# 体系结构

数据库可以比作类，实例就是数据库的实例化对象

oracle数据库 ---> 数据库实例 ---> 表空间(逻辑单位)(用户) ---> 数据文件(物理单位)

地球 ---> 国家 ---> 省份(逻辑单位)(公民) ---> 山川河流(物理单位)

数据库:database

Oracke 数据库是数据的物理存储。这就包括(数据文件ORA或者DBF\控制文件、联机日志、参数文件)。

其实Oracle数据库的概念和其它数据库不一样，这里的数据库是一个操作系统只有一个库。

Qracle就只有一个大数据库

实例

oracle只有一个数据库，但是可以新建很多个实例(mysql是数据库)，但是一般情况下，数据库=实例。默认实例：oracle，需要解锁学习的实例：scott\hr

一个oracle实例(Oracle instance)有一系列的后台进程（Backguound Processes）和内存结构（Memory Structures）组成

Java连接数据库就是连接实例：如创建了一个实例wwl；下面是连接url：

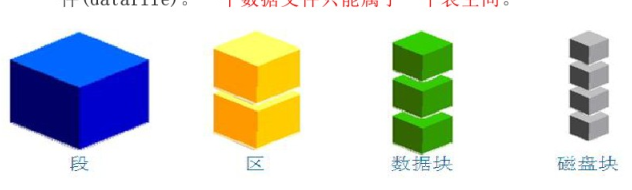
jdbc:oracle:thin:@localhost:1521:wwl

数据库文件(dbf)

数据文件是数据库的物理存储单位。数据库的数据是存储在表空间中的，真正是在某一个或者多个数据文件中。而一个表空间可以由一个或多个数据文件组成，一个数据文件只能属于一个表空间。一旦数据文件被加入到某个表空间后，就不能删除这个文件，如果要删除某个数据文件，只能删除其所属于的表空间才行

表空间

表空间是oracle对物理数据库上相关数据文件(ORA或者DBF文件)的逻辑映射。一个数据库在逻辑上呗划分成一到若干个表空间，每个表空间包含了在逻辑上相关联的一组结构。每个数据库至少有一个表空间(称之为system表空间)每个表空间由同一磁盘上的一个或多个文件组成。这些文件叫数据文件（datafile）。一个数据文件只能属于一个表空间



用户

用户是在实例下建立的。不同实例中可以建相同名字的用户

oracle与MySQL区别

新建一个项目区别:

MYSQL : 创建一个数据库,创建相应的表

Oracle: 创建一个表空间,创建用户,用户去创建表

Oracle是多用户的, MYSQL是多数据库的

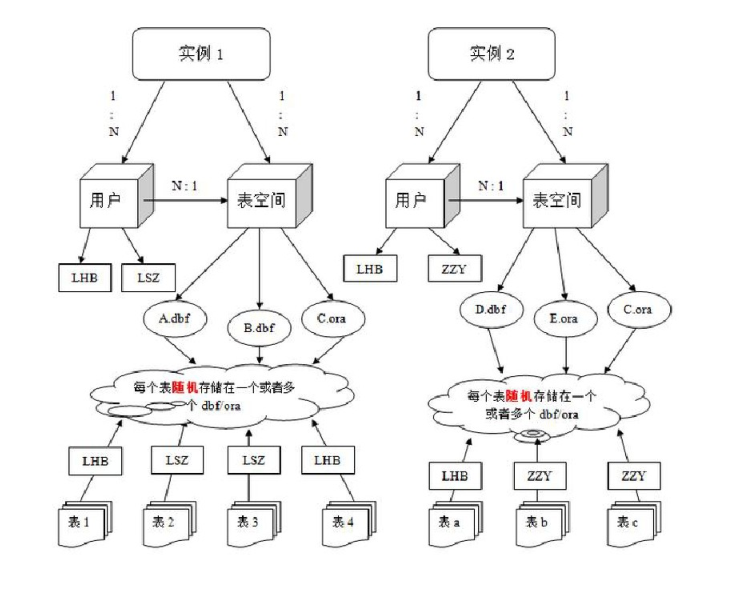
1. 遵循SQL标准

2. 不同厂商,不同的数据库产品,但是有自己的方言

3. 使用自己的方言,也能够完成相同的功能

4. Oracle安全级别要高,MYSQL开源免费

创建过程图



# 数据库操作

DDL : 数据定义语言 create alter drop truncate

DML : 数据操纵语言 insert update delete

DCL : 数据控制语言 安全 授权 grant revoke

DQL : 数据查询语言 select from子句 where子句

## 基本查询

编写顺序和执行顺序

Sql的编写顺序

Select……from ….where ….group by ……having….order by

Sql的执行顺序

From .. where ..group by ….having……select…order by

where和having区别

where 后面不能接聚合(多行)函数，可以接单行函数

having 是在group by之后执行，可以接聚合(多行)函数

查询语句的结构:

select [列名] [\*] from 表名 [where 条件] [group by 分组条件] [having 过滤] [order by 排序]

### 伪表(dual)

dual : oracle中的虚表 ,伪表, 主要是用来补齐语法结构

select 1+1; --在Oracle等于报错 ,在MYSQL中输出结果是2

select 1+1 from dual;

### 去除重复(distinct)

单列（字段）

如：scott表中的emp(员工表)部门编号字段(deptno)是有重复的

select deptno from emp; //查询到有重复的

select distinct deptno from emp //使用关键字distinct去除重复的

多列

多列去除重复: 每一列都一样才能够算作是重复

如:还是emp表：现在查询两个字段： job,deptno

select distinct deptno from emp：//会显示有：这两个字段都相同的

select distinct job,deptno from emp; //排除了两个字段都一致的

### 四则运算

select 1+1 from dual;

--查询员工年薪 = 月薪\* 12

select sal\*12 from emp;

--查询员工年薪+奖金：这里comm会有null值

注意: null值 , 代表不确定的 不可预知的内容 , 不可以做四则运算

select sal\*12 + comm from emp; //会出现很多空值行

使用--nvl 函数 : 如果参数1为null 就返回参数2；这样就能解决问题

select sal\*12 + nvl(comm,0) from emp;

### 字符串拼接(||)

oracle数据库拼接是：||(oracle特有的) 而不是+

在Oracle 中 ,双引号主要是别名的时候使用, 单引号是使用的值, 是字符

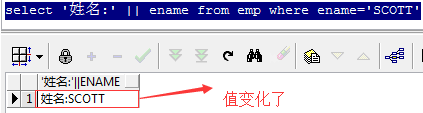
查询员工姓名 : 姓名为:SCOTT

select ename from emp where ename='SCOTT'

使用oracle特有的拼接字符，查询出的值带”姓名：”

select “姓名：” ename from emp where ename='SCOTT'

查询结果值：



注意：Mysql是没有的，使用函数：concat(str1,str2) 函数效果和上面一样, 在mysql和Oracle中都有

### 条件查询

条件查询 : [where后面的写法]

关系运算符: > >= = < <= != <>

三个逻辑运算符: and、or、 not

其它运算符:

like 模糊查询

in(set) 在某个集合内

between..and.. 在某个区间内

is null 判断为空

is not null 判断不为空

例子

--查询每月能得到奖金的员工信息

select \* from emp where comm is not null;

--查询工资在1500--3000之间的员工信息

select \* from emp where sal between 1500 and 3000; 第一种方式：between是英语当中的一个介词，意思为在......之间，常与and连用,如between A and B

select \* from emp where sal >= 1500 and sal <= 3000; 第二种方式

--查询名字在某个范围的员工信息 ('JONES','SCOTT','FORD') in

select \* from emp where ename in ('JONES','SCOTT','FORD');

### 模糊查询

--查询员工姓名第三个字符是O的员工信息

select \* from emp where ename like '\_\_O%';

--查询员工姓名中,包含%的员工信息

%需要转义：需要关键字配合：escape：告诉oracle这是转义字符

select \* from emp where ename like '%\%%' escape '\'; //告诉oracle转义字符是\

select \* from emp where ename like '%#%%' escape '#'; //告诉oracle转义字符是#

### 排序

排序 : order by

1默认升序: asc ascend降序: desc descend

2排序注意null问题 : nulls first | last

3同时排列多列, 用逗号隔开

重要例子：有null，和多个排序

查询员工信息,按照奖金由高到低排序：因此字段有null，需要用到关键字nulls

select \* from emp order by comm desc nulls last; //如果有null，默认是把null排在前面

--查询部门编号和按照工资 按照部门升序排序, 部门中的工资降序排序

select deptno, sal from emp order by deptno asc, sal desc; //asc可以不写

### rownum(伪列)

常用与：oracle的分页查询操作

rownum : 伪列, 系统自动生成的一列, 用来表示行号

rownum是Oracle中特有的用来表示行号的, 默认值/起始值是 1 ,在每查询出结果之后,再添加1

rownum最好不要做大于号判断(但是可以做大于0),可以做小于号判断

select rownum,e1.\* from emp e1 where rownum > 2 --没有任何记录，因为语句执行顺序问题：rownum关键字是先查询完了才生成一个编号

select rownum,e1.\* from emp e1 where rownum > =1 –这才有结果

select rownum,e1.\* from emp e1 where rownum > 0 --也有结果

select rownum,e1.\* from emp e1 where rownum < 8000 –有结果；查询7999之前的显示

--只查询员工表中工资最高的前三名信息，其它不需要

select rownum,t.\* from (select \* from emp order by sal desc)t where rownum<4; //这里要注意：要先排完顺序，在主查询显示rownum，不然是乱的

--分页查询emp表：下面是查询包括6到包括9的数据

第一种方式：自己想出来的(仿照网上)

select \* from (select rownum rowno,t.\* from emp t where rownum<=9) e where rowno>=6 //先查询出最终要到几的数据，然后主查询进行判断rownum的数字

第二种方式：老师方式

select \* from (select rownum rowno, emp.\* from emp) t where t.rowno between 6 and 9; //使用between关键字方式

### rowed

rowid : 伪列 每行记录所存放的真实物理地址

--去除表中重复记录 ：老师说这是华为面试的

delete from p p1 where rowid > (select min(rowid) from p p2 where p1.name = p2.name); //先查询出重复的最小的物理地址，删除剩余重复的物理地址

## 多表查询

### 内连接

隐式内联接:

等值内联接: where e1.deptno = d1.deptno;

不等值内联接: where e1.deptno <> d1.deptno; 没有意义

自联接: 自己连接自己：select \* from emp1,emp2 where

Scott账号中的emp表：--查询员工编号,员工姓名,此员工经理的编号, 此员工经理的姓名(都在一张表中记录)

select e1.empno,e1.ename,e1.mgr,m1.ename from emp e1, emp m1 where e1.mgr= m1.empno;

显示内联接:

select \* from 表1 inner join 表2 on 连接条件 // inner 关键字可以省略

--查询员工姓名和员工部门所处的位置

select e1.ename,d1.loc from emp e1,dept d1 where e1.deptno = d1.deptno; // 隐式

select \* from emp e1 inner join dept d1 on e1.deptno = d1.deptno; //显式

两个区别

当只是两张表时，显示和隐式都可以使用，而当需要多张表进行内链接时，就必须使用显示内连接的格式了

显式连接可以减少字段的扫描，有更快的执行速度。这种速度优势在3张或更多表连接时比较明显

显式条件需要用到是：on 不是where

### 外连接

外连接: (标准,通用写法)

左外连接: left outer join 左表中所有的记录,如果右表没有对应记录,就显示空

右外连接: right outer join 右表中的所有记录,如果左表没有对应记录,就显示空

outer 关键字可以省略

Oracle中的独有写法在内连接基础上加 (+) 实际上是如果没有对应的记录就加上空值：相当于就是左外连接

select \* from emp e1,dept d1 where e1.deptno = d1.deptno(+); //这是一个普通的内连接查询，但是加上(+)就变成右外连接了

### 子查询

子查询: 查询语句中嵌套查询语句; 用来解决复杂的查询语句

单行查询结果用在where条件判断语句中

查询出最高工资的人(很多实现方式，这只是告诉你子查询使用方式)

select \* from emp where sal=(select max(sal) from emp) ;

--查询出比雇员7654的工资高,同时和7788从事相同工作的员工信息

如果不使用子查询这个需要3步完成

--1.先查询出雇员7654的工资 1250

select sal from emp where empno = 7654;

--2.然后查询出7788从事的工作 ANALYST

select job from emp where empno = 7788;

--3.最后两个条件合并得出结果

select \* from emp where sal > 1250 and job = 'ANALYST';

使用嵌套子查询，一条语句查询出来：

select \* from emp where sal > (select sal from emp where empno = 7654) and job = (select job from emp where empno = 7788);

有多行查询结果用在from后使用（当作一张表）

--查询每个部门最低工资的员工信息

1、首先查询每个部门最低工资员工信息：涉及到根据部门进行分组

select deptno,min(sal) m from emp group by deptno

2、进行整合：把第一次查询到当作一张表放入from下：进行条件查询

select \* from emp t,(select deptno,min(sal) m from emp group by deptno) t2 where t.sal=t2.m and t.deptno=t2.deptno

子查询注意事项

1. 空值问题通常情况下, 数据库中不要出现null 最好的做法加上Not null( null值并不代表不占空间, char(100) null 100个字符)

--查询不是领导的信息

select \* from emp where empno not in (select mgr from emp); --错误：查询出来的空，因为此字段有null，

select \* from emp where empno not in (select mgr from emp where mgr is not null); --正确 子查询使用is not null排除了null值

### exists(查询语句)

exists(查询语句) : 存在的意思,判断一张表里面的记录是否存在与另外一张表中

1. 当作布尔值来处理(用的比较少，一般where a=b就可以解决):当查询语句有结果的时候, 就是返回true否则返回的是false

--使用exists判断子查询是否有值：有就是true执行总查询

select \* from emp where exists(select \* from emp where deptno = 1234567);--查看有没有员工为：1234567 没有返回false就不进行主查询了

select \* from emp where 3=4; //上面语句相当于这条，所以恨少用布尔值来处理

1. 数据量比较大的时候是非常高效的

--查询有员工的部门的信息

select \* from dept d1 where exists(select \* from emp e1 where e1.deptno = d1.deptno ); --查询到数据：拿dept表中deptno字段去emp中进行循环判断

## 集合运算

集合运算:

注意事项

1.每个表查询结果列的类型要一致

2.每张表查询查询结果列要按照顺序匹配

3.每张表查询结果列的数量要一致,如果不足,用空值填充

并集: 将两个查询结果进行合并,关键字：unio union all

union : 去除重复的,并且排序

union all : 不会去除重复的

一般用在两张不同表以上：进行合并信息

查询工资大于1500和部门为20的所有员工信息：

select \* from emp where sal > 1500

union

select \* from emp where deptno = 20

交集:查询A表和B表中有关联的信息；关键字intersect

--工资大于1500,并且是20号部门下的员工

select \* from emp where sal > 1500; //查询到所有员工1500元以上的工资

select \* from emp where deptno = 20; //查询到全部20部门编号的员工

--进行交集：就获得编号为20号部门，并且大于1500元的员工信息

select \* from emp where sal > 1500

intersect

select \* from emp where deptno = 20;

差集运算:两个结果相减；如：A表和B表：获取A表信息所有信息(排除跟B表的重复信息) 关键字minus

--1981年入职员工(不包括总裁和经理)

--1981年入职员工

select \* from emp where to\_char(hiredate,'yyyy')='1981'; //查询的结果这里包括总裁和经理的信息

--查询总裁和经理信息

select \* from emp where job = 'PRESIDENT' or job = 'MANAGER';

--进行差集关键字minus，就获得去除总裁和经理后1981年入职员工信息

select \* from emp where to\_char(hiredate,'yyyy')='1981'

minus

select \* from emp where job = 'PRESIDENT' or job = 'MANAGER';

# 表达式

## 条件表达式(decode)

条件判断表达式：跟if else if else 效果一样

通用表达式：case ‘字段’ when 值1 then 值 when 值2 then 值 else 值 end “双引号取别名”

Scott账号：select case ename when 'SMITH' then '孙悟空' when 'ALLEN' then '人渣' else '傻帽' end "昵称" from emp

Oracle独有表达式：decode(字段,if1,值1,if2,值2,其它值)

select decode(ename,'SMITH','孙悟空','ALLEN','人渣','傻帽') from emp

## 分组表达式(group by)

select 分组的条件(这里只能写分组条件，也可以不写),分组之后的操作 from 表名 group by 分组的条件 having 条件过滤

注意事项：

1. Select与from之间：只能写1、group by 的条件字段， 2、分组后的操作：如统计count(1)、求平均值等avg(这里可以别的字段)

//统计emp表中的部门各有多少人

select deptno,count(1) from emp group by deptno //这样是可以的，因为deptno就是分组的条件，

----错误的示范：添加了一个字段ename

select ename, deptno,count(1) from emp group by deptno //希望显示名字，这样是不正确的，

1. Group by 不能在having过滤中使用别名来操作：因为执行语句顺序：having是在select之前运行的 看[语句执行顺序](#_基本查询)

--分组统计所有部门的平均工资

select deptno,avg(sal) from emp group by deptno

--分组统计所有部门的平均工资，找出平均工资大于2000的部门

select deptno,avg(sal) from emp group by deptno having avg(sal)>2000

----错误的示范：表达式中使用别名 tt

select deptno,avg(sal) tt from emp group by deptno having tt>2000 // 这是错误的因为having是在select之前运行的，在这之前无法得到别名值

# 函数

函数: 必须要有返回值

## 单行函数

单行函数: 对某一行中的某个值进行处理

### 1、数值函数

--取大于等于数值value的最小整数：ceil(value)

select ceil(45.926) from dual; --46

--取小于等于数值value的最大整数：floor(value)

select floor(45.926) from dual; --45

--四舍五入：round(value,要取小数点第几位)

select round(45.926,2) from dual; --45.93

select round(45.926,1) from dual; -- 45.9

select round(45.926,0) from dual; --46

select round(45.926,-1) from dual; --50

select round(45.926,-2) from dual; --0 为啥0，-2意思就是看十位数：十位数是4所以四舍五入进位0

select round(65.926,-2) from dual; --100：为啥100，-2意思就是看十位数：十位数是6所以四舍五入进位100

--截断：trunc(value, 要取小数点第几位)：不会进行四舍五入

select trunc(45.926,2) from dual; --45.92

select trunc(45.926,1) from dual; -- 45.9

select trunc(45.926,0) from dual; --45

select trunc(45.926,-1) from dual; --40

select trunc(45.926,-2) from dual; --0

select trunc(65.926,-2) from dual; --0

--求余

select mod(9,3) from dual; --0

select mod(9,4) from dual; --1

### 2、字符函数

-- substr(str1,起始索引,长度) 截取字符

--注意: 起始索引不管写 0 还是 1 都是从第一个字符开始截取

select substr('abcdefg',0,3) from dual; --abc

select substr('abcdefg',1,3) from dual; --abc

select substr('abcdefg',2,3) from dual; --bcd

--获取字符串长度 24 28

select length('abcdefg') from dual;

--去除字符左右两边的空格

select trim(' hello ') from dual;

--替换字符串

Select replace('hello','l','a') from dual;

### 3、日期函数

--查询今天的日期

select sysdate from dual;

--查询3个月后的今天的日期

select add\_months(sysdate,3) from dual;

--查询3天后的日期

select sysdate + 3 from dual;

--查询员工入职的天数

select sysdate - hiredate from emp;

select ceil(sysdate - hiredate) from emp;

--查询员工入职的周数

select (sysdate - hiredate)/7 from emp;

--查询员工入职的月数

select months\_between(sysdate,hiredate) from emp;

--查询员工入职的年份

select months\_between(sysdate,hiredate)/12 from emp;

### 4、转换函数

--字符转数值 to\_number(str) 鸡肋：看下面列子就知道是鸡肋了

select 100+'10' from dual; --110 默认已经帮我们转换

select 100 + to\_number('10') from dual; --110

--数值转字符 to\_char(sal,'$9,999.99')

select to\_char(sal,'$9,999.99') from emp;

select to\_char(sal,'L9,999.99') from emp;//L是代表本地符号

--日期转字符 to\_char()

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') from dual; --只想要年 --2017

select to\_char(sysdate,'d') from dual; --2 参数一个d 代表一个星期中第几天

select to\_char(sysdate,'dd') from dual; --10 参数两个d 代表一个月中的第几天

select to\_char(sysdate,'ddd') from dual; --100参数三个d 代表一年中的第几天

select to\_char(sysdate,'day') from dual; --星期一；当天日期

select to\_char(sysdate,'dy') from dual; --当前日期 星期的简写

--字符转日期

select to\_date('2017-04-10','yyyy-mm-dd') from dual;

--查询1981年 -- 1985年入职的员工信息

select \* from emp where hiredate between to\_date('1981','yyyy') and to\_date('1985','yyyy');

### 5、通用函数(nvl)

nvl(参数1,参数2) 如果参数1 = null 就返回参数2

nvl2(参数1,参数2,参数3) 如果参数1 = null ,就返回参数3, 否则返回参数2

select nvl2(null,5,6) from dual; --6;

select nvl2(1,5,6) from dual; --5;

nullif(参数1,参数2) 如果参数1 = 参数2 那么就返回 null , 否则返回参数1

select nullif(5,6) from dual; --5

select nullif(6,6) from dual; --null

coalesce: 返回第一个不为null的值

select coalesce(null,null,3,5,6) from dual; --3

## 多行函数

多行函数: 对某一列的所有行进行处理

1、max()

2、min

3、count

4、sum

5、avg

例子

--统计员工工资总和

select sum(sal) from emp;

--统计员工奖金总和 2200

select sum(comm) from emp;

--统计员工人数 14

select count(1) from emp;

--统计员工的平均奖金 550 错误 2200/14 =

select avg(comm) from emp; // comm字段有空值

--统计员工的平均奖金 157.

select sum(comm)/count(1) from emp;

select ceil(sum(comm)/count(1)) from emp;