

SQL基础知识

- SQL是结构化查询语言（Structures Query Language），专门用于数据存取、数据更新及数据库管理等操作,主要针对关系型数据库的语言

▼ SQL基础知识（一）

▼ 一、DCL 数据控制语言（DATA CONTROL LANGUAGE）

- ORACLE中系统预设的用户有：sys(管理员权限，非锁定)、system（管理员权限，非锁定）、scott（普通用户权限，锁定）

▼ （一）创建系统用户

- 1、创建系统用户语法：

create user 用户名 identified by 口令 [account lock / unlock]

代码演示：

```
create user 123 identified by admin account unlock
```

- 2、系统用户创建成功后，默认是锁定状态的，不能登录oracle系统，因为没有权限

▼ 3、ORACLE用户对数据库的权限

- 系统权限，比如create session / create table等
- 数据库对象权限，比如对表中数据进行增删改操作

▼ （二）数据库角色 role：若干个系统权限的集合

- 1、connect 角色：简单的权限，主要应用在临时用户，可与数据库建立对话，不需要创表的
- 2、resource 角色：更可靠和正式的数据库用户，可以创建表、序列、过程、触发器、索引等
- 3、DBA角色：拥有所有的系统权限，包括无限的系统空间和给其他系统用户授权的能力，system用户拥有DBA角色
- 4、一个普通用户，拥有connect 和resource角色就可以进行常规的数据库开发工作

▼ （三）授权相关语法

- 1、授权语法：GRANT 角色/权限 TO 用户/角色

代码演示：

```
GRANT RESOURCE TO 123;
```

```
GRANT CONNECT TO 123;
```

```
GRANT CREATE TABLE TO 123;
```

```
GRANT CREATE SESSION TO 123;
```

- 2、收回(撤销)权限：REVOKE 角色/权限 FROM 用户/角色

代码演示：

```
REVOKE RESOURCE FROM 123
```

- 3、修改用户密码：ALTER USER 用户名 IDENTIFIED BY 新密码

代码演示：

```
ALTER USER 123 IDENTIFIED BY 123456
```

- 4、 修改用户处于锁定（非锁定状态）：ALTER USER 用户名 ACCOUNT LOCK/UNLOCK

代码演示：

```
ALTER USER 123 ACCOUNT LOCK
```

▼ 二、DDL 数据定义语言 (DATA DEFINITION LANGUAGE)

包括：create\drop \alter 等命令

▼ （一）常用的数据类型

- 1、CHAR(length)：存储固定长度的字符串。
- 2、VARCHAR2(length)：存储可变长度的字符串
- 3、NUMBER(p, s)：既可以存储浮点数，也可以存储整数，p表示数字的最大位数（如果是小数包括整数部分和小数部分和小数点，p默认是38位），s是指小数位数。
- 4、DATE：存储日期和时间
- 5、TIMESTAMP：不但存储日期的年月日，时分秒，以及秒后6位，同时包含时区

▼ 6、各种数据类型的表达式

- 字符型用单引号加字符表示，例如，'ABC'
- 数字型直接用阿拉伯数字表示，例如，123
- 日期型不能直接表示，必须使用函数转换，例如 DATE'2020-01-29'，TO_DATE('2016-12-31 01:27: 7 ','YYYYMMDD')

▼ 7、各种数据类型的不同特点

- 字符型可以进行拼接，'yuan' || 'laoshi' --- 'yuanlaoshi'
- 数字型可以进行算术运算，+ - * /
- 日期型也能进行算术运算，但是只能日期减日期，或者日期加减数字

▼ （二）CREATE 创建表命令：CREATE TABLE 命令

- 方法一：直接建表 create table 表名 (列名 数据类型, 列名 数据类型,...)
- 方法二：根据结果集创建表（快速建表） create table 表名1 as select * from 表名2 --where 1=1 （当where 1=1 时，复制表结构和数据，当1=2或其他不为1的数，仅复制表结构,数据类型也会被复制，但主键等不会被复制）
- 注：数据库中的数据是以表的形式储存的，表是最基本的数据库对象，创建表时，oracle 会在指定的表空间中为表分配存储空间

▪ （三）DROP 删除命令（删除整个表）：DROP TABLE 表名

代码演示：

```
drop table emp1
```

- (四) TRUNCATE 清空表数据命令 (只清空表数据, 留下表结构) : TRUNCATE TABLE 表名

代码演示:

```
truncate table emp1
```

会清空表数据, 不会删除表结构

▼ (五) ALTER 修改命令

- 1、修改表名: ALTER TABLE 表名 RENAME TO 新表名

代码演示:

```
alter table emp2 rename to emp3
```

- 2、删除列: ALTER TABLE 表名 DROP COLUMN 列名

代码演示:

```
alter table emp2 drop column empno
```

- 3、添加列: ALTER TABLE 表名 ADD 列名 数据类型

代码演示:

```
alter table emp2 add empno number(2)
```

- ★ 4、修改列类型: ALTER TABLE 表名 MODIFY 列名 数据类型

代码演示:

```
alter table emp1 modify empno varchar2(10)
```

- 5、修改列名: ALTER TABLE 表名 RENAME COLUMN 旧列名 TO 新列名

代码演示:

```
alter table emp1 rename column deptno to deptno1
```

▼ (六) 创建约束

- 1、添加主键约束: ALTER TABLE 表名 ADD CONSTRAINT 约束名 PRIMARY KEY(列名1,列名2,.....)

代码演示:

```
ALTER TABLE EMP ADD CONSTRAINT PK_EMP_EMPNO PRIMARY KEY  
(ENMPNPO)
```

- 2、添加外键约束: ALTER TABLE 表名 ADD CONSTRAINT 约束名 FOREIGN KEY (列名1,列名2,...) REFERENCE 从表名 (列名1,列名2,.....)

代码演示:

```
alter table emp ad constraint FK_dept_deptno foreign key (deptno) reference  
dept(deptno)
```

- 3、添加CHECK约束: ALTER TABLE 表名 ADD CONSTRAINT 约束名 CHECK(条件)

代码演示:

```
ALTER TABLE EMP1 ADD CONSTRAIN CK_JOB CHECK(JOB = 'TEACHER' OR JOB  
= 'SALESMAN')
```

- 4、添加唯一约束: ALTER TABLE 表名 ADD CONSTRAINT 约束名 UNIQUE(列名)

```
ALTER TABLE EMP ADD CONSTRAINT UN_DEPT_DEPTNO UNIQUE(DEPTNO)
```

- ★ 5、添加非空约束：ALTER TABLE 表名 MODIFY 列名 NOT NULL

代码演示：

```
ALTER TABLE EMP MODIFY ENAME NOT NULL
```

- 6、删除约束：ALTER TABLE 表名 DROP CONSTRAINT 约束名

约束不能修改，只能删除后再重新创建

代码演示：

```
ALTER TABLE EMP DROP CONSTRAINT PK_EMP_EMPNO
```

▼ 7、注：

- 可以在创建表时制定，也可以在表创建后制定，可以对一个或多个字段进行约束
- 表的约束是ORACLE 数据库中应用在表数据上的一系列强制性规则
- 1、一张表只有一个主键，主键非空且唯一
- 2、外键只能依赖于另外一张表的主键
- 3、约束不能修改，只能删除重建

▼ 三、DML 数据操纵语言（DATA MANIPULATION LANGUAGE）

包括insert / select / update / delete等命令

▼ （一）INSERT 插入命令

- 1、一行一行的插入数据：INSERT INTO 表名 VALUES(值1, 值2,)

INSERT INTO EMP2

```
VALUES(1111, 'winds','student',1111,to_date(20191224,'YYYYMMDD'),1111,111,11)
```

- 2、部分字段插入数据：INSERT INTO 表名 (列名1, 列名2,...) VALUES (值1, 值2,)

代码演示：

```
INSERT INTO EMP1(EMPNO,SAL,DEPTNO) VALUES(1111, 3333.33,40)
```

- 3、一次性插入整个查询结果集：INSERT INTO 表名 (列名1, 列名2,)
SELECT 查询结果集，注：字段的数量、数据类型、约束必须一致，但字段名不做要求

代码演示：

```
INSERT INTO EMP1 SELECT * FROM EMP
```

- （二）UPDATE 更新命令：UPDATE 表名 SET 列名1 = 值1,列名2= 值2, ...WHERE 条件 --注：WHERE条件没有的话会全表更新

代码演示：

```
UPDATE EMP SET SAL = 999.99
```

```
UPDATE EMP SET SAL = 999.99 WHERE ENAME = 'Smith'
```

▼ （三）DELETE 删除命令

- 1、语法结构：DELETE FROM 表名 WHERE 条件 --注：WHERE条件没有的话则删除全表数据

代码演示：

```
DELETE FROM EMP WHERE EMPNO = 10;--删除empno为10的所有记录
```

```
DELETE FROM EMP --删除全表数据
```

▼ 2、DELETE 和 TRUNCATE 的区别

- TRUNCATE是DDL命令，删除数据不可恢复；
- DELETE 是DML命令，删除后的数据可通过日志文件恢复；
- 记录很多的时候，TRUNCATE 相对DELETE性能好，但要慎用

- （四）COMMIT 提交结果：新增、删除、修改等DML操作完成后，需要COMMIT（按F10）才能对数据库进行修改，只有COMMIT后数据才能更新到表中，否则其他用户查询不到当前用户的操作结果

▼ （五）SELECT FROM 简单查询命令

- （1）查询逻辑：1、from 表 2、where 筛选条件（可以是字段的关系运算、算术运算、关系运算和算术运算结合） 3、group by 按条件分组； 4、having 条件； 5、select 要显示的内容（可以是原字段、也可以是处理后的字段及字段相关、固定值，别名有空格的时候必须加双引号； 6、order by 排序字段，可以原字段、处理后的字段、select后字段等）

▼ （2）算术运算

- 1、ORACLE 中的数值类型和日期类型都可以进行算术运算，算术运算符有+、-、*、/ 四个
- 2、可以用在select后显示字段运算结果，也可以用在where 后面限定查询结果集

代码演示：

```
select sal, 12* sal from emp;
```

```
select sal, 12*sal as year_sal from emp where 12*sal >3000;
```

```
select sal, 12*sal as year_sal from emp  
order by year_sal;
```

- 3、ORACLE会做隐式转换，字符串也可以做运算，运算结果：字符串靠左，数值靠右

▼ （3）字符串的拼接 ||

- 1、在ORACLE中，字符串的连接用双竖线（||）表示，拼接函数为CONCAT(A,B)
- 2、可以用在select 后面显示字段运算结果，也可以用在where 后面 限定查询结果集

```
select empno,ename,empno||ename, '工资 ' ||sal from emp ;
```

```
select * from emp where ename||job = 'ALLENSALESMAN';
```

- 3、可以各种类型任意拼接：字符串与字段拼接、日期的拼接等，拼接数量不限
- 4、ORACLE中字符串可以加用单引号，也可以用双引号，在别名中存在空格时，必须用双引号，在表名、列名时用双引号

- (4) 关系运算：=、>、<、>=、<=、<>或!=（不等于），ORACLE中常出现在WHERE字句中

代码演示：

```
SELECT * FROM EMP WHERE SAL > 3000;
```

- (5) 逻辑运算：AND、OR、NOT，三个逻辑运算符优先级：NOT > AND > OR, ORACLE中常出现在WHERE字句中

代码演示：

```
SELECT * FROM EMP WHERE JOB = 'CLERK' AND SAL > 1000;
```

```
SELECT * FROM EMP WHERE JOB = 'CLERK' OR JOB = 'SALESMAN';
```

▼ (六) 高级查询

- (1) DISTINCT操作：根据SELECT后面显示的结果进行去重，和group by对比，group by是基于分组实现去重，性能比distinct好

代码演示：

```
SELECT DISTINCT DEPTNO FROM EMP;
```

▼ (2) NULL操作

- 1、定义：某条记录缺少数据值，就是空值（NULL值），空值不是0或空格，是指未赋值、未知或不可用的值
- 2、任何数据类型的列都可以包括NULL值，除非该列被定义为非空或主键，查询条件中空值：IS NULL，非空值用 IS NOT NULL

代码演示：

```
SELECT * FROM EMP WHERE COMM IS NULL;
```

▼ 3、空值特性

- 空值不能进行算术运算，空值和任何值进行算术运算，结果都为空
- 空值不能进行关系运算，空值和任何值进行关系运算，结果都不成立
- 空值不参与任何聚合运算
- 排序的时候，空值永远是最大的

- (3) IN, NOT IN 操作：在WHERE字句中使用IN操作符来查询其列值在指定的列表中的行

代码演示：

```
WHERE DEPTNO = 10 OR DEPTNO = 20;
```

```
WHERE DEPTNO IN (10, 20)
```

- (4) BETWEEN...AND...操作：在WHERE 字句中，可以使用BETWEEN操作符来查询列值包含在指定区间内的行，包含边界值，即包头包尾

代码演示：

```
SELECT *  
FROM EMP  
WHERE SAL >=800 AND SAL <= 1500;
```

```
SELECT *  
FROM EMP  
WHERE SAL BETWEEN 800 AND 1500;
```

- (5) LIKE 模糊查询：通配符'%' ---表示零个或多个任意字符、 '_' --表示一个任意字符

▼ (6) 集合运算

- 1、INTERSECT（交集）：返回两个查询共有的记录
- 2、UNION ALL（交集）：返回各个查询的所有记录，包括重复记录，不去重
- 3、UNION（并集）：返回各个查询的所有记录，不包括重复记录，去重
- 4、MINUS（补集）：返回第一个查询记录减去第二个查询记录之后剩余的记录
- 注：1、查询所返回的列数以及数据类型必须匹配，列名可以不同；2、只有UNION ALL 不会去重，其他三个都需要排序后去重，性能差

▼ SQL基础知识（二）

▼ 一、聚合分组

- (一) 定义：是一个多行函数，同时读取多行数据，返回一个值，对明细数据做统计
- (二) 常见的聚合函数：AVG(表达式)--平均值、SUM(表达式)--求和、MIN(表达式)--最小值、MAX(表达式)--最大值、COUNT(表达式)--数据统计

▼ (三) 聚合函数配合GROUP BY 进行使用，对一组数据进行操作，返回一行结果

- 1、语法结构：SELECT 字段 FROM 表名 WHERE 限定条件 1 GROUP BY 字段 HAVING 限定条件2
- 2、GROUP BY 添加HAVING 作为每个分组的限定条件，
- 3、分组后，SELECT 子句之后，只能出现分组的字段和统计函数，不能出现其他字段

▼ (四) HAVING

- 1、GROUP BY 后面还可以添加一个HAVING作为每个分组的限定条件。
- 2、语法结构：
SELECT 字段 FROM 表名 WHERE 限定条件1 GROUP BY 字段 HAVING 限定条件1(只能为聚合函数)
- 3、having后面只能跟聚合函数或分组字段相关的

- 注：1、空值不参与聚合函数；2、select后显示聚合函数不可和字段一起用，需配合group by ,但可以放固定值；3、能先select做筛选再做分组，性能会提高

▼ （五）group by 和distinct 的区别

- 1、distinct是在select 后对数据去重，group by 是分组，基于分组后进行去重
- 2、小表DISTINCT快，不用建索引。大表GROUP BY快。
一般来说小表就算建索引，也不会慢到哪去，但是如果是TB级大表，遍历简直就是灾难。
- 3、很多ORACLE项目都禁止使用DISTINCT语句，全部要求替换成GROUP BY。

▼ （六）WHERE 和 HAVING 的区别

- 1、WHERE 是在执行GROUP BY 操作之前进行过滤，从全部数据中筛选出部分数据，在WHERE 后不能使用聚合函数；
- 2、HAVING 是在GROUP BY 分组之后的再次过滤，可以在HAVING子句中使用聚合函数和分组字段进行条件限制。

- （七）ON 、WHERE、HAVING 的区别：ON 和WHERE都是针对明细数据，HAVING 针对分组数据，ON和HAVING后面可跟聚合函数

▪ （八）查询语法的执行顺序：

- 1、from子句组装来自不同数据源的数据；
- 2、where子句基于指定的条件对记录行进行筛选；
- 3、group by子句将数据划分为多个分组；
- 4、使用聚合函数进行计算；
- 5、使用having子句筛选分组；
- 6、计算所有的表达式；
- 7、select 的字段；
- 8、使用order by对结果集进行排序。

▼ 二、表连接

▼ （一）什么是表连接

- （1）表连接就是在笛卡尔（SELECT * FROM EMP CROSS JOIN DEPT ）的基础上，根据两个表的相同字段，选取有效信息
- （2）表连接的实质：1、表结构：合并字段；2、数据做笛卡尔积；3、关联条件，筛选出数据。

▼ （二）发散

- 1、一条记录对应多条就会发散；
- 2、统计粒度：数据库中数据的细化和综合程度，细化程度越高，粒度越小；
- 3、两个表关联，粒度大的会发散

▼ （三）表连接种类

- ▼ 1、内连接（JOIN）：笛卡尔积中，只显示有关联的数据，无关联不显示(只显示符合关联条件的数据)

- 语法一：SELECT */列名/表达式 AS 列别名 FROM 表名1 表别名 JOIN 表名2 表别名 ON 关联条件

代码演示：

```
SELECT *  
FROM EMP A  
JOIN DEPT B  
ON A.DEPTNO = B.DEPTNO
```

- 语法二(ORACLE写法)：SELECT */列名/表达式 AS 列别名 FROM 表名1 表别名, 表名2, 表别名 WHERE 关联条件

代码演示：

```
SELECT *  
FROM EMP A, DEPT B  
WHERE A.DEPTNO = B. DEPTNO;
```

▼ 2、左（右）连接（LEFT JOIN）：笛卡尔积中，左边表中的数据必须显示

- 左外关联语法一：SELECT */列名/表达式[AS 列别名] FROM 表名1 [表别名] LEFT JOIN 表名2 [表别名] ON 关联条件

代码演示：

```
SELECT *  
FROM EMP A  
LEFT JOIN DEPT B  
ON A.DEPTNO = B.DEPTNO;
```

- 语法二（ORACLE写法）：SELECT */列名/表达式[AS 列别名] FROM 表名1 [表别名], 表名2 [表别名],....WHERE 关联条件

代码演示：

```
SELECT *  
FROM EMP A, DEPT B  
WHERE A.DEPTNO (+) = B.DEPTNO--A表为从表，B表为主表，主表要全部显示
```

- 注：方法二中，关联条件字段加（+）的为从表，不加（+）的为主表，主表全部显示

▼ 3、外连接（FULL JOIN）：笛卡尔积中，把两张表中没有的数都显示出来

- 语法：SELECT */列名/表达式 [AS 列别名] FROM 表名1 [表别名] FULL JOIN 表名2 [表别名] ON 关联条件

代码演示：

```
SELECT *  
FROM EMP A  
FULL JOIN DEPT B  
ON A.DEPTNO = B.DEPTNO;
```

- 总结：1、表跟表关联，一定要有公有字段，否则关联的信息没有意义；2、连接查询要避免笛卡尔积，一定要有关联字段，可以先用WHERE 筛选数据后再关联，提高性能；3、表连接除了关联不同的表，也可以与自己关联；4、ON后面字段不仅能加关联条件，也可加单表的限制条件，可以是等值也可以是不等值

▼ 思考：在使用left join时，on和where当限制条件的区别是什么？

- 1、on条件是在生成临时表时使用的条件，它不管on中的条件是否为真，都会返回左边表中的记录。

- 2、where条件是在临时表生成好后，再对临时表进行过滤的条件。这时已经没有left join的含义（必须返回左边表的记录）了，条件不为真的就全部过滤掉。

▼ SQL基础知识（三）

▼ 一、子查询

- （一）定义：子查询在SELECT、UPDATE、DELETE语句内部可以出现SELECT语句，内部的SELECT语句结果可以作为外部语句中条件字句的一部分，也可以作为外部查询的临时表
- （二）子查询可以看成是一个结果集，所以可以用在from后面和where后面，子查询，可以当做表、值、单列集合，

▼ （三）子查询的种类

▼ 1、单行子查询

- （1）定义：不向外部返回结果，或只返回一行结果
- （2）在单行子查询中外部查询可以使用 =、>、<、>=、<=、<>等比较运算符
- （3）语法结构：SELECT * FROM EMP WHERE SAL = (SELECT SAL FROM EMP WHERE ENMAE = 'SCOTT') AND ENAME != 'SCOTT'

▼ 2、多行子查询

- （1）定义：向外部返回零行、一行或多行结果
- （2）语法结构（IN）：SELECT ENAME,SAL FROM EMP WHERE JOB IN (SELECT JOB FROM EMP WHERE DEPTNO = 30) AND DEPTNO != 30
- （3）语法结构（ANY）：SELECT * FROM EMP WHERE JOB = ANY(SELECT JOB FROM EMP WHERE DEPTNO = 30) AND DEPTNO != 30

▼ （四）ANY子查询，ANY()：表示括号内任何一个条件，只要有一个满足即可

- < ANY ()：比子查询结果中最大值还小
- > ANY：比子查询结果中最小的还大
- = ANY：等于括号中的任意一结果即可

▼ （五）ALL子查询：表示括号内全部条件，要全部满足才可以

- < ALL：比子查询结果中最小值都小
- > ALL：比子查询结果中最大值都大

▼ 二、常用函数

- （一）ORACLE SQL提供了用于执行特定操作的专用函数，大大增强了SQL语言功能
- （二）函数定义：函数可接受零个或多个输入参数，并返回一个输出结果，Oracle数据库中主要使用单行函数和多行（聚合）函数2个类型
- ▼ （三）单行函数：对每一个函数应用在表的记录中时，只能输入一行结果，返回一个结果
 - ▼ 1、字符函数：对字符串进行操作，接受字符参数

- (1) ASCII(X): 返回字符X 的ASCII码

代码演示:

```
SELECT ENAME,ASCII(ENAME) FROM EMP
```

- (2) 字符串拼接函数: CONCAT(X,Y), 字符串拼接, 也可用|| 进行拼接

```
SELECT ENAME, JOB ,ENAME||JOB ,CONCAT(CONCAT(ENAME,JOB),SAL)
FROM EMP
```

- (3) 读取字符位置函数: INSTR(X,STR[,START][,N]), 在X中查找STR, 可以指定从start开始, 也可以指定从第N次开始

代码演示:

```
INSTR('Hello world','l')--第一个l出现的位置
```

```
INSTR('Hello word','l',4)---从第四个数开始数, l出现的位置
```

```
INSTR('Hello word','l',4,2)---从第四个位置开始数, l第二次出现的位置
```

- (4) 字符截取函数: SUBSTR(X,A[,B]), 返回X的字串, 从A 处开始, 截取B个字符, 缺省B,默认到结尾,如果从结尾开始截取, A用负数, 但截取都是从左到右判断

代码演示:

```
SELECT
ENAME,
SUBSTR(ENAME,1,1), --从第1个截1个
SUBSTR(ENAME,1, 3),--从第1个截3个
SUBSTR(ENAME,-3, 2)--从倒数第3个截2个
FROM EMP
```

▼ (5) 其他的常见的字符函数

- LENGTH(X): 返回X的长度
- LOWER(X): X转换为小写
- UPPER(X):X转换为大写
- LTRIM(X[,trim_str]): 把X的左边截去trim_str字符串, 缺省截去空格
- RTRIM(X[,trim_str]): 把X的右边截去trim_str字符串, 缺省截去空格
- TRIM([trim_str FROM] X): 把X的两边截去trim_str字符串, 缺省截去空格
- REPLACE(X,old,new): 在X中查找old, 并替换为New

▼ 2、数字函数: 对数字进行计算, 返回一个数字

- (1) 数字函数接受数字参数, 参数可以是来自表中的一列, 也可以是一个数字表达式

▼ (2) 数字函数的精度

- ROUND(X[,Y]) 四舍五入函数: Y为正数时, 保留小数点后Y位数四舍五入, Y为负时, 保留小数点前Y位数四舍五入

- TRUNC(X[,Y]) :直接掉, 不四舍五入: Y为正时, 截掉小数点Y位数后的数据, Y为负时, 截掉小数点前Y位的数据

▼ (3) 其他常见数字函数

- ABS(X): 返回X的绝对值
- MOD(X,Y): 返回X/Y的余数
- POWER(X,Y): X的Y次幂

▼ 3、日期函数: 对日期和时间进行处理

- (1)ADD_MONTHS(d,n): 在某个日期D上, 加上指定的月数N, 返回计算后的新日期

- (2) LAST_DAY(D): 返回指定日期D当月的最后一天

代码演示:

```
SELECT HIREDATE,
       ADD_MONTHS(HIREDATE,2),
       LAST_DAY(HIREDATE)
FROM EMP;
```

▼ (3) 关于日期的精度函数

- ▼ (1)ROUND(D [,fmt]) : 返回一个以fmt为格式四舍五入日期值, 默然fmt为DDD, 即某月中的某一天

- ROUND(D, 'YEAR'): 舍入到某年的1月1日, 前半年舍去, 后半年作为下一年
- ROUND(D, 'MONTH'):舍入到某月的1日, 前半月舍去, 后半月作为下一月
- ROUND(D, 'DDD'): 舍入到日期D月中的某一天, 前半天舍去, 后半天作为下一天
- ROUND(D, 'DAY'): 返回最近的周的周日, 上半周舍去, 下半周作为下个周的周日

- ▼ (2) TRUNC(D [,FMT]) : 将D截取到fmt指定的形式, 如果fmt省略, 默然截取到最近的日期

- TRUNC(D, 'YEAR'): 返回到某年的1月1日
- TRUNC(D, 'MONTH'): 返回到某月的1日
- TRUNC(D, 'DDD') :返回月中的某一天, 即返回当天
- TRUNC(D, 'DAY'): 返回本周的周日

▪ (3) 代码演示

```
SELECT SYSDATE,  
       ROUND(SYSDATE,'YEAR'), ----舍入到某年的1月1日, 上半年舍去, 下半年作为下一年  
       ROUND(SYSDATE,'MONTH'), ----舍入到某月的1日, 上半月舍去, 下半月作为下一月  
       ROUND(SYSDATE,'DDD'), ----舍入到月中的某一天, 上半天舍去, 下半天作为下一天  
       ROUND(SYSDATE,'DAY') ----舍入到最近的周的周日, 上半周舍去, 下半周作为下周周日  
FROM DUAL  
  
UNION ALL  
SELECT SYSDATE,  
       TRUNC(SYSDATE,'YEAR'), ----返回某年的1月1日  
       TRUNC(SYSDATE,'MONTH'), ----返回某月的1日  
       TRUNC(SYSDATE,'DDD'), ----返回当天  
       TRUNC(SYSDATE,'DAY') ----返回本周的周日  
FROM DUAL;
```

▼ 4、转换函数：将一种数据类型转换为另一种数据类型

▼ (1) TO_CHAR(d/n [,fmt])：将指定的日期或数字转换为指定格式fmt的字符串

- ● 隐式转换：字符串类型靠左，数值类型靠右

▼ (2) TO_DATE(X [,fmt])：把字符串X转换为fmt格式的日期类型,

- ❶ 使用日期类型的日期，必须使用TO_DATE函数进行转换
- ❷ 输入的参数精度和转换的参数精度要一致

▪ ❸ 代码演示：

```
SELECT TO_date('20191226','YYYYMMDD') FROM DUAL;
```

▪ (3) TO_NUMBER(X [,fmt])：把字符串X转换为fmt格式的数值类型

- (4) 日期类型不能转换成TO_NUMBER,只能转换成TO_CHAR,而且显示的格式和精度都可以控制

▪ (四) 聚合函数：多行函数，可以对多行数据进行操作，并返回一个结果，比如SUM() 函数

▼ (五) 其他常用的函数

- 1、NVL(列，默认值)：如果列值为空null，则使用默认值表示，可以运用在去空值，因为空值不参运算，可以先将空值换成为0

- 2、NVL2(列, 返回值1, 返回值2): 如果列值非NULL, 返回值1, 如果列值为null, 返回值2, 可应用在去空值

----NVL() 函数的应用: 去空值

代码演示:

```
SELECT SAL,  
       COMM,  
       SAL+COMM,  
       SAL + NVL(COMM,0),  
       12*SAL + NVL(COMM,0),  
       NVL2(COMM,12*SAL + COMM,12*SAL)  
FROM EMP;
```

▼ 3、DECODE () 多值判断函数

- 多值判断函数, 不是所有数据库都可用多值判断, 如果列值与判断值相同, 则显示对应返回值输出, 如果没有满足条件, 则显示默认值
- 语法: DECODE(列值, 判断值1, 返回值1, 判断值2, 返回值2, ...,默认值)

▪ 代码演示

案例3: 列出EMP员工的姓名, 以及工作 (中文)

```
SELECT ENAME, JOB,  
       DECODE(JOB,  
              'CLERK','业务员',  
              'SALESMAN','销售员',  
              'MANAGER','经理',  
              'ANALYST','分析员',  
              'PRESIDENT','总裁',  
              '其他')  
FROM EMP;
```

▼ 4、CASE WHEN 多条件判断函数

- CASE WHEN 和DECODE相似, 但DECODE只用于多值判断, CASE WHEN是多条件判断
- 是一个字段, 内含有逻辑判断, 比DECODE灵活
- 所有数据库都支持用于实现多条件判断, 如果都不满足条件, 则返回默认值

▼ 语法

▪ 语法1

```
CASE  
WHEN 条件1 THEN 返回值1  
WHEN 条件2 THEN 返回值2  
.....
```

```
ELSE 默认值 END
```

当条件成立, 则返回对应的返回值, 没有条件成立则返回默认值

- 语法2

- CASE 参数

- WHEN 判断值1 THEN 返回值1

- WHEN 判断值2 THEN 返回值2

-

- ELSE 默认值 END

- 当参数的值为判断值1，则返回返回值1.....当参数的值匹配不到时，则返回默认值

- 代码演示

- 案例4：列出EMP员工的姓名，工资，以及工资评价

- SELECT ENAME,SAL,

- CASE

- WHEN SAL > 3000 THEN '工资很高'


- WHEN SAL > 1000 THEN '工资一般'

- ELSE '工资很低'

- END AS 工资评价

- FROM EMP;

- ▼ 5、EXISTS函数

- EXISTS（子查询）：用于判断子查询是否有数据返回，如果有则成立，否则不成立
 - EXISTS(查询结果集)：查询结果集有则成立，否则不成立
 - NOT EXISTS(查询结果集)：和EXISTS相反
 -  --子查询中SELECT后面可以加任意数字

- 注意:

- EXISTS、IN、关联必然可以相互转换。

- 同理NOT EXISTS、NOT IN、外关联+从表IS NULL也能相互转换

- EXISTS 关联性能最好

案例5: 查询EMP表和30号部门员工相同岗位的员工的员工信息, 30号部门员工除外;

--EXISTS语法

```
SELECT *
FROM EMP A
WHERE EXISTS(
    SELECT 1
    FROM EMP
    WHERE DEPTNO = 30
    AND A.JOB = JOB )
AND DEPTNO != 30;
```

---IN语法

```
SELECT *
FROM EMP
WHERE JOB IN (SELECT JOB FROM EMP WHERE DEPTNO = 30)
AND DEPTNO != 30;
```

---JOIN 语法

```
SELECT A.*
FROM EMP A
JOIN (SELECT JOB FROM EMP WHERE DEPTNO = 30) B
ON A.JOB = B.JOB
AND A.DEPTNO != 30;
```

- ● EXISTS去重, 性能比DISTINCT性能好

代码演示:

---DISTINCT

```
SELECT DISTINCT A.DEPTNO,A.DNAME
FROM DEPT A
JOIN EMP B
ON A.DEPTNO = B.DEPTNO;
```

---EXISTS去重

```
SELECT A.DEPTNO,A.DNAME
FROM DEPT A
WHERE EXISTS(SELECT 1 FROM EMP WHERE A.DEPTNO = DEPTNO) ;
```

```
SELECT JOB FROM EMP WHERE DEPTNO = 30
```


- 为什么EXISTS去重比DISTINCT性能好，其运行原理是怎样的？
用EXISTS，只要找到第一个符合条件的值，就返回了，而不管后面有多少条符合条件的重复记录。
而DISTINCT，是全扫描，必须查找全部符合条件的记录后，再返回唯一值。
- 总结：
用distinct需要每次都遍历匹配表进行比对，而使用exists只需要比对匹配表的一部分，在匹配表数据十分庞大时，这种性能差别就能更好的体现出来。

注意，EXIST去重只适用于一对多查询时的去重，如果只是对单表数据进行去重，就无法使用EXIST！

▼ 三、伪列

▼ （一）什么是伪列

- 在ORACLE的表使用过程中，实际表中还有一些附加列，称为伪列，伪列和表中的列一样，但在表中并不存储，
- 伪列只能查询，不能进行增删改操作

▼ （二）2个伪列

▼ 1、ROWID

- （1）定义：表中的每一行在数据文件中都有一个物理地址，ROWID伪列返回该行的物理地址
- ▼ （2）特性
 - a. 使用ROWID可以快速定位表中的某一行
 - b. ROWID值可以唯一标识表中的某一行
 - c. 由于ROWID返回的是该行的物理地址，因此使用ROWID可以显示行是如何存储的
 - d. 是字符串
 - e. 可以比较大小
 - f. 可以删除重复数据
- ▼ （3）应用
 - 删除重复数据，相同数据只保留一条
 - 语法结构：
DELETE FROM 表名 别名
WHERE ROWID NOT IN
(SELECT MIN(ROWID) FROM 表名 别名 GROUP BY 列名)

- 代码演示:
 ----使用伪列删除DEPT1 表中的重复数据
 DELETE FROM DEPT1
 WHERE ROWID NOT IN (SELECT MIN(ROWID) FROM DEPT1 GROUP BY
 DEPTNO,DNAME,LOC);

▼ 2、ROWNUM

- (1) 定义：在查询结果集中，ROWNUM为查询结果集中的每一行标识一个行号，第一行返回1，第二行返回2，以此类推

▼ (2) 特性

- a. 通过ROWNUM伪列可以限制查询结果集中返回的行数，一般结合子查询
- b. 每一个新的结果集生成新的ROWNUM
- c. 使用ROWNUM 做限定条件时，只能使用< 或<=，大于或大于等于不返回结果，可以将其结合子查询将伪列转换为一个字段进行使用
- d. 生成ROWNUM操作比排序要早，排序时已经连同ROWNUM一起排序了，因此常常需要结合子查询，对排序的结果重新做二次查询，产生新的ROWNUM才能作为查询的条件依据

▼ (3) 应用

- 通过ROWNUM伪列可以限制查询结果集中返回的行数
- 语法结构：一般结合子查询将伪列转换为一个字段进行使用，条件限制中只能使用< 或<=

▪ 代码演示

案例3：查询出表EMP中第5条到第10条之间的记录

代码演示：

```
SELECT B.*
```

```
FROM (
```

```
    SELECT ROWNUM R,A.* FROM EMP A ---使用ROWNUM生成伪列，将其转换成一个字段列
```

```
    ) B
```

```
WHERE R BETWEEN 5 AND 10;
```

内部查询中得到ROWNUM 并且用别名R记录，供外层条件使用。

内部查询的ROWNUM，与外出的ROWNUM列是平等的两列。

使用的R是内层产生的ROWNUM，在外层看来，内层查询的ROWNUM是正常的
 一列。

▼ 3、ROWID和ROWNUM区别

- 总结：ROWNUM与ROWID不同，ROWID是插入记录时生成，ROWNUM是查询数据时生成。ROWID标识的是行的物理地址。ROWNUM标识的是查询结果中的行的次序。

▼ SQL基础知识（四）

▼ 一、分析函数

- 1、定义：分析函数是ORACLE专门用于解决复杂统计报表需求的功能强大的函数，可以在数据中进行分组然后基于组计算统计值，并每组每行都可以返回一个统计值

▼ 2、语法结构

- a. 分析函数带有一个开窗函数 OVER(), 包含三个分析语句:分组 (PARTITION BY), 排序 (ORDER BY), 窗口 (ROWS)

▼ b. 分析函数语法:

FUNCTION_NAME(<参数>,...) OVER (<PARTITION BY 表达式,...> <ORDER BY 表达式 <ASC DESC>)

- PARTITION BY 是分组，partition by 后面可以按需求跟多个字段，来达到你想要的筛选目的
- ORDER BY 在求和时是累计

- c. MAX(),MIN(),SUM(),AVG(),COUNT() --加了ORDER BY 是累计求值

- 注：分析函数是一个整体，不可分割

▪ 代码演示

---按照部门分组求和，求部门总收入，每行返回对应部门总收入

--- partiton by deptno 是按部门分组，分别求各部门的总收入，并返回给各组的每一行

```
SELECT A.*,  
       SUM(SAL) OVER(PARTITION BY DEPTNO)  
FROM EMP A;
```

--- 按照员工编号 (order by empno) 排序，求工资的累计值

```
SELECT A.*,  
       SUM(SAL) OVER(ORDER BY EMPNO)  
FROM EMP A;
```


---按部门 (deptno) 分组，同时按员工编号 (empno) 排序取员工部门内累计收入和

```
SELECT A.*,  
       SUM(SAL) OVER(PARTITION BY DEPTNO ORDER BY EMPNO)  
FROM EMP A;
```

▼ 3、分析函数与聚合函数的区别

- a. 普通函数分组使用group by，分析函数分组使用PARTITION BY
- b. 普通函数每个分组返回一个统计值，分析函数每个分组的每一行返回一个统计值

▼ 4、RANK(),DENSE_RANK()与ROW_NUMBER()

- 为每条记录产生一个从1开始至N的自然数，
 N的值可能小于等于记录的总数。这3个函数的唯一区别在于当碰到相同数据时的排名策略。
- ①ROW_NUMBER:
ROW_NUMBER函数返回一个唯一的值，当碰到相同数据时，排名按照记录集中记录的顺序依次递增。

- ②DENSE_RANK:
DENSE_RANK函数返回一个唯一的值，当碰到相同数据时，此时所有相同数据的排名都是一样的。
- ③RANK:
RANK函数返回一个唯一的值，当碰到相同的数据时，此时所有相同数据的排名是一样的，
同时会在最后一条相同记录和下一条不同记录的排名之间空出排名。

- 注：此分析函数必须要加order by排序

- 代码演示

```
SELECT A.*,
      --- ROW_NUMBER() OVER(PARTITION BY DEPTNO ORDER BY SAL) ---按部门分组，各组按工资排序，给每一行生成序号，相同工资生成不同序号
      --- ROW_NUMBER() OVER(ORDER BY SAL) ---按工资排序，给每一行生成一个序号
      ---DENSE_RANK() OVER(PARTITION BY DEPTNO ORDER BY SAL) --按部分分组，各组按工资排序，给每一行生成一个序号，工资相同的序号相同
      RANK() OVER(PARTITION BY DEPTNO ORDER BY SAL) ---工资相同的序号相同，最后一条相同记录和下一条不同记录的排名之间空出排名
FROM EMP A;
```

- ROW_NUMBER()函数的应用：数据的去重，结合子查询、ROWID伪列去重

```
---常使用分析函数ROW_NUMBER()进行去重
DELETE FROM EMP1
WHERE ROWID NOT IN ( SELECT ROWID
                    FROM (
                        SELECT A.*,
                               ROW_NUMBER() OVER(PARTITION BY EMPNO ORDER BY EMPNO,
                                                    ROWID) RN
                        from EMP1 A)
                    WHERE RN = 1)
```

▼ 5、LAG()和LEAD()：求之前或之后的第N行

- (1) 语法使用说明LAG(ARG1,ARG2,ARG3):
第一个参数是列名，
第二个参数是偏移的offset，
第三个参数是超出记录窗口时的默认值。
- (2) LAG和LEAD函数可以在一次查询中取出同一字段的前N行的数据和后N行的值。这种操作可以使用对相同表的表连接来实现，不过使用LAG和LEAD有更高的效率

- (3) LAG () 向下偏移, LEAD() 向上偏移

代码演示:

--LAG()向下偏移

```
SELECT ENAME,
       JOB,
       HIREDATE,
       LAG(HIREDATE,2) OVER(ORDER BY HIREDATE ASC)
FROM EMP;
```

--LEAD() 向上偏移

```
SELECT ENAME,
       JOB,
       HIREDATE,
       LEAD(HIREDATE, 2) OVER(ORDER BY HIREDATE ASC)
FROM EMP;
```

▼ (4) LAG() 和 LEAD() 的应用

- 求环比: 现阶段 - 同一周期上一阶段 / 同一周期上一阶段
- 求同比: 现阶段 - 上一周期相同阶段 / 上一周期相同阶段
- 代码演示:

--求环比和同比

```
SELECT Y,Q,AMT,
       (AMT - LAG(AMT,1) OVER(PARTITION BY Q ORDER BY Y))/ LAG(AMT,1)
       OVER(PARTITION BY Q ORDER BY Y) AS 同比,
       (AMT - LAG(AMT,1) OVER(ORDER BY Y,Q) ) / LAG(AMT,1) OVER(ORDER BY
       Y,Q) AS 环比
FROM AAA
ORDER BY Y,Q;
```

▼ 二、行列转换

- (一) 在用户制作数据报表时, 经常会使用到报表数据的行列转换操作
- (二) 行列转换主要思路: 要把基表做出来了, 基表包含大类、小类、统计字段, 在通过基表做条件限定或分组统计

▼ (三) 列转行方法

- 方法一: 使用分析函数 LAG() 和 LEAD(), 结合子查询, 先做数据偏移做出基础表, 再通过子查询, 限定对应条件, 筛选出对应的数据

代码演示:

```
SELECT A.Y,A.Q1,A.Q2,A.Q3,A.Q4
FROM (
  SELECT Y,
         Q,
         AMT AS Q1,
         LEAD(AMT,1) OVER(PARTITION BY Y ORDER BY Q) AS Q2,
         LEAD(AMT,2) OVER(PARTITION BY Y ORDER BY Q) AS Q3,
         LEAD(AMT,3) OVER(PARTITION BY Y ORDER BY Q) AS Q4
  FROM AAA ) A ----先做出基础表
WHERE Q = 1;
```

- 方法二：使用SUM + CASE WHEN 或 DECODE，结合多值判断或条件判断，不符合条件的返回NULL,做出基础表，因NULL不参与算数运算，做分组做求和统计

代码演示：

```
SELECT Y,
       SUM(CASE WHEN Q =1 THEN AMT ELSE 0 END) Q1,
       SUM(CASE WHEN Q =2 THEN AMT ELSE 0 END) Q2,
       SUM(CASE WHEN Q =3 THEN AMT ELSE 0 END) Q3,
       SUM(CASE WHEN Q =4 THEN AMT ELSE 0 END) Q4
FROM AAA
GROUP BY Y;
```

---也可以用DECODE

```
SELECT Y,
       SUM(DECODE(Q,1,AMT,NULL)) Q1,
       SUM(DECODE(Q,2,AMT,NULL)) Q2,
       SUM(DECODE(Q,3,AMT,NULL)) Q3,
       SUM(DECODE(Q,4,AMT,NULL)) Q4
FROM AAA
GROUP BY Y;
```

- 方法三：JOIN表关联，结合表关联和子查询，以大类字段作为关联条件，关联出想要的数据库

代码演示：

```
SELECT A.Y,A.AMT Q1,B.AMT Q2,C.AMT Q3,D.AMT Q4
FROM AAA A
JOIN (SELECT * FROM AAA WHERE Q =2) B
  ON A.Y = B.Y
JOIN (SELECT * FROM AAA WHERE Q =3) C
  ON A.Y = C.Y
JOIN (SELECT * FROM AAA WHERE Q =4) D
  ON A.Y = D.Y
AND A.Q = 1
```

- ▼ 方法四：使用PIVOT（）行列转换公式，PIVOT直接在FROM 后使用

- ● 首先要确定统计大类、小类、统计字段、统计类型
- PIVOT语法结构：直接在FROM后使SELECT *
FROM 表名
● PIVOT(统计类型(统计字段) FRO 小类 IN (小类参数1，小类参数2，....)) -
--注：IN后面是最后结果集的字

- 代码演示：

```
--以AMT、Q外的其它全部字段为大类进行分组
大类： Y
小类： Q
统计字段： AMT
统计类型： SUM(AMT)
```

```
SELECT *
FROM AAA
PIVOT(SUM(AMT) FOR Q IN (1 Q1 ,2 Q2,3 Q3,4 Q4))
```

▼ (四) 行转列

- 方法一：UNION ALL 方法

```
--UNION ALL代码演示  
SELECT Y,1,Q1 FROM AAB  
UNION ALL  
SELECT Y,2,Q2 FROM AAB  
UNION ALL  
SELECT Y,3,Q3 FROM AAB  
UNION ALL  
SELECT Y,4,Q4 FROM AAB
```

- 方法二：UNPIVOT

```
--UNPIVOT( )代码演示  
SELECT Y,Q,AMT  
FROM AAB  
UNPIVOT ( AMT FOR Q IN (Q1,Q2,Q3,Q4) );
```

- 注：主要方法是case when ,oracle 自带pivot() 行列转换函数