

SELECT MAX(last\_name ) FROM employees;--Zlotkey 排序后的最大值

SELECT MIN(last\_name ) FROM employees;--Abel 排序后的最小值

--MIN MAX 参数 可以是数 字符串 日期

--SUM AVG 参数 只能是数

--查询员工表中有多少个部门 DISTINCT去除重复行

SELECT COUNT (DISTINCT department\_id) FROM employees;

--每个部门的平均工资

SELECT department\_id,AVG (salary) FROM employees

GROUP BY department\_id

--统计每个部门的平均工资/最大工资/最小工资/每个部门的人数

SELECT department\_id, trunc(avg(salary),2),MAX(salary),MIN(salary),COUNT (employee\_id)

FROM employees

GROUP BY department\_id

--统计每个部门的平均工资/最大工资/最小工资/每个部门的人数 ,按照部门降序排列

SELECT department\_id, trunc(avg(salary),2),MAX(salary),MIN(salary),COUNT (employee\_id)

FROM employees

GROUP BY department\_id

ORDER BY department\_id DESC

--查询每个部门的人数 显示 部门编号及名称

SELECT d.department\_id,d.department\_name, COUNT(e.employee\_id) "人数"

FROM departments d

LEFT JOIN employees e ON e.department\_id = d.department\_id

GROUP BY d.department\_id ,d.department\_name

--查询部门号大于50号的人数 显示 部门编号及名称 ,按照部门号升序排列

SELECT d.department\_id,d.department\_name, COUNT(e.employee\_id) "人数"

FROM departments d

LEFT JOIN employees e ON e.department\_id = d.department\_id

WHERE d.department\_id>50

GROUP BY d.department\_id ,d.department\_name

ORDER BY d.department\_id

--组函数做判断不能放在where 后 放在 having 后 having 必须与group by 连用 写在group by 后面

--查询部门人数大于3人的部门编号及人数

SELECT department\_id,COUNT (employee\_id) FROM employees

GROUP BY department\_id HAVING COUNT(employee\_id) >=3

--查询部门号大于50号的人数 并且人数>=3的部门编号及名称 ,按照部门号升序排列

SELECT d.department\_id,d.department\_name, COUNT(e.employee\_id) "人数"

FROM departments d

LEFT JOIN employees e ON e.department\_id = d.department\_id

WHERE d.department\_id>50

GROUP BY d.department\_id ,d.department\_name HAVING COUNT (e.employee\_id)>=3

ORDER BY d.department\_id

--from where group by having select order by

--查询部门的最大工资大于7000的部门编号及名称,按照工资降序排序

SELECT d.department\_id,d.department\_name,MAX (e.salary) "最高工资"

FROM employees e

JOIN departments d ON e.department\_id =d.department\_id

GROUP BY d.department\_id,d.department\_name HAVING MAX(e.salary) >7000

ORDER BY "最高工资" DESC;

--查询每年入职的员工人数

SELECT to\_char(hire\_date,'yyyy') ,COUNT(employee\_id) FROM employees

GROUP BY to\_char(hire\_date,'yyyy')

--查询每月入职的员工人数

SELECT to\_char(hire\_date,'fmmm') ,COUNT(employee\_id) FROM employees

GROUP BY to\_char(hire\_date,'fmmm')

--海量排序:数据过大时不建议排序

SELECT e.\* FROM employees e,employees e1,employees e2,employees e3

--部门平均工资中最高的工资

SELECT MAX (AVG (salary)) FROM employees GROUP BY department\_id;

--多个分组条件 group by 后面 不同记录的字段值必须一致才叫做一组

SELECT department\_id,job\_id

FROM employees GROUP BY department\_id ,job\_id

--查询每个部门的人数--同一个部门

SELECT department\_id ,count (employee\_id) FROM employees

GROUP BY department\_id;

--查询每个职务的人数--同一个职务

SELECT job\_id ,count (employee\_id) FROM employees

GROUP BY job\_id;

--查询每个部门职务的人数--同一个部门同一个职务

SELECT department\_id,job\_id,COUNT(employee\_id)

FROM employees GROUP BY department\_id ,job\_id

# 8.分组练习

--7.查询统计每年入职的人数（年份，入职人数）

SELECT to\_char(hire\_date,'yyyy'),COUNT(e.employee\_id)

FROM employees e

GROUP BY to\_char(hire\_date,'yyyy');

--7-1.查询公司中按年份月份统计各地的录用职工数量

SELECT to\_char(hire\_date,'yyyy-mm'),d.location\_id,COUNT(e.employee\_id)

FROM employees e

JOIN departments d ON e.department\_id= d.department\_id

GROUP BY to\_char(hire\_date,'yyyy-mm'),d.location\_id;

--8.统计每个城市的员工人数（城市名称，人数）

SELECT COUNT (e.employee\_id),l.city

FROM employees e

JOIN departments d ON e.department\_id=d.department\_id

JOIN locations l ON d.location\_id=l.location\_id

GROUP BY l.city

--9.统计每个城市的员工平均工资（城市名称，平均工资）

SELECT city,AVG(salary)

FROM employees e

JOIN departments d ON e.department\_id=d.department\_id

JOIN locations l ON d.location\_id=l.location\_id

GROUP BY l.city

--10.显示非销售人员工作名称以及从事同一工作雇员的月工资的总和，并且要满足从事同一工作的雇员的月工资合计大于5000，输出结果按每月工资的合计降序排序

SELECT j.job\_title,SUM(e.salary)

FROM employees e

JOIN jobs j ON e.job\_id=j.job\_id

WHERE j.job\_title NOT LIKE '%Sales%'

GROUP BY j.job\_title

HAVING sum(e.salary)>5000

ORDER BY SUM(e.salary) DESC;

--11.现在求出部门平均工资的最高工资额。

SELECT MAX (AVG (salary)) FROM employees GROUP BY department\_id;

--12.查询出部门平均工资在1500元以上的部门名称和平均工资。

SELECT round(AVG(salary),2),d.department\_name

FROM departments d

JOIN employees e ON d.department\_id = e.department\_id

GROUP BY d.department\_name HAVING AVG(salary)>1500;

--13.查询部门人数大于等于3人的部门的部门名称及最低工资和最高工资，按最高工资降序排序。

SELECT d.department\_name,count(e.employee\_id),MIN(e.salary),Max(e.salary)

FROM departments d

JOIN employees e ON e.department\_id=d.department\_id

GROUP BY d.department\_name

HAVING count(e.employee\_id)>=3

ORDER BY Max(e.salary);

--14.查询部门人数不少于3人的部门(部门名称,人数)

SELECT d.department\_name,count(e.employee\_id)

FROM departments d

JOIN employees e ON e.department\_id=d.department\_id

GROUP BY d.department\_name

HAVING count(e.employee\_id)>=3

--15.查询部门名称,及部门人数

SELECT d.department\_name,count(e.employee\_id)

FROM departments d

JOIN employees e ON e.department\_id=d.department\_id

GROUP BY d.department\_name

# 9.表管理

--DDL 数据定义语言 操作对象是表 自动提交事务,执行后无法撤回

--表名与字段名

/\*表名和列名:

必须以字母开头

必须在 1–30 个字符之间

必须只能包含 A–Z, a–z, 0–9, \_, $, 和 #

必须不能和用户定义的其他对象重名

必须 不能是Oracle 的保留字

\*/

/\*

create table 表名(

名字 类型 ,

名字 类型 ,

名字 类型 ,

...

名字 类型

);

\*/

--创建一个学生表 字段: 学号 名字 年龄 成绩保留两位小数

CREATE TABLE student(

s\_id NUMBER ,

s\_name varchar2(40),

s\_age NUMBER(3),

s\_score NUMBER (5,2)

);

--添加性别列

ALTER TABLE student

ADD s\_sex VARCHAR2(2);

--修改名字的空间

ALTER TABLE student

MODIFY s\_name VARCHAR2 (60);

--删除性别字段

ALTER TABLE student

DROP COLUMN s\_sex;

--查询学生数据

SELECT \* FROM student

--清除数据

TRUNCATE TABLE student;

--删除表

DROP TABLE student;

--修改表名

ALTER TABLE student1 RENAME TO student

RENAME student1 TO student;

--修改字段名字

ALTER TABLE student

RENAME COLUMN s\_name1 TO s\_name;

# 10.子查询

--比Abel的工资高的员工信息

SELECT \*

FROM employees

WHERE salary > (

SELECT salary

FROM employees

WHERE last\_name = 'Abel');

--查询与101号员工的手机尾号一致的员工信息

SELECT \*

FROM employees

WHERE employee\_id <> 101

AND substr(phone\_number,

-1) = (SELECT substr(phone\_number,

-1)

FROM employees

WHERE employee\_id = 101);

--查询比103员工管理者工资要高的员工信息

SELECT \*

FROM employees

WHERE salary > (

SELECT salary

FROM employees

WHERE employee\_id = (

SELECT manager\_id

FROM employees

WHERE employee\_id = 103));

--查询高于平均工资的员工信息

SELECT \*

FROM employees

WHERE salary > (

SELECT AVG(salary)

FROM employees)

--查询比206员工所在部门的部门经理的管理者的工资要高的员工信息

SELECT \*

FROM employees

WHERE salary > (

SELECT salary

FROM employees

WHERE employee\_id = (

SELECT manager\_id

FROM employees

WHERE employee\_id = (

SELECT manager\_id

FROM departments

WHERE department\_id = (

SELECT department\_id

FROM employees

WHERE employee\_id = 206))));

--子查询 : 嵌套查询 一个查询语句中包含另一个查询语句 (外层查询叫做主查询,内层子查询)

--放在的位置:where 或者是 having 后 单行单列,多行单列

-- select 后 单行单列

-- from 后 多行多列

-- order by后 单行单列

--员工的名字/工资/最高工资

SELECT last\_name,salary,(SELECT MAX(salary) FROM employees) FROM employees;

--查询员工表中前5条数据

SELECT ROWNUM, e.\* FROM employees e WHERE ROWNUM<=5;

--查询员工表中工资最高的前5条数据

SELECT ROWNUM, e.\* FROM employees e WHERE ROWNUM<=5 ORDER BY salary DESC ;

SELECT \* FROM (SELECT \* FROM employees ORDER BY salary DESC )

WHERE ROWNUM<=5

--查询员工表中工资最高的前5~10条数据

SELECT \*

FROM (SELECT \*

FROM (SELECT \*

FROM employees

ORDER BY salary DESC)

WHERE rownum <= 10

ORDER BY salary ASC)

WHERE rownum <= 5

--查询员工表中工资最高的前5~10条数据

SELECT \*

FROM (SELECT rownum rn,

e1.\*

FROM (SELECT \*

FROM employees

ORDER BY salary DESC) e1)

WHERE rn <= 10

AND rn >= 5

--查询员工表中5~10条

SELECT \*

FROM (

SELECT rownum rn,

e.\*

FROM employees e)

WHERE rn BETWEEN 5 AND 10;

--查询员工表10~15的数据

SELECT \*

FROM (

SELECT rownum rn,

e.\*

FROM employees e)

WHERE rn BETWEEN 10 AND 15;

--查询员工表中名字排在前10~15的数据

SELECT \*

FROM (SELECT rownum rn,

e.\*

FROM (SELECT \*

FROM employees

ORDER BY last\_name) e)

WHERE rn BETWEEN 10 AND 15;

--查询是部门经理的员工信息

SELECT \* FROM employees WHERE employee\_id IN (

SELECT manager\_id FROM departments) = 101 OR =102 OR =103

--查询不是部门经理的员工信息

SELECT \* FROM employees WHERE employee\_id NOT IN (

SELECT nvl(manager\_id,-1) FROM departments)

-- any 任何一个 或者 or

--all 所有 与 and

--查询员工的工资大于80号部门任何一个工资的信息(工资大于80号部门最小一个)

SELECT \*

FROM employees

WHERE salary > ANY (SELECT salary

FROM employees

WHERE department\_id = 80);

SELECT \*

FROM employees

WHERE salary > (SELECT MIN (salary)

FROM employees

WHERE department\_id = 80);

--查询员工的工资大于80号部门所有员工的工资的信息(工资大于80号部门最大一个)

SELECT \*

FROM employees

WHERE salary > ALL (SELECT salary

FROM employees

WHERE department\_id = 80);

SELECT \*

FROM employees

WHERE salary > (SELECT MAX (salary)

FROM employees

WHERE department\_id = 80);

--0.查询电话号码尾数和107员工相同的员工

SELECT \*

FROM employees

WHERE employee\_id <>107 AND instr(phone\_number,(

SELECT substr(phone\_number,

-1)

FROM employees

WHERE employee\_id = 107),length(phone\_number))>0;

--1.查询与每个部门的最小工资一致的员工信息

SELECT \*

FROM employees

WHERE salary IN (SELECT MIN(salary)

FROM employees

GROUP BY department\_id);

-- 普通子查询 相关子查询

--执行顺序 子先主后 交互执行

--执行的次数 1次 一致

--结果 主查询需要用到子查询结果 相互

--查询员工的信息及所在部门的平均工资

SELECT e.\*,

nvl((SELECT AVG(salary)

FROM employees e1

WHERE e1.department\_id = e.department\_id),

e.salary)

FROM employees e

--查询员工的信息及所在部门的人数

SELECT e.\*,

(SELECT COUNT(employee\_id)

FROM employees

WHERE department\_id = e.department\_id)

FROM employees e

--查询员所在部门的人数不大于4人的员工信息

SELECT e.\*

FROM employees e

WHERE 4>= (SELECT COUNT(employee\_id)

FROM employees

WHERE department\_id = e.department\_id);

--查询每个部门信息及部门人数

SELECT d.\*,

(SELECT COUNT(employee\_id)

FROM employees

WHERE department\_id = d.department\_id)

FROM departments d;

--查询工资是正数第四名的员工信息

SELECT \*

FROM (SELECT ROWNUM rn,

e.\*

FROM (SELECT \*

FROM employees

ORDER BY salary DESC) e )

WHERE rn = 4;

--查询换过工作的员工信息

SELECT \* FROM employees WHERE employee\_id IN(SELECT DISTINCT employee\_id FROM job\_history);

--查询换工作数超过2次的员工信息

SELECT \*

FROM employees e

WHERE 2 <= (SELECT COUNT(employee\_id)

FROM job\_history

WHERE e.employee\_id = employee\_id);

--EXISTS 判断是否有返回结果 :有 真 没有 假

SELECT \* FROM employees WHERE EXISTS (SELECT 'X' FROM employees WHERE 1=1);

--显示是部门经理的信息

SELECT \* FROM employees e WHERE EXISTS (SELECT 'x' FROM departments WHERE e.employee\_id=manager\_id);

--显示不是部门经理的信息

SELECT \* FROM employees e WHERE NOT EXISTS (SELECT 'x' FROM departments WHERE e.employee\_id=manager\_id);

--显示是管理者的信息

SELECT \* FROM employees e WHERE EXISTS (SELECT 'a' FROM employees WHERE e.employee\_id = manager\_id);

--显示不是管理者的信息

SELECT \* FROM employees e WHERE NOT EXISTS (SELECT 'a' FROM employees WHERE e.employee\_id = manager\_id);

--查询员工信息按照部门名称排序

SELECT \* FROM employees e

LEFT JOIN departments d ON e.department\_id =d.department\_id

ORDER BY d.department\_name

--查询员工信息按照部门名称排序

SELECT \* FROM employees e ORDER BY (SELECT department\_name FROM departments WHERE e.department\_id=department\_id) DESC;

--连接查询效率高于子查询:分页 前n条数据 前 n~m条数据

--0.查询电话号码尾数和107员工相同的员工

SELECT \*

FROM employees

WHERE employee\_id <>107 AND instr(phone\_number,(

SELECT substr(phone\_number,

-1)

FROM employees

WHERE employee\_id = 107),length(phone\_number))>0;

--1.查询与每个部门的最小工资一致的员工信息

SELECT \*

FROM employees

WHERE salary IN (SELECT MIN(salary)

FROM employees

GROUP BY department\_id);

--2.查询是部门经理的员工（连接查询实现）

SELECT \* FROM employees e

JOIN departments d ON e.employee\_id = d.manager\_id;

--3.查询不是部门经理的员工（连接查询实现）

SELECT \* FROM employees WHERE employee\_id NOT IN(SELECT nvl(manager\_id,-1) FROM departments );

SELECT \* FROM employees e

LEFT JOIN departments d ON e.department\_id = d.department\_id

WHERE e.employee\_id <>d.manager\_id OR d.manager\_id IS NULL;

--4.查询比202号员工所在的部门的部门经理的工资还高的员工信息

SELECT \*

FROM employees

WHERE salary > (

SELECT salary

FROM employees

WHERE employee\_id =

(SELECT manager\_id

FROM departments

WHERE department\_id =

(SELECT department\_id

FROM employees

WHERE employee\_id = 202)));

--5.查询每个部门中入职最早的员工信息

SELECT \* FROM employees WHERE hire\_date IN(SELECT MIN(hire\_date) FROM employees GROUP BY department\_id) AND department\_id IS NOT NULL;

--6.查询每个员工编号，姓名，工资，部门编号，本部门工资总和(内外交互式相关子查询)

SELECT e.\*,(SELECT SUM(salary) FROM employees WHERE e.department\_id=department\_id) FROM employees e

--7.查询员工工资超过本部门平均工资的员工

SELECT \* FROM employees e WHERE salary >(SELECT AVG(salary) FROM employees WHERE e.department\_id=department\_id)

# 11.DML

--DML(数据操作语言) 对记录增加/修改/删除

--增加

/\*

insert into 表名 [(字段名,字段名,字段名,字段名...字段名)] values (字段值,字段值,字段值,字段值...字段值)

字段名与字段值必须保持一致,类型/个数/顺序

插入某几个字段,必须要列明字段名字

如果插入全部字段,可以省略不写

\*/

--创建表

CREATE TABLE student(

s\_id NUMBER ,

s\_name VARCHAR2 (20),

s\_age NUMBER (3),

s\_score NUMBER(5,2)

)

--插入数据

INSERT INTO student(s\_id,s\_name,s\_age,s\_score) VALUES (1,'tom',19,89.5);

INSERT INTO student VALUES (1,'tom',19,89.5);

--查询

SELECT \* FROM student;

--插入数据

INSERT INTO student(s\_id,s\_name,s\_age) VALUES (3,'tom',20);

--创建表并复制数据:emp表结构与employees 一致 并且复制全部数据

CREATE TABLE emp

AS SELECT \* FROM employees;

--只有表结构,无数据

CREATE TABLE newemp AS SELECT \* FROM employees WHERE 1=2;

--插入多条数据

INSERT INTO newemp SELECT \* FROM emp;

--修改

/\*

update 表名 set 字段=字段值,字段=字段值,字段=字段值,...字段=字段值 [where 条件]

没有where条件,会全部更新

\*/

UPDATE student SET s\_id = 2,s\_name = 'jack',s\_score = 90;

UPDATE student SET s\_id= 1,s\_name = 'tom',s\_score=89.5 WHERE s\_age = 19;

--删除

/\*

delete from 表名 [where 条件]

没有where条件,会全部删除掉

\*/

DELETE FROM student

DELETE FROM student WHERE s\_id = 2;

--创建菜品表 菜品id 菜品名称 菜品单价 菜品状态 点赞数

CREATE TABLE foods(

f\_id NUMBER,

f\_name VARCHAR2(20),

f\_price NUMBER(5,2),

f\_state NUMBER(1),

f\_like NUMBER

);

--插入5条数据

INSERT INTO foods VALUEs (1,'大鼻嘎嘎炒豆芽',12.5,0,0);

INSERT INTO foods VALUEs (2,'死孩子皮卷蛆',15,0,0);

INSERT INTO foods VALUEs (3,'大蛆钻豆腐',3,0,0);

INSERT INTO foods VALUEs (4,'鼻涕甩袖汤',19,0,0);

INSERT INTO foods VALUEs (5,'百年老痰',2.5,0,0);

--给3号菜加价50

UPDATE foods SET f\_price=f\_price+50 WHERE f\_id = 3;

--其他菜减价5块

UPDATE foods SET f\_price=f\_price-5 WHERE f\_id != 3;

--每一单菜点赞数加1

UPDATE foods SET f\_like=f\_like+1;

--将5号菜的状态给为售罄

UPDATE foods SET f\_state=1 WHERE f\_id = 5;

--删除掉状态是售罄

DELETE FROM foods WHERE f\_state=1;

SELECT \* FROM foods

# 12.事务与锁

--事务:避免数据灾难,保证数据的完整性

--事务特性Acid

--原子性:事务不可以分割,要么都执行,要么都不行执行

--一致性:由一个正确的状态转换另一个正确地状态

--隔离性:一个会话无法访问另一个未结束事务的脏数据

--持续性:一个会话一旦结束事务后,当前会话的数据会持久化服务器中,另一个会话则可以访问到数据

--删除19岁学生

DELETE FROM student WHERE s\_age = 19;

--开启事务组成:1.多个DML 2.一个DDL或DCL

--结束事务操作(TCL): 1.提交: 1.COMMIT, 2.DDL语句或DCL语句 3.正常退出会话EXIT

--2.回滚: 1.ROLLBACK 2.异常退出

--插入学生

INSERT INTO student VALUES(99,'rose',20,90);

INSERT INTO student VALUES(90,'july',21,91);

--COMMIT;

--ROLLBACK;

SELECT \* FROM student;

--删除EMP表

DROP TABLE emp;

# 13.约束

--约束:限制字段 DDL 定义语言

/\*

NOT NULL 不为空

UNIQUE 唯一值

PRIMARY KEY 主键 不能为空 唯一值

FOREIGN KEY 外键

CHECK 检查约束

\*/

CREATE TABLE emp

AS SELECT \* FROM employees WHERE ROWNUM<=10;

--修改first\_name 的不为null 约束

ALTER TABLE emp

MODIFY (first\_name NOT null);

--修改first\_name 的为null 约束

ALTER TABLE emp

MODIFY (first\_name NULL );

--把employee\_id 变成主键

/\*

ALTER TABLE 表名

ADD CONSTRAINTS 约束名字(主键名字) PRIMARY KEY (字段);

\*/

ALTER TABLE emp

ADD CONSTRAINTS emp\_empid\_pk PRIMARY KEY (employee\_id);

--删除主键

ALTER TABLE emp

DROP CONSTRAINTS emp\_empid\_pk;

/\*

ALTER TABLE 表名

ADD CONSTRAINTS 约束名字 UNIQUE (字段);

\*/

ALTER TABLE emp

ADD CONSTRAINTS emp\_uq UNIQUE (last\_name);

--删除唯一约束

ALTER TABLE emp

DROP CONSTRAINTS emp\_uq;

--检查约束

ALTER TABLE emp

ADD CONSTRAINTS emp\_salary\_check CHECK ( salary >2000);

--删除检查约束

ALTER TABLE emp

DROP CONSTRAINTS emp\_salary\_check;

CREATE TABLE dept

AS SELECT \* FROM departments

ALTER TABLE dept

ADD CONSTRAINTS dept\_deptid\_pk PRIMARY KEY (department\_id);

--外键

ALTER TABLE emp

ADD CONSTRAINTS emp\_deptid\_for FOREIGN KEY (department\_id) REFERENCES dept(department\_id)

--删除检查约束

ALTER TABLE emp

DROP CONSTRAINTS emp\_deptid\_for;

--创建表直接加入约束

CREATE TABLE food(

f\_id NUMBER PRIMARY key,

f\_name VARCHAR2 (20) NOT NULL UNIQUE ,

f\_price NUMBER (5,2) CHECK (f\_price BETWEEN 0 AND 100),

f\_state NUMBER (1) CHECK (f\_state IN (0,1)),

f\_like NUMBER CHECK (f\_like>=0),

f\_dept\_id NUMBER (4) REFERENCES dept(department\_id)

)