Tarena Technology
达内科技

Tarena Teaching System

Oracle 数据库基础

学员用书

Java企业应用及互联网
高级工程师培训课程

达内集团教学研发部 编著

目　录

Unit01 1
1. 数据库原理 3
1.1. 数据库简介 3
1.1.1. 文件存储 3
1.1.2. DB 和 DBMS 3
1.1.3. 关系数据库简介 3
1.1.4. 表的概念 4
1.2. 主流关系型数据库 4
1.2.1. Oracle 数据库概述 4
1.2.2. DB2 数据库概述 4
1.2.3. Sybase 数据库概述 5
1.2.4. SQL Server 数据库概述 5
1.2.5. MySQL 数据库概述 5
1.3. SQL 概述 6
1.3.1. 结构化查询语言 6
1.3.2. 数据定义语言（DDL） 6
1.3.3. 数据操作语言（DML） 6
1.3.4. 事务控制语言（TCL） 7
1.3.5. 数据查询语言（DQL） 7
1.3.6. 数据控制语言（DCL） 7
1.4. Oracle 数据库安装和访问 8
1.4.1. Oracle 数据库的安装（Windows） 8
1.4.2. Oracle 数据库的安装（Linux） 8
1.4.3. 远程登录数据库服务器 9
1.4.4. 使用 SQL*PLUS 9
1.4.5. 使用 Oracle SQL Developer 9
2. SQL（DDL、DML） 10
2.1. Oracle 数据类型 10
2.1.1. NUMBER 10
2.1.2. CHAR 10
2.1.3. VARCHAR2 11
2.1.4. DATE 11
2.2. 创建表 11
2.2.1. CREATE 语句 11
2.2.2. DESC 语句 12
2.2.3. DEFAULT 语句 12

达内IT培训集团

2.2.4. NOT NULL 12
2.3. 修改表 13
2.3.1. 修改表名 13
2.3.2. 增加列 13
2.3.3. 删除列 14
2.3.4. 修改列 14
2.4. DML 语句 14
2.4.1. INSERT 语句 14
2.4.2. UPDATE 语句 15
2.4.3. DELETE 语句 15
经典案例 17
1. Windows 下 Oracle 数据库安装 17
2. Linux 下 Oracle 数据库安装 20
3. 使用 telnet 登录数据库服务器 33
4. 使用 SQL* PLUS 访问数据库 33
5. 安装并使用 Oracle SQL Developer 访问 Oracle 34
6. NUMBER 数据类型实例 37
7. 字符数据类型实例 38
8. 日期数据类型实例 38
9. 创建员工表 39
10. 修改员工表 40
11. 插入员工数据 42
12. 更改员工数据 43
13. 删除员工数据 44
课后作业 45
Unit02 49
1. Oracle 字符串操作 51
1.1. 字符串类型 51
1.1.1. CHAR 和 VARCHAR2 类型 51
1.1.2. CHAR 和 VARCHAR2 的存储编码 51
1.1.3. CHAR 和 VARCHAR2 的最大长度 51
1.1.4. LONG 和 CLOB 类型 52
1.2. 字符串函数 52
1.2.1. CONCAT 和 “||” 52
1.2.2. LENGTH 52
1.2.3. UPPER、LOWER 和 INITCAP 53
1.2.4. TRIM、LTRIM、RTRIM 53
1.2.5. LPAD、RPAD 53
1.2.6. SUBSTR 54

1.2.7. INSTR 54
2. Oracle 数值操作 54
2.1. 数值类型 54
2.1.1. NUMBER（P）表示整数 54
2.1.2. NUMBER（P，S）表示浮点数 55
2.2. 数值函数 55
2.2.1. ROUND 55
2.2.2. TRUNC 56
2.2.3. MOD 56
2.2.4. CEIL 和 FLOOR 56
3. Oracle 日期操作 57
3.1. 日期类型 57
3.1.1. DATE 57
3.1.2. TIMESTAMP 57
3.2. 日期关键字 57
3.2.1. SYSDATE 57
3.2.2. SYSTIMESTAMP 58
3.3. 日期转换函数 58
3.3.1. TO_DATE 58
3.3.2. TO_CHAR 59
3.4. 日期常用函数 59
3.4.1. LAST_DAY 59
3.4.2. ADD_MONTHS 59
3.4.3. MONTHS_BETWEEN 60
3.4.4. NEXT_DAY 60
3.4.5. LEAST、GREATEST 60
3.4.6. EXTRACT 61
4. 空值操作 61
4.1. NULL 的含义 61
4.1.1. NULL 的含义 61
4.2. NULL 的操作 61
4.2.1. 插入 NULL 值 61
4.2.2. 更新成 NULL 值 62
4.2.3. NULL 条件查询 62
4.2.4. 非空约束 62
4.3. 空值函数 63
4.3.1. NVL 63
4.3.2. NVL2 63
经典案例 64
1. Oracle 字符串函数综合示例 64
2. Oracle 数值函数综合示例 65
3. Oracle 日期转换综合示例 65
4. Oracle 常用日期函数综合示例 67

达内IT培训集团

5. Oracle 空值函数综合示例 68
课后作业 70
Unit03 73
1. SQL（基础查询）..... 75
1.1. 基本查询语句 75
1.1.1. FROM 子句 75
1.1.2. 使用别名 75
1.1.3. WHERE 子句 75
1.1.4. SELECT 子句 76
1.2. 查询条件 76
1.2.1. 使用>,<.>=,<=，!=，<>,= 76
1.2.2. 使用 AND ,OR 关键字 76
1.2.3. 使用 LIKE 条件(模糊查询) 77
1.2.4. 使用 IN 和 NOT IN 77
1.2.5. BETWEEN...AND… 77
1.2.6. 使用 IS NULL 和 IS NOT NULL 78
1.2.7. 使用 ANY 和 ALL 条件 78
1.2.8. 查询条件中使用表达式和函数 78
1.2.9. 使用 DISTINCT 过滤重复 79
1.3. 排序 79
1.3.1. 使用 ORDER BY 字句 79
1.3.2. ASC 和 DESC 79
1.3.3. 多个列排序 80
1.4. 聚合函数 80
1.4.1. 什么是聚合函数 80
1.4.2. MAX 和 MIN 80
1.4.3. AVG 和 SUM 81
1.4.4. COUNT 81
1.4.5. 聚合函数对空值的处理 81
1.5. 分组 82
1.5.1. GROUP BY 子句 82
1.5.2. 分组查询 82
1.5.3. HAVING 子句 82
1.6. 查询语句执行顺序 83
1.6.1. 查询语句执行顺序 83
2. SQL（关联查询）..... 83
2.1. 关联基础 83
2.1.1. 关联的概念 83
2.1.2. 笛卡尔积 84

2.1.3. 等值连接 84
2.2. 关联查询 84
2.2.1. 内连接 84
2.2.2. 外连接 85
2.2.3. 全外连接 86
2.2.4. 自连接 87
经典案例 88
1. Oracle 基础查询综合示例 88
2. Oracle 分组查询综合示例 89
3. Oracle 关联查询综合示例 92
课后作业 100
Unit04 109
1. SQL（高级查询） 111
1.1. 子查询 111
1.1.1. 子查询在 WHERE 子句中 111
1.1.2. 子查询在 HAVING 子句中 112
1.1.3. 子查询在 FROM 部分 113
1.1.4. 子查询在 SELECT 部分 113
1.2. 分页查询 114
1.2.1. ROWNUM 114
1.2.2. 使用子查询进行分页 114
1.2.3. 分页与 ORDER BY 115
1.3. DECODE 函数 115
1.3.1. DECODE 函数基本语法 115
1.3.2. DECODE 函数在分组查询中的应用 116
1.4. 排序函数 117
1.4.1. ROW_NUMBER 117
1.4.2. RANK 117
1.4.3. DENSE_RANK 118
1.5. 集合操作 119
1.5.1. UNION、UNION ALL 119
1.5.2. INTERSECT 120
1.5.3. MINUS 120
1.6. 高级分组函数 121
1.6.1. ROLLUP 121
1.6.2. CUBE 123
1.6.3. GROUPING SETS 124
经典案例 126
1. Oracle 子查询综合示例 126
2. Oracle 分页查询综合示例 129
3. Oracle 高级查询综合示例 130

达内IT培训集团

课后作业 134
Unit05 138
1. 视图、序列、索引 140
1.1. 视图 140
1.1.1. 什么是视图 140
1.1.2. 视图的作用 140
1.1.3. 授权创建视图 141
1.1.4. 创建简单视图（单表） 141
1.1.5. 查询视图 142
1.1.6. 对视图进行 INSERT 操作 142
1.1.7. 创建具有 CHECK OPTION 约束的视图 143
1.1.8. 创建具有 READ ONLY 约束的视图 143
1.1.9. 通过查询 user_views 获取相关信息 144
1.1.10. 创建复杂视图（多表关联） 145
1.1.11. 删除视图 145
1.2. 序列 146
1.2.1. 什么是序列 146
1.2.2. 创建序列 146
1.2.3. 使用序列 147
1.2.4. 删除序列 148
1.3. 索引 148
1.3.1. 索引的原理 148
1.3.2. 创建索引 149
1.3.3. 创建基于函数的索引 149
1.3.4. 修改和删除索引 150
1.3.5. 合理使用索引提升查询效率 150
2. 约束 150
2.1. 约束概述 150
2.1.1. 约束的作用 150
2.1.2. 约束的类型 151
2.2. 非空约束 151
2.2.1. 建表时添加非空约束 151
2.2.2. 修改表时添加非空约束 152
2.2.3. 取消非空约束 152
2.3. 唯一性约束 152
2.3.1. 什么是唯一性约束 152
2.3.2. 添加唯一性约束 153
2.4. 主键约束 153

2.4.1. 主键的意义 153
2.4.2. 主键选取的原则 154
2.4.3. 添加主键约束 154
2.5. 外键约束 155
2.5.1. 外键约束的意义 155
2.5.2. 添加外键约束 155
2.5.3. 外键约束对一致性的维护 155
2.5.4. 外键约束对性能的降低 156
2.5.5. 关联不一定需要外键约束 156
2.6. 检查约束 157
2.6.1. 什么是检查约束 157
2.6.2. 添加检查约束 157
经典案例 158
1. Oracle 视图操作综合示例 158
2. 通过序列实现自动生成主键 161
3. 外键约束综合示例 164
课后作业 166

达内IT培训集团

Oracle 数据库基础

Unit01

知识体系.....Page 3

数据库原理	数据库简介	文件存储
		DB 和 DBMS
		关系数据库简介
		表的概念
	主流关系型数据库	Oracle 数据库概述
		DB2 数据库概述
		Sybase 数据库概述
		SQL Server 数据库概述
		MySQL 数据库概述
	SQL 概述	结构化查询语言
		数据定义语言（DDL）
		数据操作语言（DML）
		事务控制语言（TCL）
		数据查询语言（DQL）
		数据控制语言（DCL）
	Oracle 数据库安装和访问	Oracle 数据库的安装（Windows）
		Oracle 数据库的安装（Linux）
		远程登录数据库服务器
		使用 SQL*PLUS
		使用 Oracle SQL Developer
SQL（DDL、DML）	Oracle 数据类型	NUMBER
		CHAR
		VARCHAR2
		DATE
	创建表	CREATE 语句
		DESC 语句
		DEFAULT 语句
		NOT NULL
	修改表	修改表名

达内IT培训集团

		增加列
		删除列
		修改列
	DML 语句	INSERT 语句
		UPDATE 语句
		DELETE 语句

经典案例.....Page 17

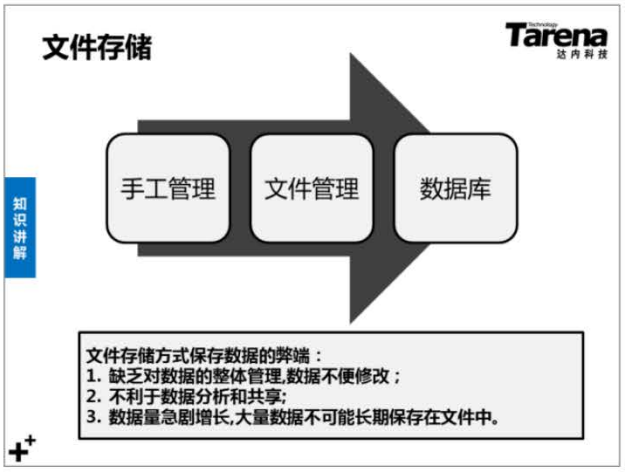
Windows 下 Oracle 数据库安装	Oracle 数据库的安装（Windows）
Linux 下 Oracle 数据库安装	Oracle 数据库的安装（Linux）
使用 telnet 登录数据库服务器	远程登录数据库服务器
使用 SQL* PLUS 访问数据库	使用 SQL*PLUS
安装并使用 Oracle SQL Developer 访问 Oracle	使用 Oracle SQL Developer
NUMBER 数据类型实例	NUMBER
字符数据类型实例	CHAR
	VARCHAR2
日期数据类型实例	DATE
创建员工表	CREATE 语句
	DESC 语句
	DEFAULT 语句
	NOT NULL
修改员工表	修改表名
	增加列
	删除列
	修改列
插入员工数据	INSERT 语句
更改员工数据	UPDATE 语句
删除员工数据	DELETE 语句

课后作业.....Page 45

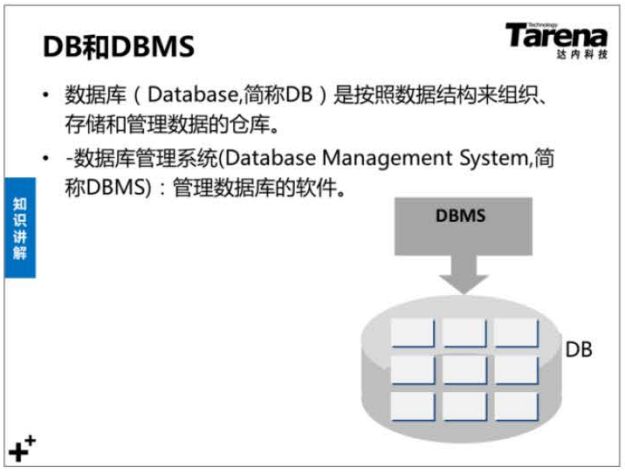
1. 数据库原理

1.1. 数据库简介

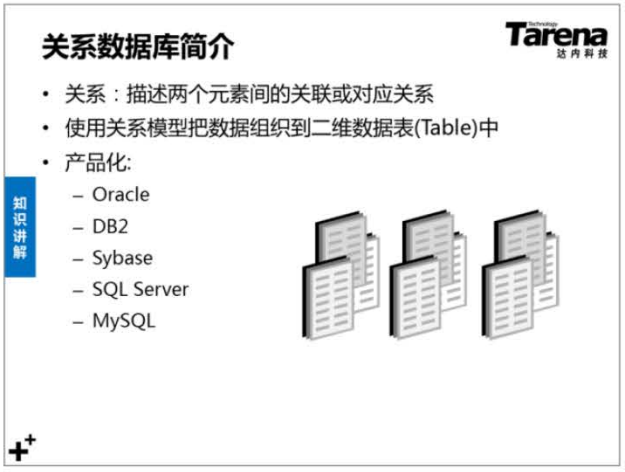
1.1.1. 【数据库简介】文件存储



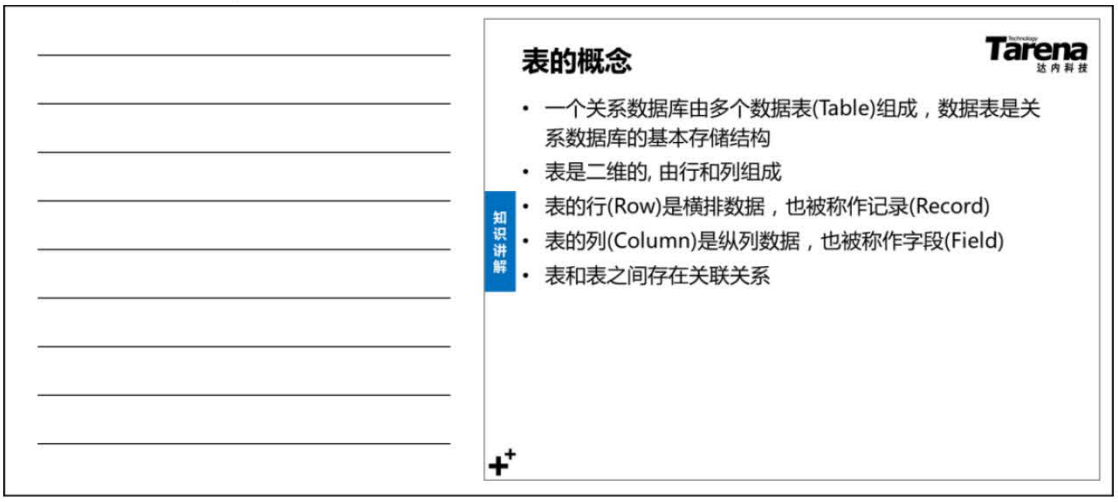
1.1.2. 【数据库简介】DB 和 DBMS



1.1.3. 【数据库简介】关系数据库简介

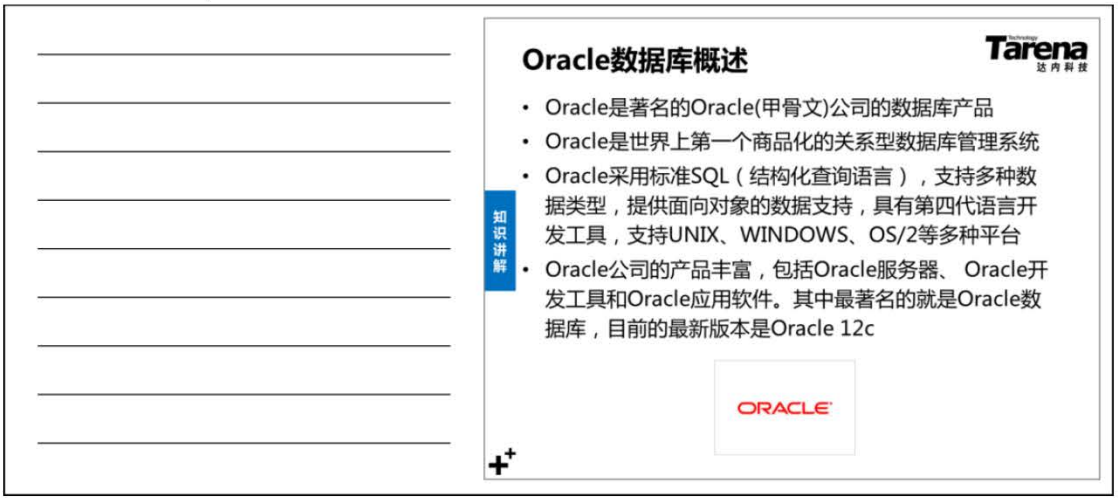


1.1.4. 【数据库简介】表的概念



1.2. 主流关系型数据库

1.2.1. 【主流关系型数据库】Oracle 数据库概述



1.2.2. 【主流关系型数据库】DB2 数据库概述

1.2.3. 【主流关系型数据库】Sybase 数据库概述

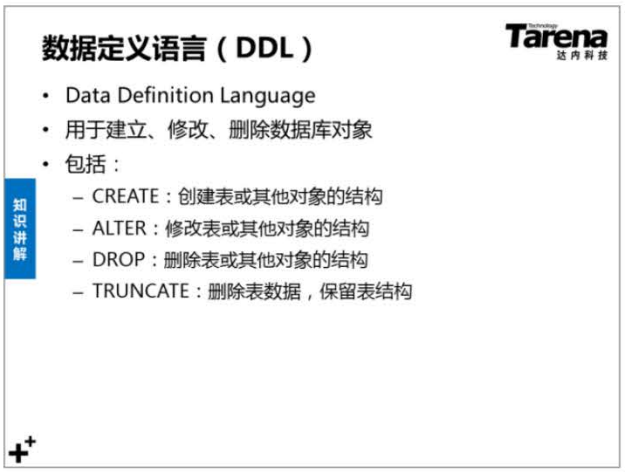
1.2.4. 【主流关系型数据库】SQL Server 数据库概述

1.2.5. 【主流关系型数据库】MySQL 数据库概述

1.3. SQL 概述

1.3.1. 【SQL 概述】结构化查询语言

1.3.2. 【SQL 概述】数据定义语言（DDL）



1.3.3. 【SQL 概述】数据操作语言（DML）

1.3.4. 【SQL 概述】事务控制语言（TCL）

1.3.5. 【SQL 概述】数据查询语言（DQL）

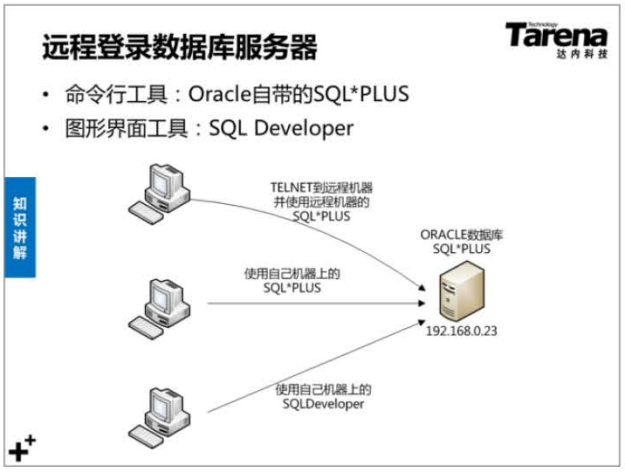
1.3.6. 【SQL 概述】数据控制语言（DCL）

1.4. Oracle 数据库安装和访问

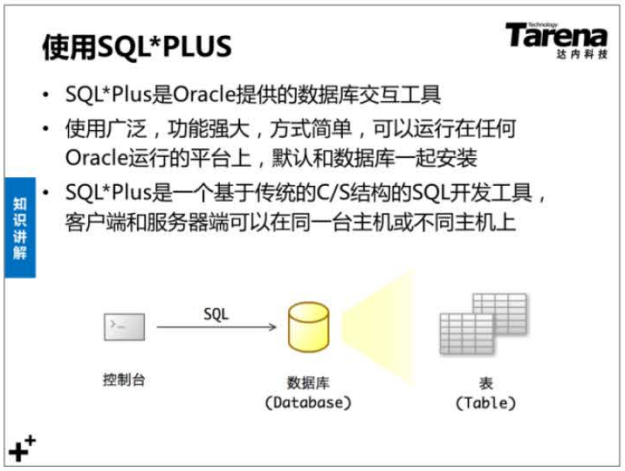
1.4.1. 【Oracle 数据库安装和访问】Oracle 数据库的安装（Windows）

1.4.2. 【Oracle 数据库安装和访问】Oracle 数据库的安装（Linux）

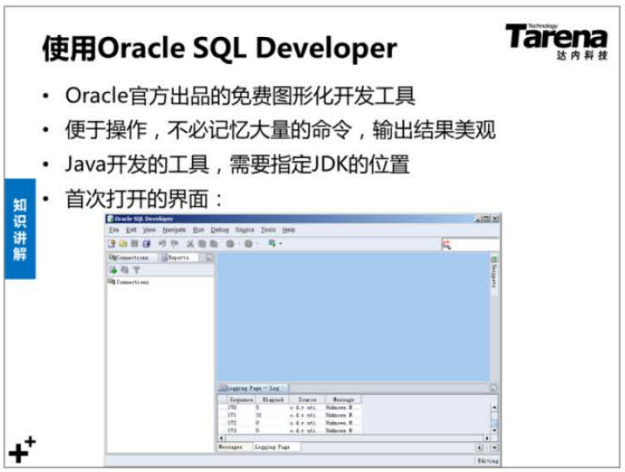
1.4.3. 【Oracle 数据库安装和访问】远程登录数据库服务器



1.4.4. 【Oracle 数据库安装和访问】使用 SQL*PLUS

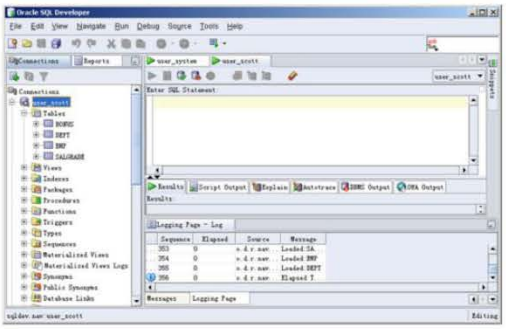


1.4.5. 【Oracle 数据库安装和访问】使用 Oracle SQL Developer



使用Oracle SQL Developer（续1）

- 先建立到指定数据库的连接，在界面中执行SQL语句



解说结束

2. SQL（DDL、DML）

2.1. Oracle 数据类型

2.1.1. 【Oracle 数据类型】NUMBER

NUMBER

- NUMBER表示数字类型
- 经常被定义成NUMBER（P，S）形式，其中：
  - P表示数字的总位数
  - S表示小数点后面的位数
- 例如在表emp中的sal列的定义如下：
  sal NUMBER(6,2)
  表示sal列中的数据，整数位最大为4位，小数位最大位数是2位，也就是最大取值：9999.99

解说结束

2.1.2. 【Oracle 数据类型】CHAR

CHAR

- 表示固定长度的字符类型
- 经常被定义成CHAR（N）形式, N表示占用的字节数
- 最大长度是2000字节
- 例如在表emp中的ename列的定义如下：
  ename CHAR(20)；
  表示ename列中最多可存储20个字节的字符串，并且占用的空间是固定的20个字节。

解说结束

2.1.3. 【Oracle 数据类型】VARCHAR2

VARCHAR2

- 表示变长的字符类型
- 定义格式是VARCHAR2（N），N表示最多可占用的字节数
- 最大长度是4000字节
- 例如在表emp中的job列的定义如下：
  job VARCHAR2(100)；
  表示job列中最多可存储长度为100个字节的字符串。根据其中保存的数据长度，占用的空间是变化的，最大占用空间为100个字节。

2.1.4. 【Oracle 数据类型】DATE

DATE

- 用于定义日期时间的数据
- 长度是7个字节
- 默认格式是：DD-MON-RR, 例如：11-APR-71
- 例如在表emp中的hiredate列的定义如下：
  hiredate DATE；
  表示hiredate列中存放的是日期数据

2.2. 创建表

2.2.1. 【创建表】CREATE 语句

CREATE语句

```
CREATE TABLE [schema.]table_name(
   column_name datatype[DEFAULT expr][,...]
);
```

```
--创建表employee
CREATE TABLE employee(
        id NUMBER(4),
        name VARCHAR2(20),
        gender CHAR(1),
        birth DATE,
        salary NUMBER(6,2),
        job VARCHAR2(30),
        deptno NUMBER(2)
);
```

2.2.2. 【创建表】DESC 语句

DESC语句

DESC table_name;

```
--查看表的结构
DESC employee

名称                                    是否为空? 类型
----------------------------------------- -------- ----------------

ID                                        NUMBER(4)
NAME                                      VARCHAR2(20)
GENDER                                    CHAR(1)
BIRTH                                     DATE
SALARY                                    NUMBER(6,2)
JOB                                       VARCHAR2(30)
DEPTNO                                    NUMBER(2)
```

2.2.3. 【创建表】DEFAULT 语句

DEFAULT语句

- 可以通过DEFAULT子句给列指定默认值
- 给gender列赋默认值‘M’,如果没有指定性别的员工，默认是男性。

```
--先将employee表删除，再创建表employee
DROP TABLE employee;
CREATE TABLE employee(
        id NUMBER(4),
        name VARCHAR2(20),
        gender CHAR(1) DEFAULT 'M',
        birth DATE,
        salary NUMBER(6,2),
        job VARCHAR2(30),
        deptno NUMBER(2)
);
```

2.2.4. 【创建表】NOT NULL

NOT NULL

- 非空(Not Null)是一种约束条件，用于确保字段值不为空
- 默认情况下，任何列都允许有空值
- 当某个字段被设置了非空约束条件，这个字段中必须存在有效值
- 当执行插入数据的操作时，必须提供这个列的数据
- 当执行更新操作时，不能给这个列的值设置为NULL

NOT NULL（续1）

解详识知

```
--创建表employee
DROP TABLE employee;
CREATE TABLE employee(
        id NUMBER(4),
        name VARCHAR2(20) NOT NULL,
        gender CHAR(1),
        birth DATE,
        salary NUMBER(6,2),
        job VARCHAR2(30),
        deptno NUMBER(2)
);
```

2.3. 修改表

2.3.1. 【修改表】修改表名

修改表名

解详识知

- 在建表后如果希望修改表名，可以使用RENAME语句实现
- 语法如下，将改变表名old_name为new_name：

RENAME old_name TO new_name;

```
--修改表名employee为myemp
RENAME employee TO myemp;
```

2.3.2. 【修改表】增加列

增加列

解详识知

- 给表增加列可以使用ALTER TABLE的ADD子句实现。
- 语法：

ALTER TABLE table_name **ADD**
(column datatype [DEFAULT expr] [, column datatype...])

- 列只能增加在最后，不能插入到现有的列中

```
--给表增加一列hiredate，并设置默认值为当前日期
ALTER TABLE myemp ADD (hiredate DATE DEFAULT sysdate);
```

2.3.3. 【修改表】删除列

删除列

- 使用ALTER TABLE的DROP子句删除不需要的列。
- 语法如下
- ALTER TABLE table_name **DROP** (column);
- 删除字段需要从每行中删掉该字段占据的长度和数据，并释放在数据块中占据的空间，如果表记录比较大，删除字段可能需要比较长的时间。

```
--删除表myemp的列hiredate
ALTER TABLE myemp DROP (hiredate);
```

2.3.4. 【修改表】修改列

修改列

- 建表之后，可以改变表中列的数据类型、长度和默认值
- 修改仅对以后插入的数据有效
- 如果把长度由大改小，有可能不成功
- 语法：
  ALTER TABLE table_name **MODIFY**
  (column datatype [DEFAULT expr] [, column datatype...])

```
--修改表myemp的列job，并增加默认值的设置
ALTER TABLE myemp
MODIFY(job VARCHAR2(40) DEFAULT 'CLERK' );
```

2.4. DML 语句

2.4.1. 【DML 语句】INSERT 语句

INSERT语句

- 给数据表增加记录
- 语法如下：
  INSERT INTO table_name[(column[, column...])]
  VALUES(value[, value...]);
- 执行DML操作后，需要再执行commit语句，才算真正确认了此操作

```
--插入一条记录
INSERT INTO myemp(id, name, job, salary)
VALUES(1001, 'rose', 'PROGRAMMER', 5500);
```

INSERT语句（续1）

- 如果插入的列有日期字段，需要考虑日期的格式
- 默认的日期格式 'DD-MON-RR'
- 可以自定义日期格式，用TO_DATE函数转换为日期类型的数据

解详识知

```
--使用默认日期格式插入记录
INSERT INTO myemp (id, name, job, birth)
VALUES(1002, 'martha', 'ANALYST', '01-SEP-03');

--使用自定义日期格式插入记录
INSERT INTO myemp (id, name, job,birth)
VALUES(1003, 'donna', 'MANAGER',
TO_DATE('2009-09-01', 'YYYY-MM-DD'));
```

2.4.2. 【DML 语句】UPDATE 语句

UPDATE语句

- 更新表中的记录
- 语法如下：

  UPDATE table_name
  SET column = value [, column = value]...
  [WHERE condition];
- 如果没有WHERE子句，则全表的数据都会被更新，务必小心

解详识知

```
--更改职员ROSE的薪水为8500
UPDATE myemp SET salary = 8500
WHERE name = 'ROSE';
```

2.4.3. 【DML 语句】DELETE 语句

DELETE语句

- 删除表中的记录
- 语法如下：

  DELETE [FROM] table_name [WHERE condition];
- 如果没有WHERE子句，**则全表的数据都会被删除！**

解详识知

```
--删除职位是空的员工记录
DELETE FROM myemp WHERE job is null;
```

DELETE语句（续1）

- 在DDL语句中的TRUNCATE语句，同样有删除表数据的作用。
- 和DELETE语句的区别：
  - DELETE可以有条件删除，TRUNCATE将表数据全部删除
  - DELETE是DML语句，可以回退，TRUNCATE是DDL语句，立即生效，无法回退
  - 如果是删除全部表记录，且数据量较大，DELETE语句效率比TRUNCATE语句低。

知识讲解

```
--删除全部记录
 DELETE FROM myemp;
--或者
TRUNCATE  TABLE myemp;
```

经典案例

1. Windows 下 Oracle 数据库安装

- **问题**

在 Windows 下安装 Oracle 数据库。

- **步骤**

通过网址“http://www.oracle.com”打开 Oracle 官网首页，点击 Downloads 进入 Oracle 数据库的下载页面，如图-1 所示：

图-1

安装步骤如下（以本地安装为例）：

步骤一：解压，双击 setup.exe，启动安装向导

首先，将下载的 oracle 安装包进行解压缩；然后，双击 setup.exe 文件来启动安装向导，打开安装界面，如图-2 所示：

图-2

在此步骤中需要设置的内容如下：

1）选择基本安装，指定要安装的 Oracle 数据库的基位置和主目录位置；

2）安装类型为企业版；

3）选择“创建启动数据库”（可以在安装数据库软件同时创建数据库，也可以在安装完毕后单独创建，此处选择默认值，即为安装数据软件同时创建数据库）；

4）指定全局数据库名称为 orcl ,以及口令 ,需要记住自己的密码 ,这里设置为 oracle。

设置完成以上内容后，点击“下一步”按钮。

步骤二：检测安装环境

点击“下一步”按钮后，进入检查安装环境的界面，如图-3 所示：该步骤用来检查软硬件环境是否符合 Oracle11g 的安装要求，如果满足安装要求，则点击“下一步”按钮。

图-3

步骤三：进入安装概要页面

检查安装环境通过后，进入安装概要页面，如图-4 所示：在该步骤可以检查之前的设置，包括安装类型和要安装的组件，如果有问题，可以点击“上一步”按钮回去修改，如果确认没问题，则点击“安装”按钮，进入安装阶段。

图-4

步骤四：提示正在安装。

根据机器的软硬件配置不同，这个步骤可能耗费不同的时间。图略。

步骤五：进入配置助手步骤，这个过程不需要用户干涉。图略。

步骤六：创建数据库，图略。

步骤七：数据库创建完成后，提示配置助手界面，显示数据库的初始信息。图略。

在此步骤中可以点击“口令管理”按钮进行数据库初始用户的口令管理，这个步骤此时可以忽略，待安装完成后再进行口令管理。完成后，点击“确定”按钮离开口令管理界面。

步骤八：弹出安装成功窗口，图略。

步骤九：在安装成功界面上点击“下一步”按钮，提示安装结束，如图-5 所示：

图-5

2. Linux 下 Oracle 数据库安装

- **问题**

如何在 Linux 系统下安装 Oracle 数据库。

- **步骤**

在 Linux 下安装 Oracle 数据库的步骤如下：

步骤一： 检查安装 oracle 时所需要的 RPM 包是否存在

检查安装 oracle 时所需要的 RPM 包命令为： rpm -q 指定的 RPM 包。

需要检查的 RPM 包如下：

```
binutils-2.17.50.0.6
compat-libstdc++-33-3.2.3
elfutils-libelf-0.125
elfutils-libelf-devel-0.125
elfutils-libelf-devel-static-0.125
gcc-4.1.2
gcc-c++-4.1.2
glibc-2.5-24
glibc-common-2.5
glibc-devel-2.5
glibc-headers-2.5
kernel-headers-2.6.18
ksh-20060214
libaio-0.3.106
libaio-devel-0.3.106
libgcc-4.1.2
libgomp-4.1.2
libstdc++-4.1.2
libstdc++-devel-4.1.2
make-3.81
sysstat-7.0.2
```

如果以上 RPM 包没有被安装，请到 CentOS5.5 的 ISO 文件中找到这些包并上传到 linux 系统中。已上传的 RPM 包，双击就可以安装，界面安装的好处是系统可以自己解决依赖关系（保持你的网络畅通）。

步骤二：配置 oinstall 组

配置 oinstall 组，命令如下：

```
/user/sbin/groupadd oinstall
```

步骤三:配置 dba 组合 oper 组

配置 dba 组合 oper 组，与配置 oinstall 命令相同，只是把 oinstall 换成 dba 和 oper，命令如下：

```
groupadd dba
groupadd oper
```

步骤四:创建用户 oracle 并且授权

创建用户 oracle 并且授权，命令如下：

```
useradd -g oinstall -G dba,oper oracle
```

步骤五：设置 oracle 用户的密码

设置 oracle 用户的密码，命令如下：

```
passwd oracle
```

步骤六：确定 nobody 用户不存在

确定 nobody 用户不存在，命令如下：

```
id nobody
```

步骤七：编辑 sysctl.conf 文件

编辑 sysctl.conf 文件，命令如下：

```
vi /etc/sysctl.conf
```

添加如下内容：

```
kernel.sem = 250 32000 100 128
fs.file-max = 6815744
net.ipv4.ip_local_port_range = 9000 65500
net.core.rmem_default = 262144
net.core.rmem_max = 4194304
net.core.wmem_default = 262144
net.core.wmem_max = 1048576
fs.aio-max-nr = 1048576
```

步骤八：使内核参数立即生效

```
sysctl -p
```

步骤九：编辑 limits.conf 文件

使用 vi 编辑 limits.conf 文件，命令如下：

```
vi/etc/security/limits.conf
```

添加如下内容：

```
oracle soft nproc 2047
oracle hard nproc 16384
oracle soft nofile 1024
oracle hard nofile 65536
```

步骤十：编辑 login 文件

使用 vi 编辑 login 文件，命令如下：

```
vi /etc/pam.d/login
```

添加如下选项：

```
session required pam_limits.so
```

步骤十一：创建安装 oracle 的文件夹

创建安装 oracle 的文件夹，命令如下：

```
mkdir -p /opt/db/oracle
```

步骤十二：更改权限

更改权限，命令如下：

```
chown -R oracle:oinstall /opt/db/oralce
chmod -R 775 /opt/db/oracle
```

步骤十三：切换用户到 oracle

切换用户到 oracle，命令如下：

```
su - oracle
```

此处注意空格。

步骤十四：编辑.bash_profile 文件

```
[root@oracle ~]# vi /etc/profile
if [$USER = "oracle"]; then
  if [$SHELL = "/bin/ksh"]; then
    ulimit -p 16384
    ulimit -p 65536
  else
    ulimit -u 16384 -n 65536
  fi
  umask 022
fi
```

以 oracle 用户身份登录，修改当前用户下的.bash_profile(这是用户主目录下的一个隐藏文件)。

```
PATH=$PATH:$HOME/bin;export PATH
ORACLE_BASE=/opt/oracle;export ORACLE_BASE
ORACLE_HOME=/opt/db/oracle/1102/db01; export ORACLE_HOME
ORACLE_SID=testdb;export ORACLE_SID
PATH=$ORACLE_HOME/bin:$PATH:$HOME/bin;export PATH
DISPLAY=192.168.1.88:0.0;export DISPLAY
```

步骤十五：上传 oracle 到/opt 目录

以 root 用户登录或者是把/opt 的权限赋值给 oracle：

```
chown -R oracle:oinstall /opt/db/oralce
```

这样就可以用 oracle 用户登录然后直接上传。

步骤十六：查看上传情况

查看上传情况，命令如下：

```
cd  /opt
ls
```

步骤十七：解压上传的 oracle 安装包

上传完毕后，使用 unzip 命令解压 oracle 安装包（如果你没有把/opt 的权限赋值给 oracle 那么将无法正常解压），解压命令如下：

```
Unzip linux_11gR2_database_1of2.zip
Unzip linux_11gR2_database_2of2.zip
```

步骤十八：使用 oracle 用户登录系统

解压完成以后用 oracle 用户登录系统，如图-6 所示：

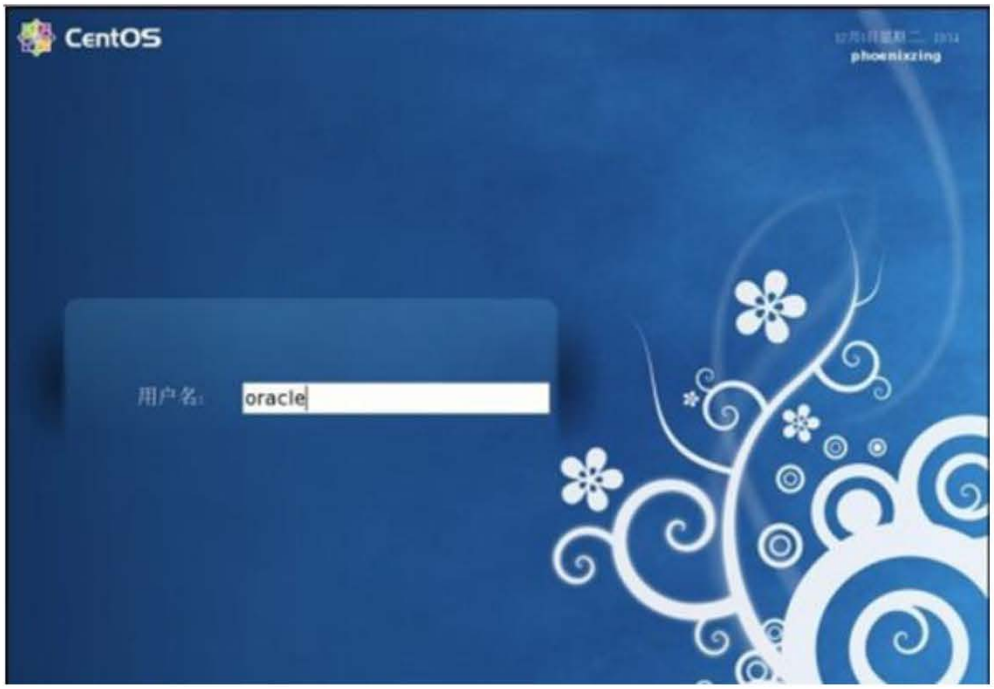


图-6

步骤十九：找到你上传的 oracle 文件的位置

在 linux 系统中找到 oracle 文件存储的位置，如图-7 所示：

图-7

步骤二十：oracle 解压完以后产生一个 database 文件夹打开该文件夹

oracle 解压完以后会产生一个 database 文件夹，打开该文件夹，如图-8 所示：

图-8

步骤二十一：运行 runinstaller

运行 runinstaller，如图-9 所示：

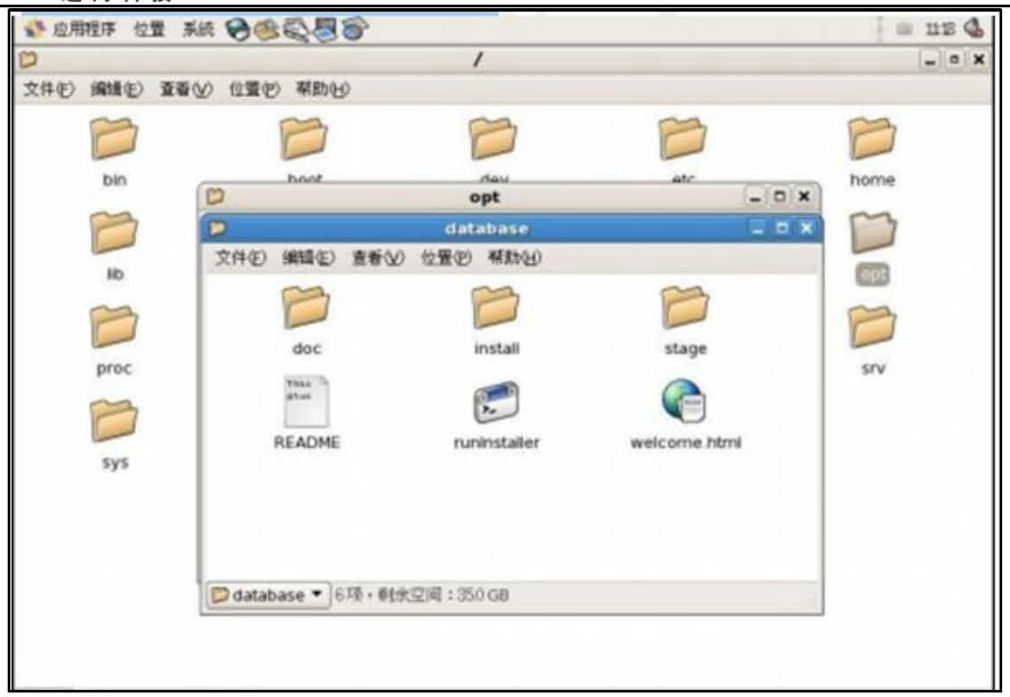


图-9

步骤二十二：点击运行

弹出界面如图-10 所示：

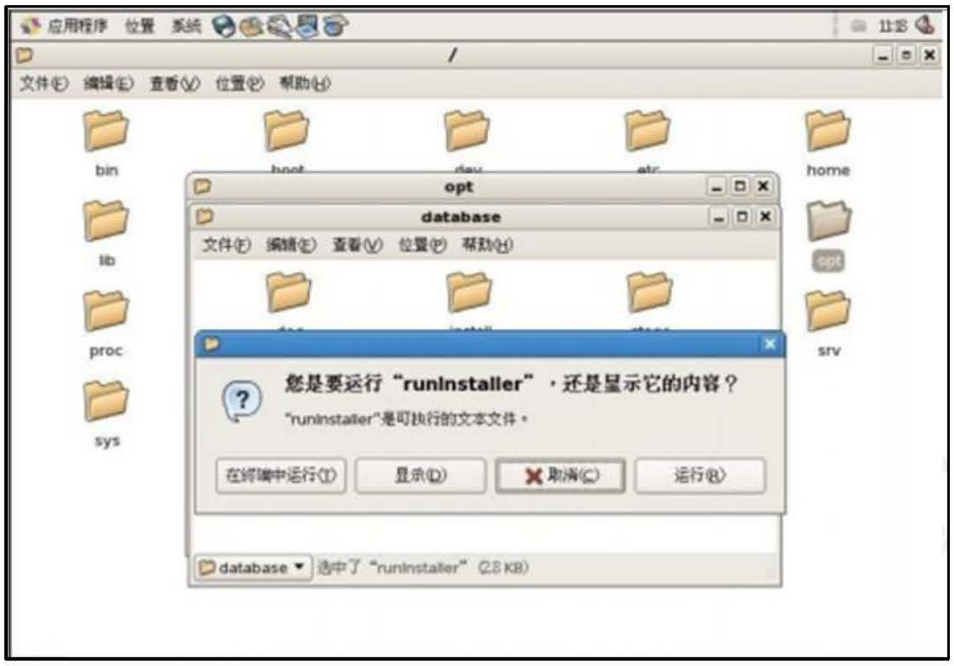


图-10

步骤二十三：安装界面

注意：创建启动数据库时，全局数据库名必须要与你在设置.base_profile 中输入的 oracle_sid 一致。你可以设置密码，在这里设置密码以后，帐户 sys、system、sysman 和 DBSNMP 都将使用该处设置的密码，如图-11 所示：

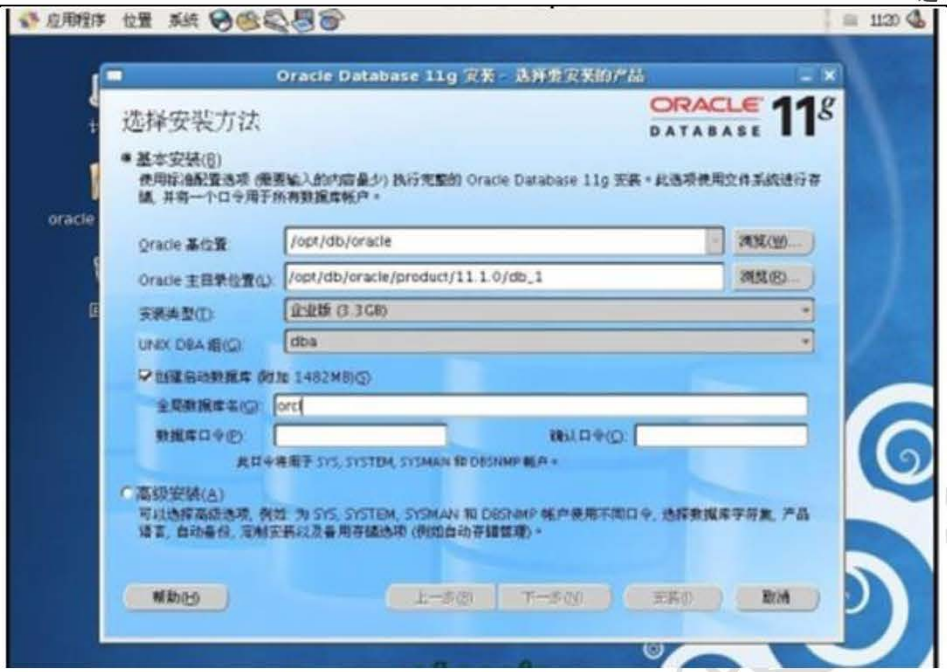


图-11

开始安装，如图-12 所示：

图-12

步骤二十四：指定产品清单目录的完整路径

指定产品清单目录的完整路径，默认即可，如图-13 所示：

图-13

步骤二十五：检查先决条件

检查界面如图-14 所示：

图-14

步骤二十六：网络配置检查

如果是 DHCP 网络，网络配置要求如图-15 所示；如果指定 IP 地址，将不会出现这个提示，则可直接进行下一步。

图-15

由于 DHCP 网络所出现的提示，无需处理，选择“是”，如图-16 所示：

图-16

步骤二十七：下一步

如图-17 所示：

图-17

步骤二十八：下一步

要安装的内容如图-18 所示：

图-18

步骤二十九：oracle 数据库安装中

点击“安装‘按钮后，如图-19 所示：

图-19

步骤三十：安装完数据库以后将出现以下两个脚本，需要用 root 用户运行

如图-20 所示：

图-20

步骤三十一：以 root 用户登录系统，利用 sh 命令运行（注意大小写）

如图-21 所示：

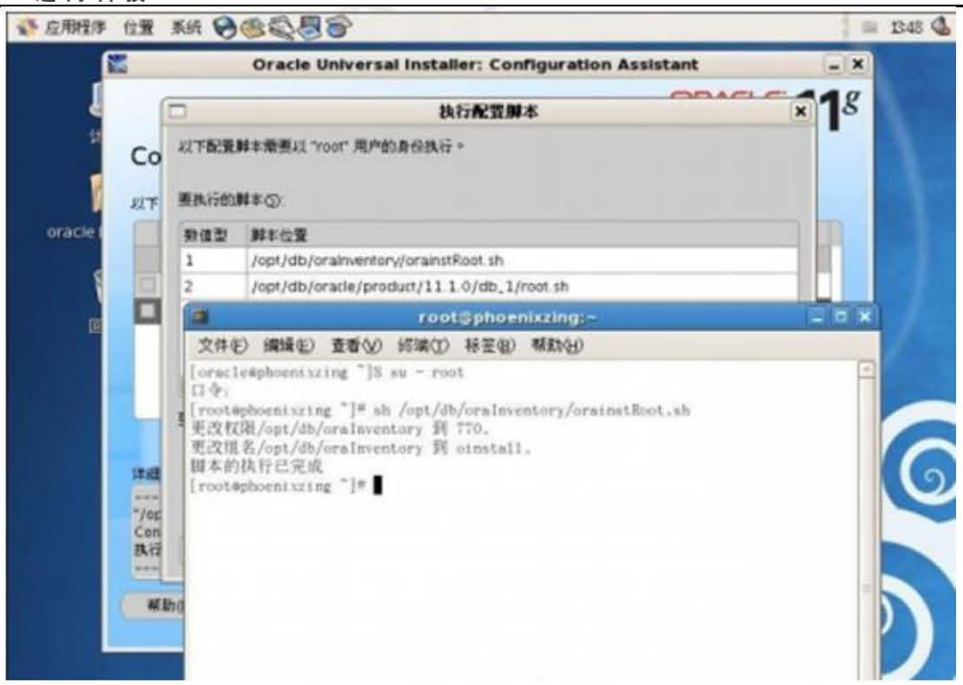


图-21

步骤三十二：oracle 安装完成（重新启动系统）

如图-22 所示：

图-22

3. 使用 telnet 登录数据库服务器

- **问题**

如何使用 telnet 登录数据库服务器。

- **方案**

远程登录到数据库所在的机器上，输入远程机器操作系统的帐号和密码，如图-23 所示：

```
soft01@java3g:~
文件(F) 编辑(E) 查看(V) 终端(T) 标签(B) 帮助(H)
[soft01@java3g ~]$ telnet 192.168.0.23
Trying 192.168.0.23...
Connected to 192.168.0.23 (192.168.0.23).
Escape character is '^]'.
login: openlab
Password:
Last login: Tue Apr  8 16:09:37 from 192.168.160.254
Sun Microsystems Inc.   SunOS 5.10      Generic January 2005
sun280%
```

图-23　远程登录数据库服务器

4. 使用 SQL* PLUS 访问数据库

- **问题**

如何使用 SQL* PLUS 访问数据库。

- **方案**

在上一案例的基础上，在%提示符下，输入 SQLPlus 命令，并输入数据库的帐号和密码，如图-24 所示：

```
soft01@ja
文件(F) 编辑(E) 查看(V) 终端(T) 标签(B) 帮助(H)
Last login: Tue Apr  8 16:09:37 fro
Sun Microsystems Inc.   SunOS 5.10
sun280% sqlplus

SQL*Plus: Release 10.2.0.1.0 - Prod

Copyright (c) 1982, 2005, Oracle.

Enter user-name: openlab
Enter password:
```

图-24　运行 sqlplus，登录数据库

如果出现 SQL 提示符，则表示登录成功，如图-25 所示：

```
Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
uction
With the Partitioning, OLAP and Data Mining options

SQL>
```

图-25　登录数据库成功

退出 SQL*PLUS 时，在 SQL>后输入 exit 即可,如图-26 所示：

```
SQL> exit
Disconnected from Oracle Database 10g Er
0.1.0 - 64bit Production
With the Partitioning, OLAP and Data Mir
sun280%
```

图-26　退出 SQL*Plus

在此输入 exit，即退出远程服务器。

5. 安装并使用 Oracle SQL Developer 访问 Oracle

- **问题**

如何安装并使用 Oracle SQL Developer 访问 Oracle。

- **步骤**

Oracle SQL Developer 是 Oracle 官方出品的免费图形化开发工具，相对 SQL*Plus 来说，图形化的界面便于操作，不必记忆大量的命令，输出结果美观。它的基本功能包括结果的格式化输出，编辑器自动提示，代码美化，显示 SQL 的执行计划，监控会话，编写以及调试存储过程等。官方和免费两个特征让这个工具极具吸引力。

步骤一：下载 Oracle SQL Developer

通过如下所示的网址进入 Oracle SQL Developer 的下载页:

```
http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html
```

下载页面如图-27 所示，可以在下载页面选择所需平台的版本进行下载。

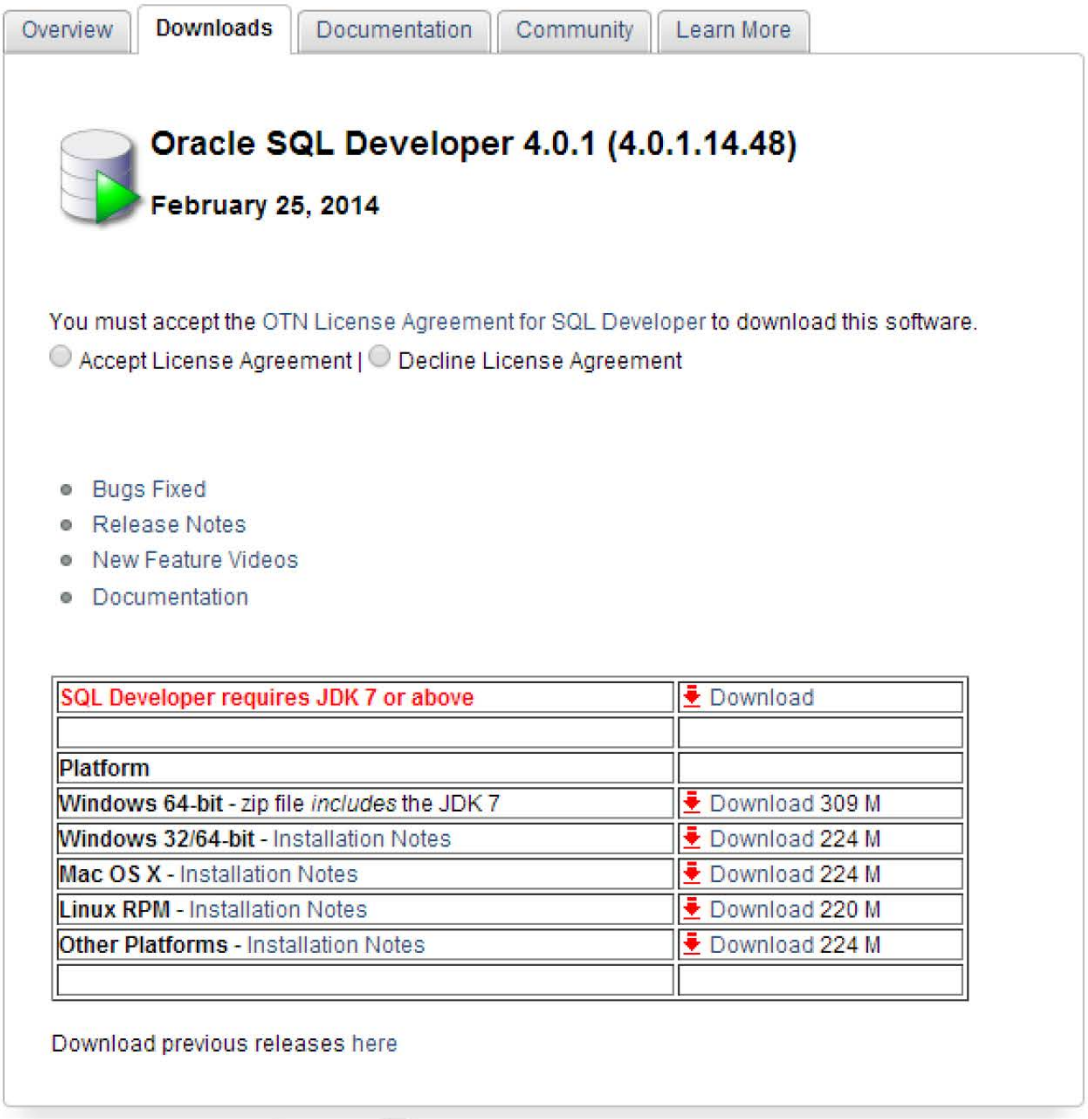


图-27

步骤二：安装

Oracle Developer 安装过程十分简单，只需要将下载的压缩包进行解压缩即可。

步骤三：使用

解压缩后的文件中有一个“sqldeveloper.exe”，双击其进入 SQL Developer 工具的主界面，如图-28 所示：

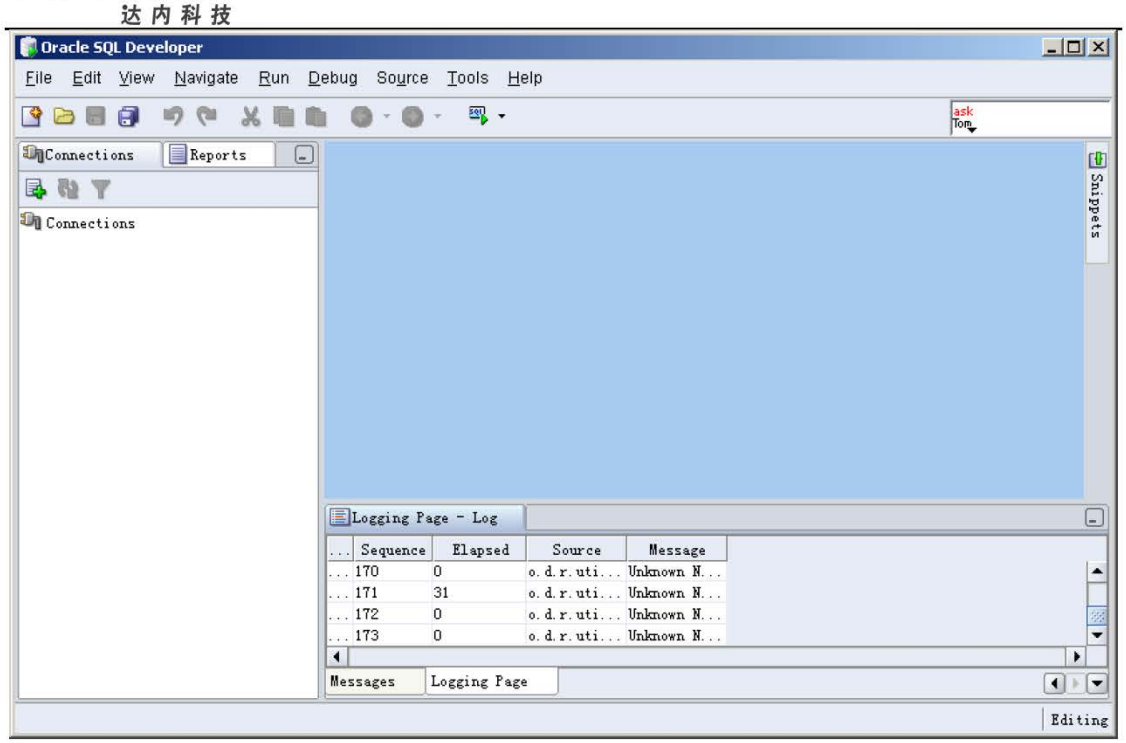


图-28

第一次进入 SQL Developer，需要设置连接参数。首先在窗口左边的 Connections 上按右键，开启一个新的连接，输入远程数据库的参数，界面如图-29 所示：

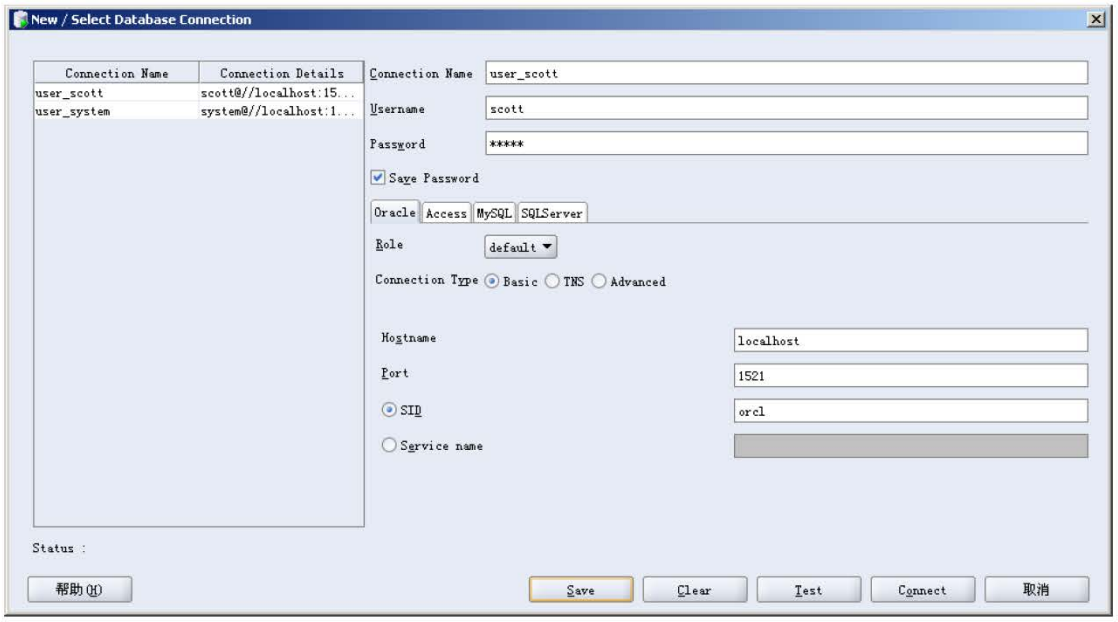


图-29

成功建立连接之后，在连接上按右键点击 Connect，将打开对应用户方案中的数据库对象，如图-30 所示：

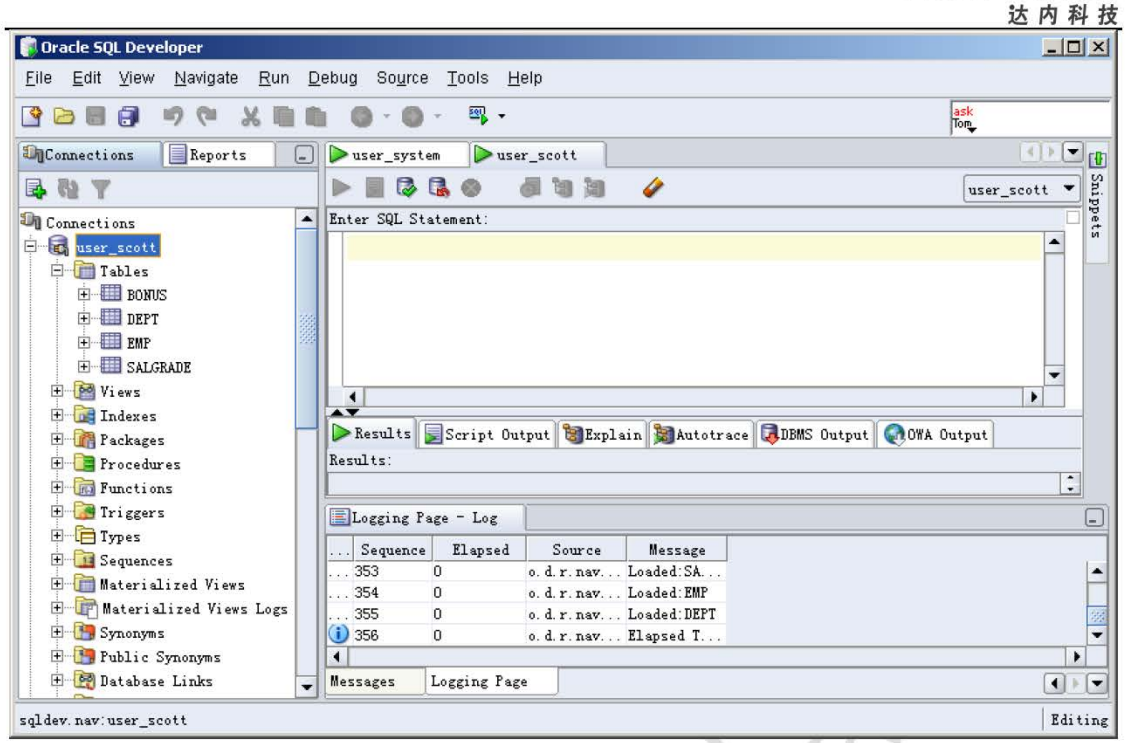


图-30

在此界面，即可操作 SQL 语句。

6. NUMBER 数据类型实例

- **问题**

创建学员信息表 student，使用 NUMBER 数据类型定义 student 表的 id 字段的总数字位数为 4，定义 fee 字段的总数字位数为 7，小数位数为 2。

- **方案**

NUMBER 表示数字类型，经常被定义成 NUMBER（P，S）形式，其中，P 表示数字的总位数；S 表示小数点后面的位数。

在表 student 中的 id 列和 fee 列的定义如下：

```
CREATE TABLE student(
 id NUMBER(4),
 name VARCHAR2(20),
 gender CHAR(1),
 fee NUMBER(7,2)
);
```

上述定义表示 id 列中的数据，整数位最大为 4 位，即最大值为 9999；fee 列整数位最大为 5 位，小数位最大位数是 2 位，即最大取值为 99999.99。

7. 字符数据类型实例

- **问题**

创建学员信息表 student1 ,使用 VARCHAR2 数据类型定义 student1 表的 name 字段，该字段的长度为最大 20 个字节变长字符串；使用 CHAR 数据类型定义 gender 字段，该字段的长度为 1 个字节定长字符串；使用 VARCHAR2 数据类型定义 residence 字段，该字段的长度为 100 个字节定长字符串。

- **方案**

在表 student1 中的 name 列、gender 列和 residence 列的定义如下：

```
CREATE TABLE student1(
 id NUMBER(4),
 name VARCHAR2(20),
 gender CHAR(1),
 residence CHAR(100)
);
```

上述代码表示 student1 表的 name 字段的长度为最大 20 个字节变长字符串；gender 字段的长度为 1 个字节定长字符串；residence 字段的长度为 100 个字节定长字符串。

8. 日期数据类型实例

- **问题**

创建学员信息表 student2，使用 DATE 数据类型定义 student2 表的 birth 字段为日期类型。

- **方案**

DATE 类型用于定义日期时间的数据，长度是 7 个字节，默认格式是：DD-MON-RR，例如："11-APR-71"，表示 1971 年 4 月 11 日；如果是中文环境，则格式为 "11-4 月-71"。

在表 student2 中的 birth 列的定义如下：

```
CREATE TABLE student2(
 id NUMBER(4),
 name VARCHAR2(20),
 gender CHAR(1),
 birth DATE
);
```

上述定义表示 birth 列中存放的是日期类型数据。

9. 创建员工表

- **问题**

员工表的信息如表-1 所示：

表-1 员工表信息

字段名	类型	值情况	描述
id	NUMBER(4)	NOT NULL	员工 ID
name	VARCHAR2(20)	NOT NULL	员工姓名
gender	CHAR(1)	默认值为“M”	员工性别
birth	DATE	NULL	员工生日
salary	NUMBER(6,2)	NULL	员工工资
comm	NUMBER(6,2)	NULL	员工绩效
job	VARCHAR2(30)	NULL	员工职位
manager	NUMBER(4)	NULL	员工的管理者 ID
deptno	NUMBER(2)	NULL	员工所在的部门 ID

请根据上述表信息创建员工表 employee。

- **方案**

使用“CREATE TABLE”语句创建表，语法是：

```
CREATE TABLE [schema.]table_name(
 column_name datatype[DEFAULT expr][,…]
);
```

上述创建表的语法表示：

1）“CREATE TABLE”为固定写法；

2）使用“[]”括起来的部分可以省略，[schema.]表示该表所属的用户，默认是当前登录 oracle 的用户；

3）table_name 表示表名，根据实际情况替换该名称，此案例的表名为 employee；

4）column_name 表示列名，根据实际情况替换该名称，此案例中有 9 列，列名如表-1 所示；

5）datatype 表示对应列的数据类型，此案例中 9 列对应的数据类型如表-1 所示；

6）[DEFAULT expr]表示给该列的默认值，可以省略；

7）[,…]表示多列之间使用“,”逗号做分隔，最后一列不使用逗号。

- **完整代码**

本案例中，创建 employee 表的 SQL 语句如下：

```
CREATE TABLE employee(
 id NUMBER(4),
```

```
name VARCHAR2(20)  NOT NULL,
gender CHAR(1) DEFAULT 'M',
birth DATE,
salary NUMBER(6,2),
comm NUMBER(6,2),
job VARCHAR2(30),
manager NUMBER(4),
deptno NUMBER(2)
);
```

10. 修改员工表

- **问题**

修改员工表后，显示表结构。详细要求如下：

1）修改 employee 表的表名为 myemp；

2）向 myemp 表增加一列 hiredate，并设置默认值为当前日期；

3）修改 myemp 表中的 job 列的长度为 40，并增加默认值的设置，默认值为‘CLERK’；

4）删除 myemp 表中的 hiredate 列。

- **方案**

1）在建表后如果希望修改表名，可以使用 RENAME 语句实现，语法如下：

```
RENAME old_name TO new_name;
```

上述 SQL 语句表示改变表名 old_name 为 new_name。

2）在建表之后，要给表增加列可以使用 ALTER TABLE 的 ADD 子句实现，该语句的语法如下：

```
ALTER TABLE table name ADD
 (column datatype [DEFAULT expr] [, column datatype…]);
```

注意一点，列只能增加在最后，不能插入到现有列的中间。

3）建表之后，可以改变表中列的数据类型、长度和默认值，注意这种修改仅对以后插入的数据有效；另外，如果表中已经有数据的情况下，把长度由大改小，有可能不成功，比如原来的类型是 VARCHAR2(100),其中已经存放了 100 个字节长度的数据，如果要改为 VARCHAR2（80），则不会修改成功。该语句的语法如下：

```
ALTER TABLE table name MODIFY
(column datatype [DEFAULT expr] [, column datatype…]);
```

4）在建表之后，使用 ALTER TABLE 的 DROP 子句删除不需要的列，该语句的语法如下：

```
ALTER TABLE table_name DROP (column);
```

删除字段需要从每行中删掉该字段占据的长度和数据，并释放在数据块中占据的空间，

如果表记录比较大，删除字段可能需要比较长的时间。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一:修改表名

将 employee 表的表名修改为 myemp，使用如下 SQL 语句：

```
RENAME employee TO myemp;
```

显示表结构的语句如下所示：

```
DESC myemp;
```

步骤二：增加列

给 myemp 表增加一列 hiredate，并设置默认值为当前日期，使用的 SQL 语句如下：

```
ALTER TABLE myemp ADD (hiredate DATE DEFAULT sysdate);
```

上述 SQL 语句中的 sysdate 表示当前日期。
显示表结构的语句如下所示：

```
DESC myemp;
```

步骤三：修改列

修改 myemp 表中的 job 列的长度为 40，并增加默认值的设置，默认值为‘CLERK’，使用的 SQL 语句如下：

```
ALTER TABLE myemp
MODIFY(job VARCHAR2(40) DEFAULT 'CLERK' );
```

显示表结构的语句如下所示：

```
DESC myemp;
```

步骤四：删除列

删除表 myemp 中的 hiredate 列，使用的 SQL 语句如下：

```
ALTER TABLE myemp DROP (hiredate);
```

显示表结构的语句如下所示：

```
DESC myemp;
```

11. 插入员工数据

- **问题**

向 myemp 表中插入三条记录，该三条记录的数据如表-2 所示：

表- 2 myemp 表

id	name	job	salary	birth
1001	rose	PROGRAMMER	5500	
1002	martha	ANALYST		01-9 月-89
1003	donna	MANAGER		1978-09-01

- **方案**

INSERT 语句用来给数据表增加记录，每次增加一条记录，该 SQL 语句的语法如下：

```
INSERT INTO table name[(column[, column…])]
VALUES(value[, value…]);
```

需要注意的一点，所有的 DML 操作，都需要执行事务提交语句 commit，才算真正确认了此操作。

- **步骤**

步骤一：插入数据

向myemp表的列id、name、job、salary插入数据，这些列的数据分别为 1001、'rose'、'PROGRAMMER'、5500，SQL 语句如下所示：

```
INSERT INTO myemp (id, name, job, salary)
VALUES(1001, 'rose', 'PROGRAMMER', 5500);
```

步骤二：插入日期数据

Oracle 中的日期数据比较特殊，如果插入的列有日期字段，需要考虑日期的格式。Oracle 的默认的日期格式为 'DD-MON-RR '。可以按照默认格式插入数据，也可以自定义日期格式，用 TO_DATE 函数转换为日期类型的数据。使用默认日期格式插入数据的 SQL 语句如下：

```
--使用默认日期格式插入记录
INSERT INTO myemp (id, name, job,birth)
VALUES(1002, 'martha', 'ANALYST', '01-SEP-89');
```

如果为中文环境，那么日期的格式应该为：'01-9 月-89'。

使用自定义日期格式插入数据的 SQL 语句如下：

```
--使用自定义日期格式插入记录
INSERT INTO myemp (id, name, job, birth)
VALUES(1003, 'donna', 'MANAGER',
     TO_DATE('1978-09-01', 'YYYY-MM-DD'));
```

- **完整代码**

本案例的完整代码请参考步骤。

12. 更改员工数据

- **问题**

本案例的详细要求如下：

1）将员工表中姓名为 rose 的员工的工资（salary）更改为 8500。

2）将员工表中 id 为 1003 的员工的工资（salary）更改为 6500，职位改为“ANALYST”。

- **方案**

UPDATE 语句用来更新表中的记录，其语法如下：

```
UPDATE table name
SET column = value [, column = value]…
[WHERE condition];
```

其中 WHERE 子句是可选的，但是如果没有 WHERE 子句，则全表的数据都会被更新，务必小心。

- **步骤**

步骤一：更改员工 rose 的薪水为 8500

更改员工 rose 的薪水为 8500，SQL 语句如下所示：

```
UPDATE myemp SET salary = 8500 WHERE name = 'rose';
```

步骤二：更改员工表中 id 为 1003 的员工的工资和职位

将员工表中 id 为 1003 的员工的工资（salary）更改为 6500，职位改为“ANALYST”，SQL 语句如下所示：

```
UPDATE myemp SET salary = 6500, job = 'ANALYST'
WHERE id = 1003;
```

步骤三：更新后，检查数据变化情况

使用如下 SQL 语句检查数据变化情况：

```
SELECT * FROM myemp;
```

- **完整代码**

本案例的完整代码请参考步骤。

13. 删除员工数据

- **问题**

本案例的详细要求如下：

1）删除员工表中职位（job）为空的员工记录；

2）删除员工表中名字（name）为“rose”的员工。

- **方案**

DELETE 语句用来删除表中的记录，其语法如下：

```
DELETE [FROM] table_name [WHERE condition];
```

和 UPDATE 语句一样，WHERE 子句是可选的，但是如果没有 WHERE 子句，则全表的数据都会被删除，务必确认后再予以删除。

- **步骤**

步骤一：删除职位（job）为空的员工记录

删除职位（job）为空的员工记录，SQL 语句如下所示：

```
DELETE FROM myemp WHERE job is null;
```

步骤二：删除员工表中名字（name）为“ROSE”的员工

删除员工表中名字（name）为“ROSE”的员工，SQL 语句如下所示：

```
DELETE FROM myemp WHERE name ='ROSE';
```

步骤三：删除后，检查数据变化情况

检查员工表中名字为“ROSE”的员工是否存在，SQL 语句如下所示：

```
SELECT * FROM myemp WHERE name ='ROSE';
```

- **完整代码**

本案例的完整代码请参考步骤。

课后作业

1. 什么是关系型数据库管理系统

2. 下列关于表的结构说法正确的是

A．表是二维的，由行和列组成。

B．表的行(Row)是横排数据，也被称作字段(Field)。

C．表的列(Column)是纵列数据，也被称作记录(Record)。

D．表是关系数据库的基本存储结构，一个关系数据库由多个数据表(Table)组成。

3. 简述几种主流的数据库及其厂商

4. 简述结构化查询语言的分类

5. 关于 number 类型，下列说法正确的是

A．在 Oracle 中，number 表示数字类型。

B．“salary NUMBER(6,2)“表示 salary 列中的数据，整数位最大为 4 位，小数位最大位数是 2 位，即最大取值：9999.99。

C．“id NUMBER(4)“表示 id 列中的数据，整数位最大为 4 位，没有小数。

D．“fee NUMBER(7,2)“表示 fee 列中的数据，整数最大位数为 7 位，小数为 2 位。

6. 关于 char 和 varchar2，下列说法正确的是

A．char 和 varchar2 都表示字符类型，无区别。

B．char 表示固定长度的字符类型，而 varchar2 表示变长的字符类型。

C．“job VARCHAR(100)” 表示 job 列中最多可存储长度为 100 个字节的字符串。根据其中保存的数据长度，占用的空间是变化的，最大占用空间为 100 个字节。

D．“ename CHAR(20)” 表示 ename 列中最多可存储 20 个字节的字符串，并且占用的空间是固定的 20 个字节。

7. 已知表如下，下列 insert 语句正确的是：

已知创建 student 表的 SQL 语句如下：

```
CREATE TABLE student(
 id NUMBER(4),
 name VARCHAR2(20),
```

```
    gender CHAR(1),
    birth DATE
    );
```

下列对 student 表实施插入的 SQL 语句正确的是：( )。

A．insert into student(id,name,birth)
values(1001,'smith','1989-01-12');

B．英文环境可以采用如下插入语句实施插入：
insert into student(id,name,birth)
values(1001,'smith','12-JAN-89');

C．中文环境可以采用如下插入语句实施插入：
insert into student(id,name,birth)
values(1001,'smith','12-1月-89');

D．insert into student(id,name,birth)
values(1001,'smith',to_date('1989-01-12','yyyy-mm-dd'));

8. 创建账务账户表（Account）

账务账户表的信息如表-1 所示：

表-1 帐务账号表信息

字段名称	类型	备注	字段描述
ID	NUMBER(9)	NOT NULL	帐务帐号 ID
RECOMMENDER_ID	NUMBER(9)	NULL	推荐人帐务帐号 ID
LOGIN_NAME	VARCHAR2(30)	NOT NULL	登录 NetCTOSS 系统的名称，用于用户自服务
LOGIN_PASSWD	VARCHAR2(8)	NOT NULL	登录 NetCTOSS 的口令
STATUS	CHAR（1）	NOT NULL	0：开通；1：暂停；2：删除
CREATE_DATE	DATE	DEFAULT SYSDATE	创建日期
PAUSE_DATE	DATE	NULL	暂停日期（开通状态时为空）
CLOSE_DATE	DATE	NULL	删除日期
REAL_NAME	VARCHAR2(20)	NOT NULL	客户姓名
IDCARD_NO	CHAR(18)	NOT NULL	身份证号码
BIRTHDATE	DATE	NULL	出生日期
GENDER	CHAR(1)	NOT NULL	性别　　0:男 1:女
OCCUPATION	VARCHAR2(50)	NULL	职业
TELEPHONE	VARCHAR2(15)	NOT NULL	联系电话（座机或手机）
EMAIL	VARCHAR2(50)	NULL	电子邮箱
MAILADDRESS	VARCHAR2(50)	NULL	邮箱地址
ZIPCODE	CHAR(6)	NULL	邮编
QQ	VARCHAR2(15)	NULL	QQ
LAST_LOGIN_TIME	Date	NULL	最后一次登录时间
LAST_LOGIN_IP	VARCHAR2(15)	NULL	最后一次登录 IP 地址

请根据上述表信息创建账务账户表 Account。

9. 修改账务账户表（Account）

修改账务账户表，详细要求如下：

1）修改 account 表的表名为 t_account；

2）向 t_account 表增加一列 bak，其数据类型为 varchar2，长度为 50；

3）修改 t_account 表中 bak 列的长度为 40，并增加默认值的设置，默认值为"login"；

4）删除 t_account 表中的 bak 列。

10. 插入账务账户数据

向 t_account 表中插入一条记录，该条记录的数据如表-2 所示：

表- 2

ID	LOGIN_NAME	LOGIN_PASSWD	CREATE_DATE	REAL_NAME	IDCARD_NO	TELEPHONE
1	shiyl	256528	2008-1-28	shiyuanli	410381194302256523	13669351234

11. 更改账务账户数据

将账务账户表 ID 为 1 的账务账户的密码更改为 801206。

12. 删除账务账户数据

删除账务账户表中 ID 为 1 的账务账户信息。

13. 创建示例表 emp 和 dept

创建职员表 emp，表结构如表 - 3 所示：

表 - 3 职员表 emp 信息

字段名	类型	描述
empno	NUMBER(4,0)	员工 ID
ename	VARCHAR2(10)	员工姓名
job	VARCHAR2(9)	职位
mgr	NUMBER(4,0)	员工管理者的 ID
hiredate	DATE	入职日期
sal	NUMBER(7,2)	薪资

comm	NUMBER(7,2)	绩效
deptno	NUMBER(2,0)	员工所在的部门 ID

创建部门表 dept，表结构如表 - 4 所示：

表 - 4 部门表 dept 信息

字段名	类型	描述
deptno	NUMBER(2,0)	部门 ID
dname	VARCHAR2(14 BYTE)	部门名称
loc	VARCHAR2(13 BYTE)	部门所在地

14. 为示例表 emp 和 dept 插入示例数据

为职员表 emp 插入示例数据，示例数据如图 - 1 所示：

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	1980/12/17	800.00		20
7499	ALLEN	SALESMAN	7698	1981/2/20	1600.00	300.00	30
7521	WARD	SALESMAN	7698	1981/2/22	1250.00	500.00	30
7566	JONES	MANAGER	7839	1981/4/2	2975.00		20
7654	MARTIN	SALESMAN	7698	1981/9/28	1250.00	1400.00	30
7698	BLAKE	MANAGER	7839	1981/5/1	2850.00		30
7782	CLARK	MANAGER	7839	1981/6/9	2450.00		10
7788	SCOTT	ANALYST	7566	1987/4/19	3000.00		20
7839	KING	PRESIDENT		1981/11/17	5000.00		10
7844	TURNER	SALESMAN	7698	1981/9/8	1500.00	0.00	30
7876	ADAMS	CLERK	7788	1987/5/23	1100.00		20
7900	JAMES	CLERK	7698	1981/12/3	950.00		30
7902	FORD	ANALYST	7566	1981/12/3	3000.00		20
7934	MILLER	CLERK	7782	1982/1/23	1300.00		10

图 - 1

为部门表 dept 插入示例数据，示例数据如图 - 2 所示：

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

图 - 2

Oracle 数据库基础

Unit02

知识体系.....Page 51

Oracle 字符串操作	字符串类型	CHAR 和 VARCHAR2 类型
		CHAR 和 VARCHAR2 的存储编码
		CHAR 和 VARCHAR2 的最大长度
		LONG 和 CLOB 类型
	字符串函数	CONCAT 和 “||”
		LENGTH
		UPPER、LOWER 和 INITCAP
		TRIM、LTRIM、RTRIM
		LPAD、RPAD
		SUBSTR
		INSTR
Oracle 数值操作	数值类型	NUMBER（P）表示整数
		NUMBER（P, S）表示浮点数
	数值函数	ROUND
		TRUNC
		MOD
		CEIL 和 FLOOR
Oracle 日期操作	日期类型	DATE
		TIMESTAMP
	日期关键字	SYSDATE
		SYSTIMESTAMP
	日期转换函数	TO_DATE
		TO_CHAR
	日期常用函数	LAST_DAY
		ADD_MONTHS
		MONTHS_BETWEEN
		NEXT_DAY
		LEAST、GREATEST
		EXTRACT

达内IT培训集团

空值操作	NULL 的含义	NULL 的含义
	NULL 的操作	插入 NULL 值
		更新成 NULL 值
		NULL 条件查询
		非空约束
	空值函数	NVL
		NVL2

经典案例.....Page 64

Oracle 字符串函数综合示例	CONCAT 和 “||”
	LENGTH
	UPPER、LOWER 和 INITCAP
	TRIM、LTRIM、RTRIM
	LPAD、RPAD
	SUBSTR
Oracle 数值函数综合示例	NUMBER（P）表示整数
	NUMBER（P, S）表示浮点数
	ROUND
	TRUNC
	MOD
	CEIL 和 FLOOR
Oracle 日期转换综合示例	DATE
	TIMESTAMP
	SYSDATE
	SYSTIMESTAMP
	TO_DATE
	TO_CHAR
Oracle 常用日期函数综合示例	LAST_DAY
	ADD_MONTHS
	MONTHS_BETWEEN
	NEXT_DAY
	LEAST、GREATEST
	EXTRACT
Oracle 空值函数综合示例	插入 NULL 值
	更新成 NULL 值
	NULL 条件查询
	非空约束
	NVL
	NVL2

课后作业.....Page 70

1. Oracle 字符串操作

1.1. 字符串类型

1.1.1. 【字符串类型】CHAR 和 VARCHAR2 类型

1.1.2. 【字符串类型】CHAR 和 VARCHAR2 的存储编码

1.1.3. 【字符串类型】CHAR 和 VARCHAR2 的最大长度

1.1.4. 【字符串类型】LONG 和 CLOB 类型

LONG和CLOB类型

- LONG: VARCHAR2加长版，存储变长字符串，最多达2GB的字符串数据
- LONG有诸多限制：每个表只能有一个LONG类型列；不能作为主键；不能建立索引；不能出现在查询条件中...
- CLOB：存储定长或变长字符串，最多达4GB的字符串数据
- ORACLE建议开发中使用CLOB替代LONG类型

```
CREATE TABLE student(
id NUMBER(4),
name CHAR(20),
detail CLOB);
```

1.2. 字符串函数

1.2.1. 【字符串函数】CONCAT 和 “||”

CONCAT和 “||”

- CONCAT(char1, char2)
  返回两个字符串连接后的结果，两个参数char1、char2是要连接的两个字符串。
- 等价操作：连接操作符” ||”
- 如果char1和char2任何一个为NULL，相当于连接了一个空格

```
SELECT CONCAT(CONCAT(ename, ' : '), sal)
FROM emp;
--多个字串连接，用||更直观
SELECT ename || ' : ' || sal FROM emp;
```

1.2.2. 【字符串函数】LENGTH

LENGTH

- LENGTH(char)
  用于返回字符串的长度
- 如果字符类型是VARCHAR2，返回字符的实际长度，如果字符类型是CHAR，长度还要包括后补的空格

```
SELECT ename, LENGTH(ename) FROM emp;
```

1.2.3. 【字符串函数】UPPER、LOWER 和 INITCAP

UPPER、LOWER和INITCAP

- 大小写转换函数，用来转换字符的大小写
- UPPER(char)用于将字符转换为大写形式
- LOWER(char)用于将字符转换为小写形式
- INITCAP(char)用于将字符串中每个单词的首字符大写，其它字符小写，单词之间用空格和非字母字符分隔
- 如果输入的参数是NULL值，仍然返回NULL值

```
SELECT UPPER('hello world'), LOWER('HELLO
WORLD'), INITCAP('hello world')
FROM DUAL;
```

1.2.4. 【字符串函数】TRIM、LTRIM、RTRIM

TRIM、LTRIM、RTRIM

- 作用：截去子串
- 语法形式：
  - TRIM(c2 FROM c1) 从c1的前后截去c2
  - LTRIM(c1[, c2]) 从c1的左边（Left）截去c2
  - RTRIM(c1[, c2]) 从c1的右边（Right）截去c2

  如果没有c2，就去除空格
- TRIM经常用来去掉字符串前后的空格

```
SELECT TRIM('e' from 'elite') AS "t1",
LTRIM('elite', 'e') AS "t2",
RTRIM('elite', 'e') AS "t3" FROM DUAL;
```

1.2.5. 【字符串函数】LPAD、RPAD

LPAD、RPAD

- 补位函数，用于在字符串char1的左端或右端用char2补足到n位， char2可重复多次
  - LPAD(char1, n, char2) 左补位函数
  - RPAD(char1, n, char2) 右补位函数
- --在emp表中使用左补位，将sal用$补齐6位

```
SELECT ename, LPAD(sal, 6, '$') as "salary" FROM
emp ;
```

1.2.6. 【字符串函数】SUBSTR

SUBSTR

- SUBSTR(char, [m[, n]])
  用于获取字符串的子串，返回char中从m位开始取n个字符
- 如果m = 0，则从首字符开始，如果m取负数，则从尾部开始
- 如果没有设置n，或者n的长度超过了char的长度，则取到字符串末尾为止

```
SELECT
SUBSTR('Doctor Who travels in TARDIS', 8, 25)
FROM DUAL;
```

- 字符串的首位计数从1开始

1.2.7. 【字符串函数】INSTR

INSTR

- INSTR(char1, char2[, n [, m]])：
  返回子串char2在源字符串char1中的位置
- 参数：
  - 从n的位置开始搜索，没有指定n，从第1个字符开始搜索
  - m用于指定子串的第m次出现次数，如果不指定取值1
  - 如果在char1中没有找到子串char2 ，返回0

```
SELECT INSTR('Doctor Who', 'Who') as "words"
FROM DUAL;
--8
```

2. Oracle 数值操作

2.1. 数值类型

2.1.1. 【数值类型】NUMBER（P）表示整数

NUMBER（P）表示整数

- 完整语法：NUMBER(precision ，scale)
  - 如果没有设置scale，则默认取值0，即NUMBER(p)表示整数
  - P表示数字的总位数，取值为1-38
- 用来在表中存放如编码、年龄、次数等用整数记录的数据

```
CREATE TABLE student (
id NUMBER(4),
name CHAR(20));
```

2.1.2. 【数值类型】NUMBER（P, S）表示浮点数

NUMBER（P, S）表示浮点数

解说识知

- NUMBER(precision ，scale)
  - precision：NUMBER可以存储的最大数字长度（不包括左右两边的0）
  - scale：在小数点右边的最大数字长度（包括左侧0）
- 指定了s但是没有指定p，则p默认为38，如：列名 number（*，s）
- 经常用来做表中存放金额、成绩等有小数位的数据

```
CREATE TABLE student (
id NUMBER(4),
name CHAR(20),
score NUMBER(5, 2));
```

NUMBER（P, S）表示浮点数（续1）

解说识知

- NUMBER的变种数据类型：内部实现是NUMBER，可以将其理解为NUMBER的别名，目的是多种数据库及编程语言兼容
  - NUMERIC(p,s)：完全映射至NUMBER(p,s)
  - DECIMAL(p,s)或DEC(p,s)：完全映射至NUMBER(p,s)
  - INTEGER或INT：完全映射至NUMBER(38)类型
  - SMALLINT：完全映射至NUMBER(38)类型
  - FLOAT(b)：映射至NUMBER类型
  - DOUBLE PRECISION：映射至NUMBER类型
  - REAL：映射至NUMBER类型

2.2. 数值函数

2.2.1. 【数值函数】ROUND

ROUND

解说识知

- ROUND(n[, m])：用于四舍五入
  - 参数中的n可以是任何数字，指要被处理的数字
  - m必须是整数
  - m取正数则四舍五入到小数点后第m位
  - m取0值则四舍五入到整数位
  - m取负数，则四舍五入到小数点前m位
  - m缺省，默认值是0

```
SELECT ROUND(45.678, 2) FROM DUAL; --45.68
SELECT ROUND(45.678, 0) FROM DUAL;--46
SELECT ROUND(45.678, -1) FROM DUAL;--50
```

2.2.2. 【数值函数】TRUNC

TRUNC

- TRUNC(n[, m])：用于截取
  - n和m的定义和ROUND(n[, m])相同，不同的是功能上按照截取的方式处理数字n

解释说明

```
SELECT TRUNC(45.678, 2) FROM DUAL; --45.67
SELECT TRUNC(45.678, 0) FROM DUAL;--45
SELECT TRUNC(45.678, -1) FROM DUAL;--40
```

2.2.3. 【数值函数】MOD

MOD

- MOD(m, n)：返回m除以n后的余数
  - n为0则直接返回m

解释说明

```
--薪水值按1000取余数
SELECT ename, sal, MOD(sal, 1000) FROM emp;
```

2.2.4. 【数值函数】CEIL 和 FLOOR

CEIL和FLOOR

- CEIL(n)、FLOOR(n)这两个函数顾名思义，一个是天花板，就是取大于或等于n的最小整数值，一个是地板，就是取小于或等于n的最大整数值
- 比如数字n = 4.5，那么它的CEIL是5，它的FLOOR是4

解释说明

```
SELECT CEIL(45.678) FROM DUAL; --46
SELECT FLOOR(45.678) FROM DUAL;--45
```

3. Oracle 日期操作

3.1. 日期类型

3.1.1. 【日期类型】DATE

DATE

- ORACLE中最常用的日期类型，用来保存日期和时间
- DATE表示的日期范围可以是公元前4712年1月1日至公元9999年12月31日
- DATE类型在数据库中的存储固定为7个字节，格式为：
  - 第1字节：世纪+100
  - 第2字节：年
  - 第3字节：月
  - 第4字节：天
  - 第5字节：小时+1
  - 第6字节：分+1
  - 第7字节：秒+1

3.1.2. 【日期类型】TIMESTAMP

TIMESTAMP

- ORACLE常用的日期类型
- 与DATE的区别是不仅可以保存日期和时间，还能保存小数秒，最高精度可以到ns(纳秒)
- 数据库内部用7或者11个字节存储，精度为0，用7字节存储，与DATE功能相同，精度大于0则用11字节存储
- 格式为：
  - 第1字节-第7字节：和DATE相同
  - 第8-11字节：纳秒，采用4个字节存储，内部运算类型为整型

```
CREATE TABLE test(
    c1 DATE,
    c2 TIMESTAMP);
```

3.2. 日期关键字

3.2.1. 【日期关键字】SYSDATE

SYSDATE

- 其本质是一个Oracle的内部函数，返回当前的系统时间，精确到秒
- 默认显示格式是DD-MON-RR

```
SELECT SYSDATE FROM DUAL;

--学生注册时间即数据记录插入的时间
  CREATE TABLE student (id NUMBER(4),
  name CHAR(20),
  registerDate DATE DEFAULT SYSDATE);
```

3.2.2. 【日期关键字】SYSTIMESTAMP

SYSTIMESTAMP

- 内部函数，返回当前系统日期和时间，精确到毫秒

解说识知

```
SELECT SYSTIMESTAMP FROM DUAL;
SELECT TO_CHAR(SYSTIMESTAMP,'SSSS.FF')
  FROM DUAL;
```

3.3. 日期转换函数

3.3.1. 【日期转换函数】TO_DATE

TO_DATE

- TO_DATE(char[, fmt[, nlsparams]])：将字符串按照定制格式转换为日期类型
  - char：要转换的字符串
  - fmt：格式
  - nlsparams：指定日期语言
  - 常用的日期格式见右表

解说识知

查询2002年以后入职的员工：

```
SELECT ename, hiredate
FROM emp
WHERE hiredate >
TO_DATE('2002-01-01',
'YYYY-MM-DD');
```

TO_DATE（续1）

解说识知

YY	2位数字的年份
YYYY	4为数字的年份
MM	2位数字的月份
MON	简拼的月份
MONTH	全拼的月份
DD	2位数字的天
DY	周几的缩写
DAY	周几的全拼
HH24	24小时制的小时
HH12	12小时制的小时
MI	显示分钟
SS	显示秒

3.3.2. 【日期转换函数】TO_CHAR

TO_CHAR

- 将其它类型的数据转换为字符类型
- TO_CHAR(date[, fmt[, nlsparams]])：将日期类型数据date按照fmt的格式输出字符串。nlsparams用于指定日期语言

解说识知

```
SELECT ename,
TO_CHAR(hiredate, 'YYYY"年"MM"月"DD"日"')
FROM emp;
```

3.4. 日期常用函数

3.4.1. 【日期常用函数】LAST_DAY

LAST_DAY

- LAST_DAY(date)：返回日期date所在月的最后一天
- 在按照自然月计算某些业务逻辑，或者安排月末周期性活动时很有用处

解说识知

```
SELECT LAST_DAY(SYSDATE) FROM DUAL;
SELECT LAST_DAY('20-2月-09') FROM DUAL;
```

3.4.2. 【日期常用函数】ADD_MONTHS

ADD_MONTHS

- ADD_MONTHS(date, i)：返回日期date加上i个月后的日期值
  - 参数i可以是任何数字，大部分时候取正值整数
  - 如果i是小数，将会被截取整数后再参与运算
  - 如果i是负数，则获得的是减去i个月后的日期值

解说识知

```
--计算职员入职20周年纪念日
SELECT ename,
ADD_MONTHS(hiredate, 20 * 12) as "20周年"
FROM emp;
```

3.4.3. 【日期常用函数】MONTHS_BETWEEN

MONTHS_BETWEEN

- MONTH_BETWEEN(date1, date2)：计算date1和date2两个日期值之间间隔了多少个月
- 实际运算是date1-date2，如果date2时间比date1晚，会得到负值
- 除非两个日期间隔是整数月，否则会得到带小数位的结果，比如计算2009年9月1日到2009年10月10日之间间隔多少个月，会得到1.29个月

```
--计算职员入职多少个月
SELECT ename,
MONTHS_BETWEEN(SYSDATE, hiredate) as "hiredate"
FROM emp ;
```

3.4.4. 【日期常用函数】NEXT_DAY

NEXT_DAY

- NEXT_DAY(date, char)：返回date日期数据的**下一个周几**，周几是由参数char来决定的
- 在中文环境下，直接使用"星期三"这种形式，英文环境下，需要使用"WEDNESDAY"这种英文的周几。为避免麻烦，可以直接用数字1-7表示周日-周六
- NEXT_DAY不是明天！

```
--查询下个周三是几号
SELECT NEXT_DAY(SYSDATE, 4) as "NEXT_WEDN"
  FROM DUAL;
```

3.4.5. 【日期常用函数】LEAST、GREATEST

LEAST、GREATEST

- GREATEST(expr1[, expr2[, expr3]]...)
- LEAST(expr1[, expr2[, expr3]]...)
- 也被称作比较函数，可以有多个参数值，返回结果是参数列表中最大或最小的值
- 参数类型必须一致
- 在比较之前，在参数列表中第二个以后的参数会被隐含的转换为第一个参数的数据类型，所以如果可以转换，则继续比较，如果不能转换将会报错

```
SELECT LEAST(SYSDATE, '10-10月 -08') FROM DUAL;
```

3.4.6. 【日期常用函数】EXTRACT

EXTRACT

- EXTRACT(date FROM datetime)：从参数datetime中提取参数date指定的数据，比如提取年、月、日

解说内容

```
SELECT EXTRACT(YEAR FROM SYSDATE)
  CURRENT_YEAR
FROM DUAL;

SELECT
EXTRACT(HOUR FROM TIMESTAMP '2008-10-10
  10:10:10')
FROM DUAL;
```

4. 空值操作

4.1. NULL 的含义

4.1.1. 【NULL 的含义】NULL 的含义

NULL的含义

- 数据库里的重要概念：NULL，即空值
- 有时表中的某些字段值，数据未知或暂时不存在，取值NULL
- 任何数据类型均可取值NULL

解说内容

4.2. NULL 的操作

4.2.1. 【NULL 的操作】插入 NULL 值

插入 NULL值

- CREATE TABLE student
  (id NUMBER(4), name CHAR(20), gender CHAR(1));
- INSERT INTO student VALUES(1000, '李莫愁', 'F');
- INSERT INTO student VALUES(1001, '林平之', NULL);
- INSERT INTO student(id, name) VALUES(1002, '张无忌');

显式插入NULL值

隐式插入NULL值

解说内容

4.2.2. 【NULL 的操作】更新成 NULL 值

更新成NULL值

- UPDATE student SET gender = NULL;
- 注意这种更新只有在此列没有非空约束的情况下才可操作
- 如果某列有非空约束，则无法更新为NULL值，上述语句会报错

解说识知

4.2.3. 【NULL 的操作】NULL 条件查询

NULL条件查询

- NULL不等于任何值
- SELECT * FROM student WHERE gender IS NULL;

注意是IS，不是=号

解说识知

4.2.4. 【NULL 的操作】非空约束

非空约束

- 非空(NOT NULL)约束用于确保字段值不为空
- 默认情况下，任何列都允许有空值，但系统的业务逻辑可能会要求某些列不能取空值
- 当某个字段被设置了非空约束条件，这个字段中必须存在有效值。即：当执行插入数据的操作时，必须提供这个列的数据，当执行更新操作时，不能给这个列的值设置为NULL

```
CREATE TABLE student
    (id NUMBER(4),
     name CHAR(20),
    gender CHAR(1) NOT NULL);
```

解说识知

4.3. 空值函数

4.3.1. 【空值函数】NVL

NVL

- NVL(expr1, expr2)：将NULL转变为非NULL值
  - 如果expr1为NULL，则取值expr2, expr2是实际值
  - expr1和expr2可以是任何数据类型，但两个参数的数据类型必须是一致的

解说知识

```
--计算员工月收入
SELECT ename, sal, comm,
       sal + nvl(comm, 0) as "salary"
FROM emp;
```

4.3.2. 【空值函数】NVL2

NVL2

- NVL2(expr1, expr2, expr3)：和NVL函数功能类似，都是将NULL转变为实际值
- NVL2用来判断expr1是否为NULL，如果不是NULL，返回expr2，如果是NULL，返回expr3。

解说知识

```
SELECT ename, sal, comm,
nvl2(comm, sal + comm, sal) as "salary"
FROM emp;
```

经典案例

1. Oracle 字符串函数综合示例

- **问题**

本案例的详细要求如下：

1） 职员表 emp 中有姓名（ename）为 smith 的员工记录，但不能确定其名字的大小写格式。下列 SQL 语句中能够查询出名字为 smith 的员工信息的是：( )。

A . SELECT ename,job, hiredate FROM emp WHERE ename= 'smith';

B . SELECT ename,job, hiredate FROM emp WHERE upper (ename)= 'smith';

C . SELECT ename,job, hiredate FROM emp WHERE ename=upper('smith');

D . SELECT ename,job, hiredate FROM emp WHERE lower(ename)= 'smith';

2）写 SQL 语句查找员工名字的长度为 5 个字符的员工信息。

3）查询员工的姓名和工资，按下面的形式显示：

```
   ENAME           SALARY
-----------------------------------------------------
   SMITH           $$$$$$$$$$$$800
   ALLEN           $$$$$$$$$$$$1600
   WARD            $$$$$$$$$$$$1250
```

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：查询名字为 smith 的员工信息

职员表中有名字（ename）为 smith 的员工记录，但不能确定其名字的大小写格式，因此使用 lower 或 upper 函数将数据库中的名字信息都转成小写或大写，与“smith”或“SMITH”使用“=”等号进行比较，即可查出该员工的信息，SQL 语句如下所示：

```
SELECT ename,job,hiredate FROM emp WHERE LOWER(ename)= 'smith';
```

或

```
SELECT ename,job,hiredate FROM emp WHERE UPPER(ename)= 'SMITH';
```

因此，在此案例中，正确的答案为 D。

步骤二：查找员工名字的长度为 5 个字符的员工信息

要查找员工名字的长度为 5 个字符的员工信息，可以使用 length 方法计算员工名字的长度，然后与 5 使用“=”进行比较，SQL 语句如下所示：

```
SELECT ename, sal, job FROM emp WHERE LENGTH(ename) = 5;
```

步骤三：工资列不足 15 使用“$”左补足 15 位

使用 LPAD 方法实现对工资列的左补位，SQL 语句如下所示：

```
SELECT ename, LPAD(sal, 15, '$') as "SALARY" FROM emp ;
```

2. Oracle 数值函数综合示例

- **问题**

查询各员工的名字 ename，并显示出各员工在公司工作的工作天数，用整数表示。提示：将入职日期与当前日期比较，得到该员工已经工作的天数。显示形式如下：

```
ENAME      HIREDATE         WORKTIME
-----------------------------------------------------
SMITH      17-12月-80        12193
ALLEN      20-2月 -81        12128
WARD       22-2月 -81        12126
JONES      02-4月 -81        12087
MARTIN     28-9月 -81        11908
BLAKE      01-5月 -81        12058
```

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：计算各员工在公司工作的天数

首先，使用当前日期时间 SYSDATE 减去入职时间列 hiredate，然后，在该差值的基础上进行四舍五入来计算员工在公司工作的天数，SQL 语句如下所示：

```
SELECT ename, hiredate,
ROUND(SYSDATE-hiredate) as "WORKTIME"
FROM emp;
```

3. Oracle 日期转换综合示例

- **问题**

本案例的详细要求如下：

1）按照”2009-4-11 20:35:10 ”格式显示系统时间。

2）现需要显示职员的入职时间格式为“17 of November 2004”，下列 SQL 语句正确的是（ ）。

A．

```
SELECT hiredate('fmDD "of" MONTH YYYY') "Date Hired" FROM emp;
```

B.

```
SELECT hiredate('DD "of" MONTH YYYY') "Date Hired" FROM emp;
```

C.

```
SELECT TO CHAR (hiredate,'DDspth of MONTH YYYY') "Date Hired"
FROM emp;
```

D.

```
SELECT TO_CHAR(hiredate,'fmDD "of" MONTH YYYY') " Date Hired"
FROM emp;
```

3）构造 SQL 语句，产生类似于下面形式的结果：

```
   ENAME           HIREDATE                REVIEW
-------------------------------------------------------------------
   SMITH          1980-12-17             1980 年 12 月 17 日
```

即 hiredate 列显示的格式为“1980-12-17”；再将 hiredate 列以“1980 年 12 月 17 日”格式显示，并且显示的列名为“REVIEW”。

- **步骤**

步骤一：按格式显示系统时间

使用 SYSDATE 表示返回当前的系统时间，精确到秒；使用“YYYY”表示 4 位数字的年份；使用 MM 表示 2 位数字的月份；使用 DD 表示 2 位数字的天；使用 HH24 表示 24 小时制的小时；使用 MI 表示分钟；使用 SS 表示秒。SQL 语句如下所示：

```
SELECT TO_CHAR(SYSDATE, 'yyyy-mm-dd hh24:mi:ss' ) FROM DUAL;
```

步骤二：按格式显示员工的入职时间

显示职员的入职时间格式为“17 of November 2004”，按 Oracle 的日期格式如下：

```
'fmDD "of" MONTH YYYY'
```

其中，fm 表示格式；DD 表示 2 位数字的天；“of”即为显示字符串“of”；MONTH 表示全拼的月份；YYYY 表示 4 位数字的年份。

然后，使用 to_char 函数，将日期类型的表示的时间，按格式转换为字符类型的时间，SQL 语句如下所示：

```
SELECT TO_CHAR(hiredate,'fmDD "of" MONTH YYYY') "Date Hired"
FROM emp;
```

上述 SQL 语句中，"Date Hired"是“TO_CHAR(hiredate,'fmDD "of" MONTH YYYY')”的别名，在 SQL 中将别名加上双引号，可以区分别名中字符的大小写。不加双引号别名默

认显示为大写。

因此，本案例中 D 选项正确。

步骤三：按格式显示员工的入职时间

使用 to_char 函数配合 Oracle 日期格式将 hiredate 列显示的格式为“1980-12-17”，并且将 hiredate 列以“1980 年 12 月 17 日”格式显示，并且显示的列名为“REVIEW”，SQL 语句如下所示：

```
SELECT  ename, TO CHAR(hiredate, 'yyyy-mm-dd') as "HIREDATE",
TO_CHAR(hiredate, 'yyyy"年"mm"月"dd')  as "REVIEW"
FROM emp;
```

4. Oracle 常用日期函数综合示例

- **问题**

查询一个订单，从下单开始到启运需要多长时间，以月为单位（例如：3 个月，6 个月），写出正确的查询语句。假设订单表名为 ord，其中订单编码、下单时间、启运时间的列名分别为：custid、orderdate、shipdate。

- **方案**

使用 MONTHS_BETWEEN 计算下单时间（orderdate）与启运时间（shipdate）这两个日期值之间间隔的月份数，然后，将该间隔的月份数使用 ROUND 方法进行四舍五入，SQL 语句如下所示：

```
SELECT custid, orderdate, shipdate,
ROUND(MONTHS_BETWEEN(shipdate, orderdate)) as "Time Taken" FROM ord;
```

- **完整代码**

本案例可按如下 SQL 语句进行测试。

创建 ord 表的 SQL 语句如下所示：

```
CREATE TABLE ord(
  custid NUMBER(4),
  orderdate DATE,
  shipdate DATE
);
```

向 ord 表中插入测试数据的 SQL 语句如下所示：

```
INSERT INTO ord(custid,orderdate,shipdate)values(1001,'12-4 月-98','10-8 月
-98');
INSERT INTO ord(custid,orderdate,shipdate)values(1002,'12-3 月-98','10-6 月
-98');
```

```
INSERT INTO ord(custid,orderdate,shipdate)values(1003,'12-2月-98','10-9月
-98');
```

查询一个订单，从下单开始到启运需要多长时间，SQL 语句如下所示：

```
SELECT custid, orderdate, shipdate,
ROUND(MONTHS_BETWEEN(shipdate, orderdate)) as "Time Taken" FROM ord;
```

5. Oracle 空值函数综合示例

- **问题**

现有数据表 Customer，其结构如表-1 所示：

表-1 customer 表结构

字段名	类型	描述
cust_id	NUMBER(4)	客户编码
cname	VARCHAR2(25)	客户姓名
birthday	DATE	客户生日
account	NUMBER	客户账户余额

1）构造 SQL 语句，列出 Customer 数据表中每个客户的信息。如果客户生日未提供，则该列值显示“not available”；如果没有余额信息，则显示“no account”。

2）构造 SQL 语句，列出生日在 1987 年的客户的全部信息。

- **步骤**

步骤一：列出 Customer 数据表中每个客户的信息

使用 TO_CHAR 函数将日期类型的 birthday 列的数据转换为字符类型。然后，使用 NVL 函数，当 birthday 为 NULL 值时，将 birthday 列的数据显示为“not available”。同理，使用 TO_CHAR 函数将数值类型的 account 列数据转换为字符类型，然后，使用 NVL 函数，当 account 为 NULL 值时，将 account 列的数据显示为“no account”，SQL 语句如下所示：

```
SELECT cust_id, cname, NVL(TO_CHAR(birthday, 'yyyy-mm-dd'), 'not available'),
NVL(TO_CHAR(account), 'no account')
FROM customer;
```

上述 SQL 语句中，使用 TO_CHAR 函数将日期类型的数据、数值类型的数据转换为字符类型。这是因为，NVL 函数要求两个参数的数据类型必须保持一致。

步骤二：列出生日在 1987 年的客户的全部信息

使用 TO_CHAR 函数将获取 birthday 列的年份数据，与“1987”使用“=”等号进行比较，SQL 语句如下所示：

```
SELECT * FROM customer
WHERE TO_CHAR(birthday, 'yyyy') = '1987';
```

- **完整代码**

本案例可按如下 SQL 语句进行测试。

创建 customer 表的 SQL 语句如下所示：

```
create table customer(
   cust_id NUMBER(4)  Primary Key,
   cname VARCHAR2(25) Not Null,
   birthday DATE,
   account NUMBER
);
```

向 customer 插入如下测试数据，SQL 语句如下所示：

```
INSERT  INTO  customer(cust_id,cname,birthday,account)values(1001,'郭靖','12-12月-13',200);
INSERT INTO customer(cust_id,cname)values(1002,'黄蓉');
INSERT  INTO  customer(cust_id,cname,birthday,account)values(1003,'梅超风','12-12月-87',200);
```

列出 Customer 数据表中每个客户的信息的 SQL 语句如下：

```
SELECT cust id, cname,
NVL(TO_CHAR(birthday, 'yyyy-mm-dd'), 'not available'), NVL(TO_CHAR(account), 'no account')
FROM customer;
```

列出生日在 1987 年的客户的全部信息的 SQL 语句如下：

```
SELECT * FROM customer
WHERE TO_CHAR(birthday, 'yyyy') = '1987';
```

课后作业

1. 下列 select 语句输出的结果是？

有职员表 emp，表结构如表 - 1 所示：

表 - 1 职员表 emp 信息

字段名	类型	描述
empno	NUMBER(4,0)	员工 ID
ename	VARCHAR2(10)	员工姓名
job	VARCHAR2(9)	职位
mgr	NUMBER(4,0)	员工管理者的 ID
hiredate	DATE	入职日期
sal	NUMBER(7,2)	薪资
comm	NUMBER(7,2)	绩效
deptno	NUMBER(2,0)	员工所在的部门 ID

emp 表中的示例数据如图 - 1 所示：

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	1980/12/17	800.00		20
7499	ALLEN	SALESMAN	7698	1981/2/20	1600.00	300.00	30
7521	WARD	SALESMAN	7698	1981/2/22	1250.00	500.00	30
7566	JONES	MANAGER	7839	1981/4/2	2975.00		20
7654	MARTIN	SALESMAN	7698	1981/9/28	1250.00	1400.00	30
7698	BLAKE	MANAGER	7839	1981/5/1	2850.00		30
7782	CLARK	MANAGER	7839	1981/6/9	2450.00		10
7788	SCOTT	ANALYST	7566	1987/4/19	3000.00		20
7839	KING	PRESIDENT		1981/11/17	5000.00		10
7844	TURNER	SALESMAN	7698	1981/9/8	1500.00	0.00	30
7876	ADAMS	CLERK	7788	1987/5/23	1100.00		20
7900	JAMES	CLERK	7698	1981/12/3	950.00		30
7902	FORD	ANALYST	7566	1981/12/3	3000.00		20
7934	MILLER	CLERK	7782	1982/1/23	1300.00		10

图 - 1

对于 emp 表或者 dual 表，执行如下查询，请描述各查询的结果。

1．select job, length(job) from emp order by job;

2．select ename || ' earns $' || sal ||' monthly but wants $' || sal * 3 as “Dream Salary” from emp;

3．select empno, ename from emp where upper (job)= 'clerk';

4．select SUBSTR('Doctor Who', 1, 6) from dual;

5．select TRIM('半' FROM '半月二更半') AS "上联" from dual;

6．select TRIM (leading '半' FROM '半月二更半') AS "上联" from dual;

7．select TRIM (trailing '半' FROM '半月二更半') AS "上联" from dual;

8．select LPAD('TARDIS', 10, '*') from dual;
9．select LPAD('TARDIS', 3, '*') from dual;
10．select * from emp where length(ename) = 5;

2. 关于类型定义 Number(9,2)，下列说法正确的是

A．整数部分 9 位，小数部分 2 位，共 11 位
B．整数部分 7 位，小数部分 2 位，共 9 位
C．整数部分 6 位，小数点一位，小数部分 2 位，共 9 位
D．整数部分 5 位，小数部分 2 位，共 7 位

3. 下列 select 语句输出的结果是？

对于 emp 表或者 dual 表，执行如下查询，请描述各查询的结果。
1．select ename , sal , round(sal*1.08) from emp;
2．select mod(11,4) from dual;
3．select trunc(123.123,-1)from dual;
4．select floor(sysdate-hiredate) as "入职天数",ename from emp;

4. 简述 date 和 timestamp 的区别

5. 按要求完成插入语句（包含日期类型）

向 Unit01 课后练习中，所创建的 t_account 表中插入一条记录，该记录的数据如表-2 所示：

表- 2 插入数据明细

ID	LOGIN_NAME	LOGIN_PASSWD	CREATE_DATE	REAL_NAME	IDCARD_NO	TELEPHONE
1	shiyl	256528	2014-03-02	shiyuanli	410381194302256523	13669351234

6. 按指定输出格式写出 select 语句（包含日期类型）

1．按照“2009 年 4 月 11 日 20 时 35 分 10 秒”格式显示系统时间。
2．构造查询语句，产生类似于下面形式的结果：

```
NAME            HIREDATE                REVIEW
-----------------------------------------------------------------------
ALLEN           1980 年 12 月 17 日     1980/12/17
```

7. 下列 select 语句输出的结果是？

对于 emp 表或者 dual 表，执行如下查询，请描述各查询的结果。

1．select * from emp where extract(year from hiredate) = '1987';

2．select ename, last_day(hiredate) from emp;

3．select next_day(sysdate, 7) from dual;

4．select ename, add_months(hiredate, 3) from emp;

5．select ename, months_between(sysdate, hiredate) from emp;

6．select greatest(to_date('20070101', 'yyyymmdd'),hiredate) from emp;

8. 已知表结构如下，按要求完成 select 语句

现有数据表 Customer，其结构如表 - 3 所示：

表 - 3 顾客表 Customer 信息

字段名	类型	描述
cust_id	NUMBER(4)	客户编码
cname	VARCHAR2(25)	客户姓名
birthday	DATE	客户生日
account	NUMBER	客户账户余额

构造 SQL 语句，列出 Customer 数据表中生日未提供的客户记录。

9. 已知表结构如下，按要求完成 select 语句

对于 emp 表，构造 SQL 语句，查询员工编码 empno，姓名 ename，以及月收入(薪水 + 奖金)。注意：有的员工暂时没有奖金，即，如果 comm 列为 null，则按照数值 0 计算。

10. 已知表结构如下，按要求完成 select 语句

查询 emp 表，列出员工的编码、姓名和入职时间。

要求：如果入职时间为 null，则显示'not available'；如果入职时间不为 null，则按照 yyyy-mm-dd 格式显示。

Oracle 数据库基础

Unit03

知识体系.....Page 75

SQL（基础查询）	基本查询语句	FROM 子句
		使用别名
		WHERE 子句
		SELECT 子句
	查询条件	使用>,<.>=,<= , != , <>,=
		使用 AND ,OR 关键字
		使用 LIKE 条件(模糊查询)
		使用 IN 和 NOT IN
		BETWEEN...AND...
		使用 IS NULL 和 IS NOT NULL
		使用 ANY 和 ALL 条件
		查询条件中使用表达式和函数
		使用 DISTINCT 过滤重复
	排序	使用 ORDER BY 字句
		ASC 和 DESC
		多个列排序
	聚合函数	什么是聚合函数
		MAX 和 MIN
		AVG 和 SUM
		COUNT
		聚合函数对空值的处理
	分组	GROUP BY 子句
		分组查询
		HAVING 字句
	查询语句执行顺序	查询语句执行顺序
SQL（关联查询）	关联基础	关联的概念
		笛卡尔积
		等值连接
	关联查询	内连接

达内IT培训集团

		外连接
		全外连接
		自连接

经典案例.....Page 88

Oracle 基础查询综合示例	FROM 子句
	使用别名
	WHERE 子句
	SELECT 子句
	使用>,<.>=,<= , != , <>,=
	使用 AND ,OR 关键字
	使用 LIKE 条件(模糊查询)
	使用 IN 和 NOT IN
	BETWEEN...AND...
	使用 IS NULL 和 IS NOT NULL
	使用 ANY 和 ALL 条件
	查询条件中使用表达式和函数
	使用 DISTINCT 过滤重复
	使用 ORDER BY 字句
	ASC 和 DESC
	多个列排序
Oracle 分组查询综合示例	什么是聚合函数
	MAX 和 MIN
	AVG 和 SUM
	COUNT
	聚合函数对空值的处理
	GROUP BY 子句
	分组查询
	HAVING 字句
	查询语句执行顺序
Oracle 关联查询综合示例	关联的概念
	笛卡尔积
	等值连接
	内连接
	外连接
	全外连接
	自连接

课后作业.....Page 100

1. SQL（基础查询）

1.1. 基本查询语句

1.1.1. 【基本查询语句】FROM 子句

FROM子句

- SELECT <*, column [alias], ...> FROM table;
  - SELECT用于指定要查询的列
  - FROM指定要从哪个表中查询

解说内容

- 如果要查询所有列，可以在SELECT后面使用*号
- 如果只查询特定的列，可以直接在SELECT后面指定列名，列名之间用逗号隔开。

```
SELECT * FROM dept;
```

1.1.2. 【基本查询语句】使用别名

使用别名

- 在SQL语句中可以通过使用列的别名改变标题的显示样式，或者表示计算结果的含义
- 使用语法是列的别名跟在列名后，中间可以加或不加一个“AS”关键字
- 如果希望别名中区分大小写字符，或者别名中包含字符或空格，则必须用双引号引起来

解说内容

```
SELECT empno AS id, ename "Name",
  sal * 12 "Annual Salary" FROM emp;
```

1.1.3. 【基本查询语句】WHERE 子句

WHERE子句

- 在SELECT语句中，可以在WHERE子句中使用比较操作符限制查询结果
- 如果和数字比较，可以使用单引号引起，也可以不用
- 如果和字符及日期类型的数据比较，则必须用单引号引起

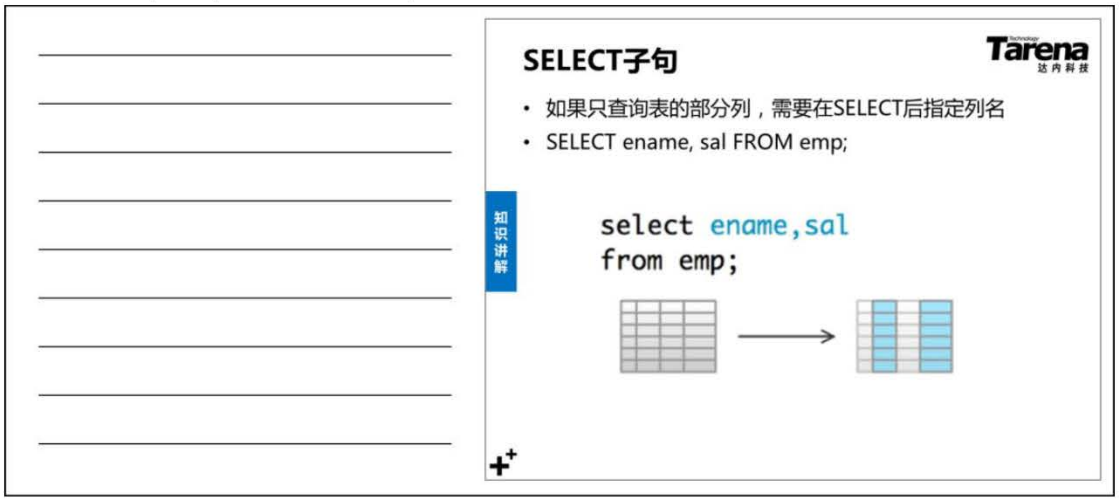
解说内容

```
--查询部门10下的员工信息
SELECT * FROM emp
WHERE deptno = 10;
```

```
select *
from emp
where deptno = 10;
```

```
--查询职员表中职位是’SALESMAN’的职员
SELECT ename, sal, job FROM emp
WHERE job = 'SALESMAN';
```

1.1.4. 【基本查询语句】SELECT 子句



1.2. 查询条件

1.2.1. 【查询条件】使用>,<.>=,<=，!=，<>,=

1.2.2. 【查询条件】使用 AND ,OR 关键字

1.2.3. 【查询条件】使用 LIKE 条件(模糊查询)

使用LIKE条件(模糊查询)

- 比较操作符LIKE用来做模糊查询
- 当用户在执行查询时，不能完全确定某些信息的查询条件，或者只知道信息的一部分，可以借助LIKE来实现
- LIKE需要借助两个通配符：
  - %：表示0到多个字符
  - _：标识单个字符
- 这两个通配符可以配合使用，构造灵活的匹配条件

```
SELECT ename, job FROM emp
  WHERE ename LIKE '_A%';
```

1.2.4. 【查询条件】使用 IN 和 NOT IN

使用IN和NOT IN

- 比较操作符 IN(list) 用来取出符合列表范围中的数据
- List 表示值列表，当列或表达式匹配于列表中的任何一个值时，条件为TRUE，该条记录则被显示出来
- IN 也可以理解为一个范围比较操作符，只不过这个范围是一个指定的值列表
- NOT IN(list) 取出不符合此列表中的数据记录

```
--查询职位是MANAGER或者CLERK的员工
SELECT ename, job FROM emp
WHERE job IN ('MANAGER', 'CLERK');
--查询不是部门10或20的员工
SELECT ename, job FROM emp WHERE deptno NOT IN (10, 20);
```

1.2.5. 【查询条件】BETWEEN...AND…

BETWEEN...AND...

- BETWEEN...AND...操作符用来查询符合某个值域范围条件的数据
- 最常见的是使用在数字类型的数据范围上，但对字符类型和日期类型数据也同样适用

```
--查询薪水在1500-3000之间的职员信息
  SELECT ename, sal FROM emp
  WHERE sal BETWEEN 1500 AND 3000;
```

1.2.6. 【查询条件】使用 IS NULL 和 IS NOT NULL

使用IS NULL和IS NOT NULL

- 空值NULL是一个特殊的值，比较的时候不能使用”=”号，必须使用IS NULL，否则不能得到正确的结果。

解说识知

```
--查询哪些职员的奖金数据为NULL
SELECT ename, sal, comm FROM emp
WHERE comm IS NULL;
```

1.2.7. 【查询条件】使用 ANY 和 ALL 条件

使用ANY和ALL条件

- ALL和ANY不能单独使用，需要配合单行比较操作符>、>=、<、<=一起使用
  - > ANY：大于最小
  - < ANY：小于最大
  - > ALL：大于最大
  - < ALL：小于最小

解说识知

```
SELECT empno, ename, job, sal, deptno
FROM emp
WHERE sal > ANY(3500,4000,4500);
```

1.2.8. 【查询条件】查询条件中使用表达式和函数

查询条件中使用表达式和函数

- 当查询需要对选出的字段进行进一步计算，可以在数字列上使用算术表达式(+、-、*、/)
- 表达式符合四则运算的默认优先级，如果要改变优先级可以使用括号
- 算术运算主要是针对数字类型的数据，对日期类型的数据可以做加减操作，表示在一个日期值上加或减一个天数

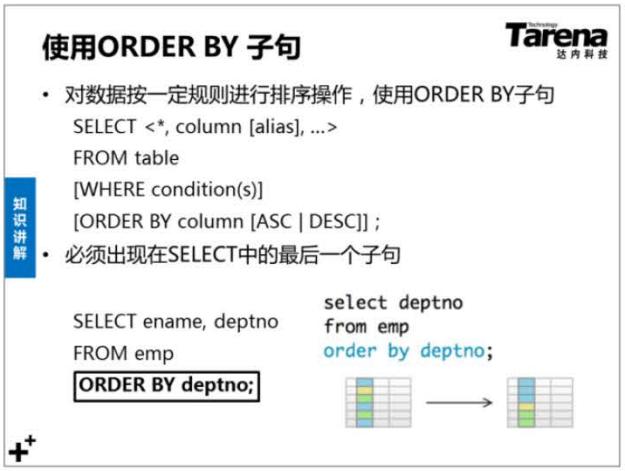
解说识知

```
--查询条件中使用函数
select ename, sal, job from emp where ename = UPPER('rose');
--查询条件中使用表达式
select ename, sal, job from emp where sal * 12 >100000;
```

1.2.9. 【查询条件】使用 DISTINCT 过滤重复

1.3. 排序

1.3.1. 【排序】使用 ORDER BY 字句



1.3.2. 【排序】ASC 和 DESC

1.3.3. 【排序】多个列排序

多个列排序

- 当以多列作为排序标准时，首先按照第一列进行排序，如果第一列数据相同，再以第二列排序，以此类推
- 多列排序时，不管正序还是倒序，每个列需要单独设置排序方式

解说识知

--对职员表中的职员排序，先按照部门编码正序排列，再按照薪水降序排列

```
SELECT ename, deptno, sal FROM emp
ORDER BY deptno ASC, sal DESC;
```

1.4. 聚合函数

1.4.1. 【聚合函数】什么是聚合函数

什么是聚合函数

- 查询时需要做一些数据统计，比如：查询职员表中各部门职员的平均薪水，各部门的员工人数
- 需要统计的数据并不能在职员表里直观列出，而是需要根据现有的数据计算得到结果
- 这种功能可以使用聚合函数来实现，即：将表的全部数据划分为几组数据，每组数据统计出一个结果
- 因为是多行数据参与运算返回一行结果，也称作分组函数、多行函数、集合函数。

解说识知

1.4.2. 【聚合函数】MAX 和 MIN

MAX和MIN

- 用来取得列或表达式的最大、最小值
- 可以用来统计任何数据类型，包括数字、字符和日期

解说识知

--获取机构下的最高薪水和最低薪水，参数是数字

```
SELECT MAX(sal) max_sal, MIN(sal) min_sal
FROM emp;
```

--最早和最晚的入职时间，参数是日期

```
SELECT MAX(hiredate) max_hire, MIN(hiredate)
  min_hire
FROM emp;
```

1.4.3. 【聚合函数】AVG 和 SUM

AVG和SUM

- 用来统计列或表达式的平均值和和值
- 只能操作数字类型
- 忽略NULL值

解释说明

```
--获得机构下全部职员的平均薪水和薪水总和
SELECT AVG(sal) avg_sal,SUM(sal) sum_sal FROM
  emp;
```

1.4.4. 【聚合函数】COUNT

COUNT

- 用来计算表中的记录条数
- 忽略NULL值

解释说明

```
--获取职员表中一共有多少名职员记录
  SELECT COUNT(*) total_num FROM emp;

--获得职员表中有多少人是有职位的（忽略没有职位的员工
记录）
  SELECT COUNT(job) total_job FROM emp;
```

1.4.5. 【聚合函数】聚合函数对空值的处理

聚合函数对空值的处理

- 聚合函数忽略NULL值
- 当emp表中的comm列有NULL值，比如某新入职员工没有绩效，比较两条语句的结果：

解释说明

```
SELECT AVG(comm) avg_sal FROM emp;
SELECT AVG(NVL(comm,0)) avg_sal FROM emp;
```

1.5. 分组

1.5.1. 【分组】GROUP BY 子句

GROUP BY子句

解详识知

```
SELECT <*, column [alias], ...>
FROM table [WHERE condition(s)]
[GROUP BY group_by_expression]
[HAVING group_condition]
[ORDER BY column [ASC | DESC]] ;
```

- 当希望得到每个部门的平均薪水，而不是整个机构的平均薪水
- 把整个数据表按部门划分成一个个小组，每个小组中包含一行或多行数据，在每个小组中再使用分组函数进行计算，每组返回一个结果
- 划分的小组有多少，最终的结果集行数就有多少

1.5.2. 【分组】分组查询

分组查询

解详识知

- 分组查询的功能

deptno	sal
10	4400
20	13000
20	6000
50	5800
50	3500
50	3100
50	2500
50	2600
60	9000
60	6000
60	4200
80	10500
80	8600
80	11000
90	24000
90	17000

4400
9500
3500
6400
10033

表中每个部门的平均薪水

deptno	avg(sal)
10	4400
20	9500
50	3500
60	6400
80	10033.3333
90	19333.3333
110	10150
	7000

...
20 rows selected

1.5.3. 【分组】HAVING 子句

HAVING子句

解详识知

- HAVING子句用来对分组后的结果进一步限制，比如按部门分组后，得到每个部门的最高薪水，可以继续限制输出结果
- 必须跟在GROUP BY后面，不能单独存在

```
--查询每个部门的最高薪水，只有最高薪水大于4000的记录才被输出显示
SELECT deptno, MAX(sal) max_sal FROM emp
GROUP BY deptno HAVING MAX(sal) >4000;
```

1.6. 查询语句执行顺序

1.6.1. 【查询语句执行顺序】查询语句执行顺序

查询语句执行顺序

- 查询语句的执行顺序依下列子句次序：
  1. from 子句：执行顺序为从后往前、从右到左
     - **数据量较少的表尽量放在后面**
  2. where子句：执行顺序为自下而上、从右到左
     - **将能过滤掉最大数量记录的条件写在Where 子句的最右**
  3. group by--执行顺序从左往右分组
     - **最好在GROUP BY前使用WHERE将不需要的记录在GROUP BY之前过滤掉**

解说知识

查询语句执行顺序（续1）

4. having 子句：消耗资源
   - **尽量避免使用，HAVING 会在检索出所有记录之后才对结果集进行过滤，需要排序等操作**
5. select子句：少用*号，尽量取字段名称。
   - **ORACLE 在解析的过程中，通过查询数据字典将*号依次转换成所有的列名，消耗时间**
6. order by子句：执行顺序为从左到右排序，消耗资源

解说知识

2. SQL（关联查询）

2.1. 关联基础

2.1.1. 【关联基础】关联的概念

关联的概念

- 实际应用中所需要的数据，经常会需要查询两个或两个以上的表
- 这种查询两个或两个以上数据表或视图的查询叫做连接查询
- 连接查询通常建立在存在相互关系的父子表之间

```
SELECT table1.column, table2.column
FROM table1, table2
WHERE table1.column1 = table2.column2;
```

解说知识

2.1.2. 【关联基础】笛卡尔积

笛卡尔积

- 笛卡尔积指做关联操作的每个表的每一行都和其它表的每一行做组合，假设两个表的记录条数分别是X和Y，笛卡尔积将返回X * Y条记录。

解说识知

– SELECT COUNT(*) FROM emp; --14条记录

– SELECT COUNT(*) FROM dept; --4条记录

– SELECT emp.ename, dept.dname
FROM emp, dept;--56条记录

2.1.3. 【关联基础】等值连接

等值连接

- 连接查询中最常见的一种，通常是在有主外键关联关系的表间建立，并将连接条件设定为有关系的列，使用等号”=”连接相关的表

解说识知

--查询职员的姓名、职位以及所在部门的名字和所在城市，使用两个相关的列做等值操作

```
SELECT e.ename, e.job, d.dname, d.loc
FROM emp e, dept d
WHERE e.deptno = d.deptno;
```

2.2. 关联查询

2.2.1. 【关联查询】内连接

内连接

- 内连接返回所有满足连接条件的记录。

解说识知

```
SELECT e.ename, d.dname
FROM emp e, dept d
WHERE e.deptno = d.deptno;
```

```
SELECT e.ename, d.dname
FROM emp e JOIN dept d
ON(e.deptno = d.deptno);
```

连接两个表

连接条件

2.2.2. 【关联查询】外连接

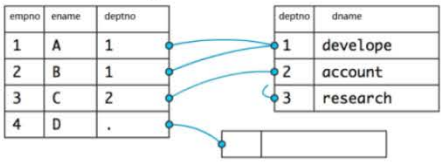
外连接

- 内连接返回满足连接条件的数据记录
- 有些情况下，需要返回那些不满足连接条件的记录，需要使用外连接
- 外联接不仅返回满足连接条件的记录，还将返回不满足连接条件的记录
- 驱动表的概念

```
SELECT table1.column, table2.column
FROM table1  LEFT | RIGHT | FULL [OUTER] JOIN table2
ON table1.column1 = table2.column2;
```

外连接（续1）

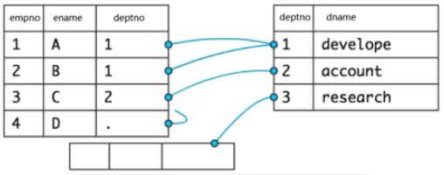
left outer join



empno	ename	deptno	dname
1	A	1	develope
2	B	1	develope
3	C	2	account
4	D		

外连接（续2）

right outer join



empno	ename	deptno	dname
1	A	1	develope
2	B	1	develope
3	C	2	account
		3	research

外连接（续3）

解详识知

- emp 表做驱动表

```
SELECT e.ename, d.dname
FROM emp e LEFT OUTER JOIN dept d
ON e.deptno = d.deptno;
```

- dept 表做驱动表

```
SELECT e.ename, d.dname
FROM emp e RIGHT OUTER JOIN dept d
ON e.deptno = d.deptno;
```

2.2.3. 【关联查询】全外连接

全外连接

解详识知

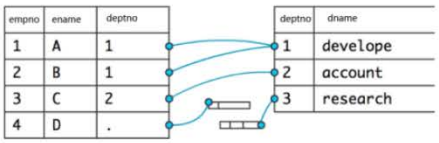
- 全外连接是指，除了返回满足连接条件的记录，还会返回不满足连接条件的所有其它行
- 是左外连接和右外连接查询结果的总和

```
SELECT e.ename, d.dname
FROM emp e FULL OUTER JOIN dept d
ON e.deptno = d.deptno;
```

全外连接（续1）

解详识知

full outer join



empno	ename	deptno	dname
1	A	1	develope
2	B	1	develope
3	C	2	account
4	D		
		3	research

2.2.4. 【关联查询】自连接

自连接

- 自连接是一种特殊的连接查询，数据的来源是一个表，即关联关系来自于单表中的多个列
- 表中的列参照同一个表中的其它列的情况称作自参照表
- 自连接是通过将表用别名虚拟成两个表的方式实现，可以是等值或不等值连接

新学识知

自连接（续1）

- 查出每个职员的经理名字，以及他们的职员编码

```
SELECT worker.empno w_empno, worker.ename
  w_ename, manager.empno m_empno,
  manager.ename m_ename
FROM emp worker join emp manager
ON worker.mgr = manager.empno;
```

JOIN 同一个表

新学识知

经典案例

1. Oracle 基础查询综合示例

- **问题**

编写 SQL 语句，查询职员表（emp 表），逐一完成如下功能：

1）查询 emp 表中的入职日期列(hiredate)，并按照时间的先后顺序列出；

2）查询工资大于 1600 的员工姓名和工资；

3）查询员工号为 7369 的员工的姓名和部门编码；

4）查询工资不在 4000 到 5000 之间的员工的姓名和工资；

5）查询那些尚未分配部门的员工的姓名；

6）已知员工的每月收入为：薪资＋绩效*0.8，如果绩效为 null，则表示绩效为 0。查询员工的姓名以及月收入（列名为 money），并按照月收入升序排列。查询效果如图－1 所示：

ENAME	MONEY
SMITH	800
JAMES	950
ADAMS	1100
MILLER	1300
TURNER	1500
WARD	1650
ALLEN	1840
MARTIN	2370
CLARK	2450
BLAKE	2850
JONES	2975
SCOTT	3000
FORD	3000
KING	5000

图－1

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：查询入职日期列

emp 表中，有 hiredate 列表示入职日期，查询该列，并使用 order by 子句进行升序排列。SQL 语句如下所示：

```
select hiredate from emp order by hiredate;
```

步骤二：查询工资大于 1600 的员工姓名和工资

emp 表中，ename 列表示员工姓名，sal 列表示工资，查询这两列，并使用 where 子句

对工资进行判断。SQL 语句如下所示：

```
select ename, sal from emp where sal > 1600;
```

步骤三：查询员工号为 7369 的员工的姓名和部门编码

emp 表中，empno 列表示员工编号，ename 列表示员工姓名，deptno 列表示部门编码，查询 ename 和 deptno 列，并使用 where 子句对员工编号进行判断。SQL 语句如下所示：

```
select ename, deptno from emp where empno = 7369;
```

步骤四：查询工资不在 4000 到 5000 之间的员工的姓名和工资

需要使用 not between…and… 子句对工资进行范围的判断。SQL 语句如下所示：

```
select ename,  sal from emp where  sal not between 4000 and 5000;
```

步骤五：查询那些尚未分配部门的员工的姓名

如果某条数据的 deptno 列为 null，则表示该员工没有分配部门。因此，需要使用 is null 判断 deptno 列是否为空。SQL 语句如下所示：

```
select ename from emp where deptno is null;
```

步骤六：查询员工的月收入

emp 表中，comm 列表示绩效，该列可能为 null。因此，首先需要使用 nvl 函数对 comm 列进行转换：如果 comm 列为 null，则转换为数值 0；然后，使用计算表达式 sal + nvl(comm,0) * 0.8 计算月收入。

计算完月收入后，使用 order by 子句进行排序。SQL 语句如下所示：

```
select ename, sal + nvl(comm,0) * 0.8 money from emp order by money;
```

2. Oracle 分组查询综合示例

- **问题**

1）查询各职位的员工工资的最大值，最小值，平均值以及总和，查询结果如图 - 2 所示：

JOB	MAX(SAL)	MIN(SAL)	AVG(SAL)	SUM(SAL)
MANAGER	2975	2450	2758.3333333333333333333333333333333333	8275
ANALYST	3000	3000	3000	6000
PRESIDENT	5000	5000	5000	5000
SALESMAN	1600	1250	1400	5600
CLERK	1300	800	1037.5	4150

图 - 2

2）查询各职位的员工人数，查询结果如图 - 3 所示：

JOB	COUNT(*)
MANAGER	3
ANALYST	2
PRESIDENT	1
SALESMAN	4
CLERK	4

图 - 3

3）查询员工的最高工资和最低工资的差距，并显示列名为 DIFFERENCE，查询结果如图 - 4 所示：

DIFFERENCE
4200

图 - 4

4）查询各个管理者属下员工的最低工资，其中最低工资不能低于 800，且没有管理者的员工不计算在内。查询结果如图 - 5 所示：

MGR	MIN(SAL)
7788	1100
7902	800
7839	2450
7698	950
7566	3000
7782	1300

图 - 5

5）查询各个部门中工资大于 1500 的员工人数，查询结果如图 - 6 所示：

DEPTNO	COUNT(*)
30	2
20	3
10	2

图 - 6

6）查询各部门的平均绩效，如果绩效为 null，则按数值 0 进行统计，查询结果如图 - 7 所示：

DEPTNO	AVG_COMM
30	366.6666666666666666666666666666666667
20	0
10	0

图 - 7

- **方案**

聚合函数用于对数据进行统计，用于统计全表的数据，也可以将表的全部数据划分为几组数据，每组数据统计出一个结果。因为是多行数据参与运算返回一行结果，也称作分组函数、多行函数、集合函数。

常见的聚合函数有：

- MAX：用来取得列或表达式的最大值
- MIN：用来取得列或表达式的最小值
- AVG：用来统计列或表达式的平均值
- SUM：用来统计列或表达式的和值
- COUNT：用来计算表中的记录条数

注意：聚合函数忽略 NULL 值。

使用聚合函数进行数据统计时，往往结合 GOURP BY 子句使用。GOURP BY 子句实现按什么分组。

HAVING 子句用来对分组后的结果进一步限制，比如按部门分组后，得到每个部门的最高薪水，可以继续限制输出结果。需要注意的是，HAVING 子句必须跟在 GROUP BY 后面，不能单独存在。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：查询各职位的员工工资的最大值，最小值，平均值以及总和

emp 表中，有 sal 列表示工资，job 列表示职位。因此，需要对 job 列进行分组，统计 sal 列的最大值、最小值、平均值以及总和即可。SQL 语句如下所示：

```
select job, max(sal), min(sal), avg(sal), sum(sal)
  from emp
  group by job;
```

步骤二：查询各职位的员工人数

emp 表中，job 列表示职位。因此，需要对 job 列进行分组，统计各组中记录的条数即可。SQL 语句如下所示：

```
select job, count(*)
  from emp
  group by job;
```

步骤三：查询员工的最高工资和最低工资的差距，并显示列名为 DIFFERENCE

实现此案例，需要分别使用 MAX 和 MIN 函数查询员工的最高工资和最低工资，并求差值。SQL 语句如下所示：

```
select max(sal)-min(sal) "DIFFERENCE" from emp;
```

步骤四：查询各个管理者属下员工的最低工资，其中最低工资不能低于 800，且没有管理者的员工不计算在内

emp 表中，有 mgr 列表示员工的管理者，因此，首先需要对 mgr 列进行分组，并统计每组中 sal 列的最小值。

这样，会统计出每个管理者下属员工的最低工资，为了达到案例的要求，还需要判断 mgr 列不为空，以及最低工资不低于 800。

SQL 语句如下所示：

```
select mgr, min(sal) from emp
   where mgr is not null
   group by mgr
   having min(sal) >= 800;
```

步骤五：查询各个部门中工资大于 1500 的员工人数

emp 表中，有 deptno 列表示员工的所属部门，因此，需要对 deptno 列进行分组，并统计各组中记录的条数。

为了只统计工资大于 1500 的记录数，需要使用 where 子句进行过滤。SQL 语句如下所示：

```
select deptno , count(*) from emp where  sal > 1500 group by deptno;
```

步骤六：查询各部门的平均绩效，如果绩效为 null，则按数值 0 进行统计

emp 表中，有 deptno 列表示员工的所属部门，comm 列表示绩效。因此，首先需要对 deptno 列进行分组，并统计 comm 列的平均值。

但是，因为 comm 列可能为 null，而聚合函数会忽略 null 值。因此，需要使用 NVL 函数对 null 值进行处理。SQL 语句如下所示：

```
SELECT deptno, AVG(NVL(comm,0))  avg_comm FROM emp group by deptno;
```

3. Oracle 关联查询综合示例

- **问题**

1）查询员工的姓名及其所在部门的名字和城市，如图 - 8 所示：

ENAME	DNAME	LOC
SMITH	RESEARCH	DALLAS
ALLEN	SALES	CHICAGO
WARD	SALES	CHICAGO
JONES	RESEARCH	DALLAS
MARTIN	SALES	CHICAGO
BLAKE	SALES	CHICAGO
CLARK	ACCOUNTING	NEW YORK
SCOTT	RESEARCH	DALLAS
KING	ACCOUNTING	NEW YORK
TURNER	SALES	CHICAGO
ADAMS	RESEARCH	DALLAS
JAMES	SALES	CHICAGO
FORD	RESEARCH	DALLAS
MILLER	ACCOUNTING	NEW YORK

图 - 8

2）查询员工的姓名和他的管理者的姓名，查询结果如图 - 9 所示：

ENAME	MGR_NAME
FORD	JONES
SCOTT	JONES
JAMES	BLAKE
TURNER	BLAKE
MARTIN	BLAKE
WARD	BLAKE
ALLEN	BLAKE
MILLER	CLARK
ADAMS	SCOTT
CLARK	KING
BLAKE	KING
JONES	KING
SMITH	FORD

图 - 9

3）查询员工的编号、姓名、部门编码、部门名称以及部门所在城市。要求：把没有部门的员工也查出来，查询结果如图 - 10 所示：

EMPNO	ENAME	DEPTNO	DNAME	LOC
7934	MILLER	10	ACCOUNTING	NEW YORK
7839	KING	10	ACCOUNTING	NEW YORK
7782	CLARK	10	ACCOUNTING	NEW YORK
7902	FORD	20	RESEARCH	DALLAS
7876	ADAMS	20	RESEARCH	DALLAS
7788	SCOTT	20	RESEARCH	DALLAS
7566	JONES	20	RESEARCH	DALLAS
7369	SMITH	20	RESEARCH	DALLAS
7900	JAMES	30	SALES	CHICAGO
7844	TURNER	30	SALES	CHICAGO
7698	BLAKE	30	SALES	CHICAGO
7654	MARTIN	30	SALES	CHICAGO
7521	WARD	30	SALES	CHICAGO
7499	ALLEN	30	SALES	CHICAGO

图 - 10

4）查询员工的信息及其所在部门的信息。要求：把没有员工的部门也查出来，查询结果如图 - 11 所示：

EMPNO	ENAME	DEPTNO	DNAME	LOC
7369	SMITH	20	RESEARCH	DALLAS
7499	ALLEN	30	SALES	CHICAGO
7521	WARD	30	SALES	CHICAGO
7566	JONES	20	RESEARCH	DALLAS
7654	MARTIN	30	SALES	CHICAGO
7698	BLAKE	30	SALES	CHICAGO
7782	CLARK	10	ACCOUNTING	NEW YORK
7788	SCOTT	20	RESEARCH	DALLAS
7839	KING	10	ACCOUNTING	NEW YORK
7844	TURNER	30	SALES	CHICAGO
7876	ADAMS	20	RESEARCH	DALLAS
7900	JAMES	30	SALES	CHICAGO
7902	FORD	20	RESEARCH	DALLAS
7934	MILLER	10	ACCOUNTING	NEW YORK
(null)	(null)	40	OPERATIONS	BOSTON

图 - 11

5）查询员工的信息及其所在部门的信息。要求：只查询没有员工的部门，查询结果如图 - 12 所示：

EMPNO	ENAME	DEPTNO	DNAME	LOC
(null)	(null)	40	OPERATIONS	BOSTON

图 - 12

6）查询并显示 SALES 部门的职位，查询结果如图 - 13 所示：

JOB
MANAGER
SALESMAN
CLERK

图 - 13

7）查询所有部门的名称、所在地、员工数量以及平均工资，查询结果如图 - 14 所示：

DNAME	LOC	EMP_COUNT	SAL_AVG
SALES	CHICAGO	6	1566.6666666666666666666666666666666667
RESEARCH	DALLAS	5	2175
ACCOUNTING	NEW YORK	3	2916.6666666666666666666666666666666667

图 - 14

8）执行下面两条查询语句，并比较查询结果。

```
SELECT e1.ename, e2.ename FROM emp e1, emp e2 WHERE e1.mgr = e2.empno
SELECT e1.ename, e2.ename FROM emp e1, emp e2 WHERE e1.empno = e2.mgr
```

- **方案**

查询两个或两个以上数据表或视图的查询叫做连接查询。连接查询分为：

- 内连接：返回所有满足连接条件的记录
- 外连接：不仅返回满足连接条件的记录，还将返回不满足连接条件的记录。分为左外连接和右外连接；
- 全外连接：左外连接和右外连接查询结果的总和；
- 自连接：数据的来源是一个表，即关联关系来自于单表中的多个列。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：查询员工的姓名及其所在部门的名字和城市

emp 表中，有 deptno 列表示员工所在部门的编码；dept 表中，也有 deptno 列表示部门编码，因此，可以通过此列对两个表进行关联查询。SQL 语句如下所示：

```
select ename , dname , loc
       from emp  e join dept  d
       on e.deptno = d.deptno ;
```

步骤二：查询员工的姓名和他的管理者的姓名

emp 表中，有 mgr 列记载员工的管理者，而管理者同时也在 emp 表中。因此，需要使

用自连接进行查询。SQL 语句如下所示：

```
select t1.ename , t2.ename mgr name
       from emp  t1 join emp  t2
       on t1.mgr = t2.empno ;
```

步骤三：查询员工的编号、姓名、部门编码、部门名称以及部门所在城市（把没有部门的员工也查出来）

emp 表中，有 deptno 列表示员工所在部门的编码；dept 表中，也有 deptno 列表示部门编码，因此，可以通过此列对两个表进行关联查询。因为要求把没有部门的员工也查出来，则需要将 emp 表中的所有数据都列出来，需要使用 emp 表左外连接 dept 表。

SQL 语句如下所示：

```
select e.empno , ename , d.deptno , d.dname , d.loc
       from emp  e left outer join dept  d
       on e.deptno = d.deptno ;
```

步骤四：查询员工的信息及其所在部门的信息（把没有员工的部门也查出来）

要求把没有员工的部门也查出来，则需要将 dept 表中的所有数据都列出来，需要使用 dept 表左外连接 emp 表，或者使用 emp 表右外连接 dept 表。

SQL 语句如下所示：

```
select e.empno , e.ename , d.deptno , d.dname , d.loc
       from dept  d left outer join emp  e
       on e.deptno = d.deptno;
```

步骤五：查询员工的信息及其所在部门的信息（只查询没有员工的部门）

实现此案例，只需要在上一个案例的基础上，添加过滤条件：员工编号为 null。

SQL 语句如下所示：

```
select e.empno , e.ename , d.deptno , d.dname , d.loc
       from dept  d left outer join emp  e
       on e.deptno = d.deptno
       where e.empno is null ;
```

步骤六：查询并显示 SALES 部门的职位

emp 表中，有 job 列表示职位，deptno 列表示部门编号，而部门名称（如 SALES）信息存储在 dept 表中。因此，需要通过 deptno 列对两个表进行关联查询，并根据部门的名称进行过滤。

SQL 语句如下所示：

```
select distinct e.job from emp e, dept d
where e.deptno = d.deptno and d.dname = 'SALES';
```

步骤七：查询所有部门的名称、所在地、员工数量以及平均工资

首先，统计 emp 表各部门的员工数量以及平均工资：需要对 deptno 列进行分组，并统计 sal 列的平均值和记录数；

其次，将上一步中的查询结果作为中间表，与 dept 表，进行关联查询，查询部门的名称、所在地以及上一步中的统计结果。

SQL 语句如下所示：

```
select d.dname, d.loc, e.EMP COUNT, e.SAL AVG
from dept d
join(
  select deptno, count(*) as "EMP_COUNT", avg(sal) as "SAL_AVG" from emp
 group by deptno
       ) e
on d.deptno = e.deptno;
```

步骤八：执行下面两条查询语句，并比较查询结果

分析第一条查询语句：

```
SELECT e1.ename, e2.ename FROM emp e1, emp e2 WHERE e1.mgr = e2.empno
```

此语句中：查询 e1 表中的员工姓名，以及 e1 中员工的管理员的姓名。查询的过程如图 - 15 所示（以 MGR 列中的数据 7902 和 7698 为例）：

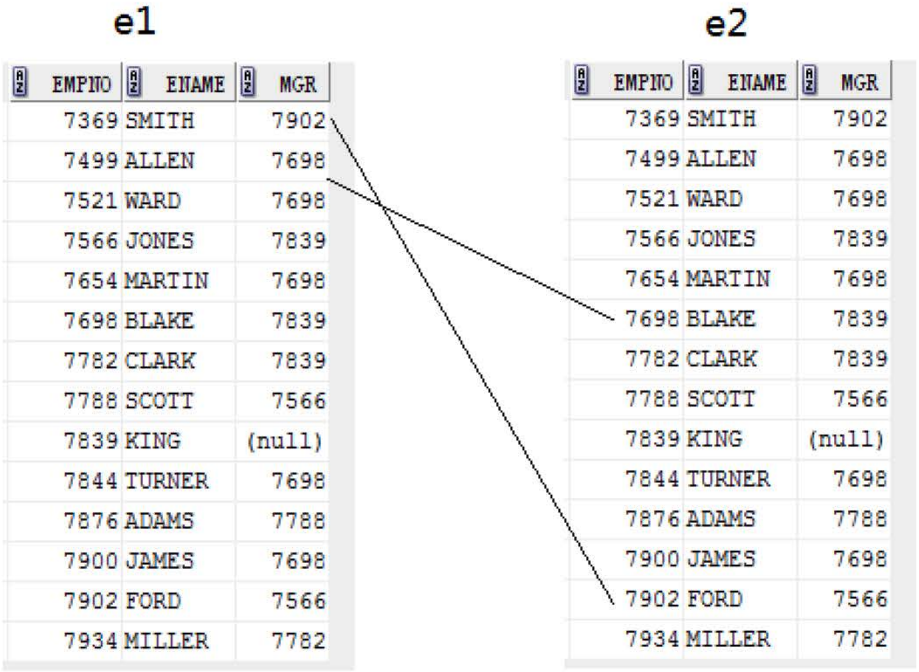


图 - 15

由图 - 15 可以看出，SMITH 的管理者为 FORD，而 ALLEN 和 WARD 拥有相同的管理者：BLAKE。

因此，第一条语句的作用在于：查询每名员工的姓名及其管理者的姓名，查询结果如图 - 16 所示：

FORD	JONES
SCOTT	JONES
JAMES	BLAKE
TURNER	BLAKE
MARTIN	BLAKE
WARD	BLAKE
ALLEN	BLAKE
MILLER	CLARK
ADAMS	SCOTT
CLARK	KING
BLAKE	KING
JONES	KING
SMITH	FORD

图 - 16

继续分析第二条查询语句：

```
SELECT e1.ename, e2.ename FROM emp e1, emp e2 WHERE e1.empno = e2.mgr
```

此语句中：查询 e1 表中的员工姓名，以及 e1 中员工的下属的姓名。查询的过程如图 - 17 所示（以 empno 值为 7566 的数据为例）：

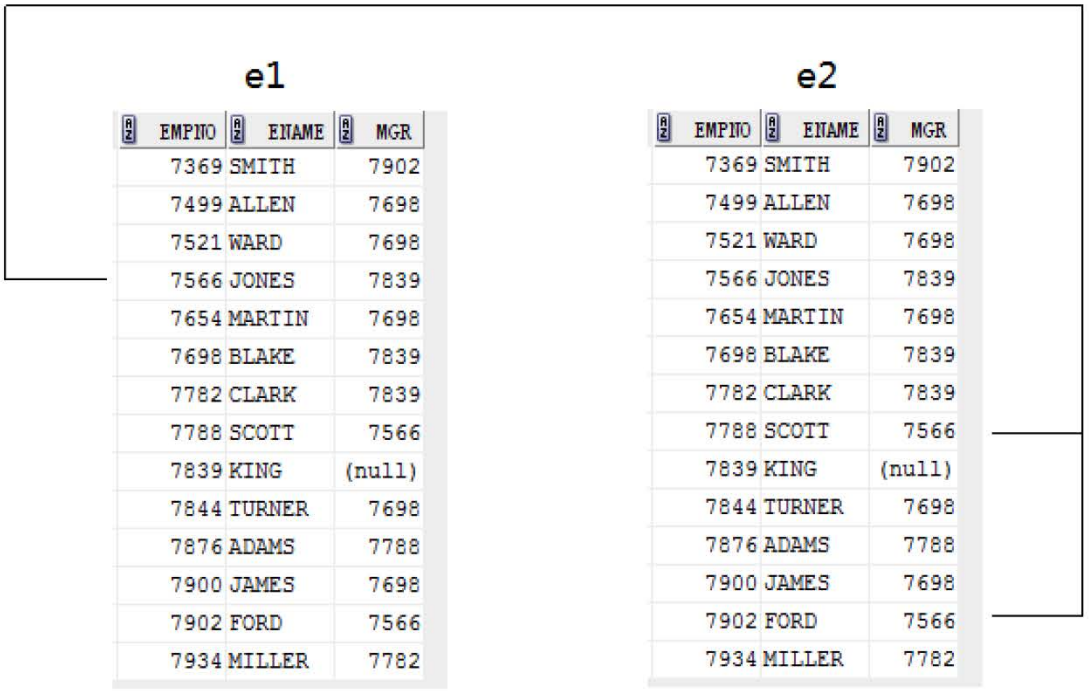


图 - 17

由图 - 17 可以看出，JONES 有两名下属：SCOTT 和 FORD。

因此，第二条语句的作用在于：查询每个管理者姓名及其下属员工的姓名，查询结果如图 - 18 所示：

FORD	SMITH
BLAKE	ALLEN
BLAKE	WARD
KING	JONES
BLAKE	MARTIN
KING	BLAKE
KING	CLARK
JONES	SCOTT
BLAKE	TURNER
SCOTT	ADAMS
BLAKE	JAMES
JONES	FORD
CLARK	MILLER

图 - 18

达内IT培训集团

课后作业

1. Oracle 基础查询综合示例

有职员表 emp，表结构如表 - 1 所示：

表 - 1 职员表 emp 信息

字段名	类型	描述
Empno	NUMBER(4,0)	员工 ID
Ename	VARCHAR2(10)	员工姓名
Job	VARCHAR2(9)	职位
Mgr	NUMBER(4,0)	员工管理者的 ID
Hiredate	DATE	入职日期
Sal	NUMBER(7,2)	薪资
Comm	NUMBER(7,2)	绩效
Deptno	NUMBER(2,0)	员工所在的部门 ID

emp 表中的示例数据如图 - 1 所示：

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	1980/12/17	800.00		20
7499	ALLEN	SALESMAN	7698	1981/2/20	1600.00	300.00	30
7521	WARD	SALESMAN	7698	1981/2/22	1250.00	500.00	30
7566	JONES	MANAGER	7839	1981/4/2	2975.00		20
7654	MARTIN	SALESMAN	7698	1981/9/28	1250.00	1400.00	30
7698	BLAKE	MANAGER	7839	1981/5/1	2850.00		30
7782	CLARK	MANAGER	7839	1981/6/9	2450.00		10
7788	SCOTT	ANALYST	7566	1987/4/19	3000.00		20
7839	KING	PRESIDENT		1981/11/17	5000.00		10
7844	TURNER	SALESMAN	7698	1981/9/8	1500.00	0.00	30
7876	ADAMS	CLERK	7788	1987/5/23	1100.00		20
7900	JAMES	CLERK	7698	1981/12/3	950.00		30
7902	FORD	ANALYST	7566	1981/12/3	3000.00		20
7934	MILLER	CLERK	7782	1982/1/23	1300.00		10

图 - 1

有部门表 dept，表结构如表 - 2 所示：

表 - 2 部门表 dept 信息

字段名	类型	描述
Deptno	NUMBER(2,0)	部门 ID
Dname	VARCHAR2(14)	部门名称
Loc	VARCHAR2(13)	部门所在地

dept 表中的示例数据如图 - 2 所示：

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

图 - 2

需要完成如下查询：

1、查询职员表中，在 20 和 30 号部门工作的员工姓名和部门号。

2、查询职员表中，没有管理者的员工姓名及职位，并按职位排序。

3、查询职员表中，有绩效的员工姓名、薪资和绩效，并按工资倒序排列。

4、查询职员表中，员工姓名的第三个字母是 A 的员工姓名。

5、查询职员表中的职员名字、职位、薪资，并显示为如图 - 3 所示效果：

	OUT_PUT
1	SMITH,CLERK,800
2	ALLEN,SALESMAN,1600
3	WARD,SALESMAN,1250
4	JONES,MANAGER,2975
5	MARTIN,SALESMAN,1250
6	BLAKE,MANAGER,2850
7	CLARK,MANAGER,2450
8	SCOTT,ANALYST,3000
9	KING,PRESIDENT,5000
10	TURNER,SALESMAN,1500
11	ADAMS,CLERK,1100
12	JAMES,CLERK,950
13	FORD,ANALYST,3000
14	MILLER,CLERK,1300

图 - 3

提示：列之间用逗号连接，列名显示成 OUT_PUT。

6、查询职员表中员工号、姓名、工资，以及工资提高百分之 20%后的结果。

7、查询员工的姓名和工资，条件限定为：工资必须大于 1200，并对查询结果按入职时间进行排列，早入职排在前面，晚入职排在后面。

8、查询 ACCOUNT 部门以外的其他部门的编号、名称以及所在地。

2. Oracle 分组查询综合示例

对于如前所示的职员表 emp，需要完成如下查询：

1、查询每个部门中每个职位的最高薪水。

2、有 SQL 语句如下所示：

```
SELECT a.ename, a.sal, a.deptno, b.maxsal
FROM emp  a,
(SELECT deptno, max(sal) maxsal
FROM emp
GROUP BY deptno)  b
WHERE a.deptno = b.deptno
AND a. sal < b.maxsal;
```

此 SQL 语句的功能是什么？写出其查询结果。

3、假设员工表中，员工和管理者中间只有一个层级，也就是说，每个员工最多只有一个上级，作为管理者的员工不再有上级管理者，并且，上级管理者相同的员工，他们属于同一个部门。找出 EMP 中那些工资高于他们所在部门的管理者工资的员工。

4、找出 EMP 中那些工资高于他们所在部门普通员工（不包含管理者）平均工资的员工。

3. Oracle 分组查询综合示例（提高题，选做）

1、下列 SQL 语句出错的原因是（ ）。

```
SELECT classid, AVG(MONTHS_BETWEEN(SYSDATE,entertime))
FROM student
WHERE AVG( MONTHS BETWEEN (SYSDATE, entertime))>12
GROUP BY classid
ORDER BY AVG(MONTHS_BETWEEN (SYSDATE, entertime));
```

A) select 短语中不能出现组函数。

B) where 短语中不能限制分组结果。

C) order by 子句中不能包含组函数。

D) 组函数中不能包含单行函数。

2、有学员表 Student，该表的结构如表 - 3 所示：

表- 3 学员表 Student 信息

字段名	类型	描述
id	NUMBER(4)	学员 ID，Not Null
name	VARCHAR2(30)	学员姓名，Not Null
subject	VARCHAR2(30)	科目
score	NUMBER(8,2)	成绩
classid	NUMBER(2)	班级编码

编写 SQL 语句，查询每班中每个科目的最高成绩。

3、针对表 - 3 所示的 Student 表，执行下述 SQL 语句：

```
1)  SELECT a.name, a.score, a.classid, b.avgscore
2)  FROM student  a,
3)  (SELECT classid, avg(score) avgscore
```

```
4)  FROM student
5)  GROUP BY classid)  b
6)  WHERE a.classid = b.classid
7)  AND a.score > b. avgscore;
```

关于运行结果，下列描述正确的是（ ）。

A) 第一行出现错误

B) 第三行出现错误

C) 第六行出现错误

D) 语句正常执行

4. Oracle 关联查询综合示例

1、有职员表 emp，若需列出所有薪水高于平均薪水值的员工信息，则有 SQL 语句如下：

```
SELECT ename, job FROM emp WHERE sal > avg(sal);
```

上述语句是否正确？如果有错，写出正确的 SQL 语句。

2、有学员分数表，如图 - 4 所示：

name	kecheng	fenshu
张三	语文	81
张三	数学	75
李四	语文	76
李四	数学	90
王五	语文	81
王五	数学	100
王五	英语	90

图 - 4

编写一条 SQL 语句，查询出每门课都大于 80 分的学生姓名。

3、有 USERLIST 表如图 - 5 所示：

TELEPHONE（Varchar(10)，主键）	ACCOUNT（varchar(10)）	RENT (numeric(10,2))
4210001	AAAA	19.50
4210002	AAAA	20.50
4210003	BBBB	100.00
4210004	CCCC	250.00

图 - 5

有 CHAEGE 表如图 - 6 所示：

TELEPHONE	FEE01	FEE02	FEE03	FEE04
4210001	11.00	12.00	13.00	14.00
4210002	21.00	22.00	23.00	24.00
4210003	31.00	32.00	33.00	34.00

图 - 6

请用最少的 SQL 语句，产生如表 - 4 所示的查询结果：

表 - 4 查询结果

ACCOUNT	USERS	RENT	FEE01	FEE02	FEE03	FEE04
AAAA	2	40.00	32.00	34.00	36.00	38.00
BBBB	1	100.00	31.00	32.00	33.00	34.00
CCCC	1	250.00	0.00	0.00	0.00	0.00

其中，数据是经过 USERLIST、CHAEGE 表进行合适的连接，并以 ACCOUNT 字段为关键字分组求和得到。

特别注意：电话号码 421004 在 USERLIST 表中有一条记录，在 CHARGE 表中并没有记录。但是，在查询结果中，合同 CCCC 具有一条记录。

4、有两个表 emp 和 taxgrade，其字段分别为：

emp（员工）表: empname，empno，sal

taxgrade（税别）表： taxmin，taxmax，grade

上述字段中，除字段 empname 外，其他字段均为数值类型。

emp 表的数据如表 - 5 所示：

表 - 5 emp 表示例数据

Empname	empno	sal
Mary	1	450
John	2	800
Jerry	3	1250
Kate	4	200
Neo	5	5750

taxgrade 表的数据如表 - 6 所示：

表 - 6 taxgrade 表示例数据

Taxmin	taxmax	grade
0	500	1
500	1000	2
1000	2000	3
2000	5000	4
5000	10000	5

编写 SQL 语句，查询编号为 1 的员工的税别。

5、有学员表 student，用于记录：学号，姓名，性别，年龄，组织部门；有课程表 course，用于记录：课程编号，课程名称；还有选课表 sc，用于记录：学号，课程编号，成绩。三表的结构以及关联如图 - 7 所示：

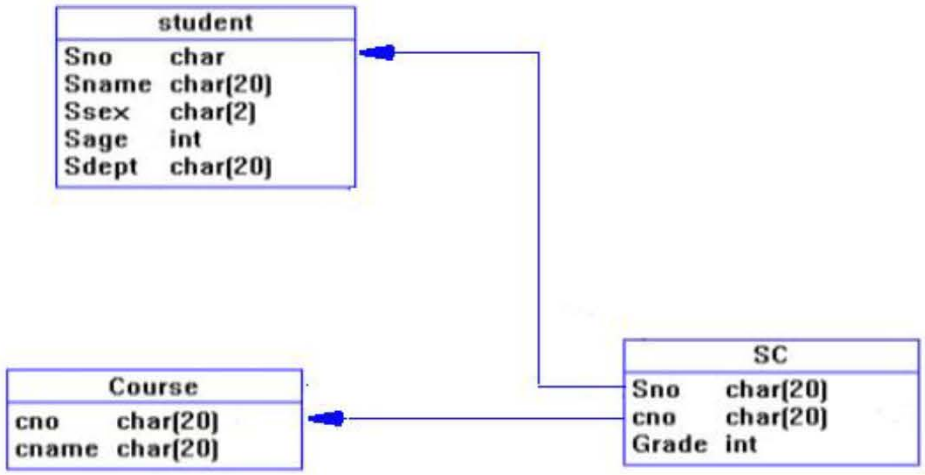


图 - 7

完成如下要求的 SQL 语句：

1)写一个 SQL 语句，查询选修了’计算机原理’的学生学号和姓名

2)写一个 SQL 语句，查询’周星驰’同学选修了的课程名字

6、有表 test ，表结构如表 - 7 所示：

表 - 7 test 表

ID	NAME	AGE	MANAGER(所属主管人 ID)
106	A	30	104
109	B	19	104
104	C	20	111
107	D	35	109
112	E	25	120
119	F	45	NULL

编写 SQL 语句，查询所有年龄比所属主管年龄大的人的 ID 和 NAME。

7、有表 city 记载城市信息，如表 - 8 所示：

表 - 8 city 表

CityNo	CityName	StateNo
BJ	北京	(Null)
SH	上海	(Null)
GZ	广州	GD
DL	大连	LN

有表 state，记载省份信息，如表 - 9 所示：

表 - 9 state 表

StateNo	StateName
GD	广东
LN	辽宁
SD	山东
NMG	内蒙古

欲得到如表 - 10 所示的查询结果：

表 - 10 查询结果

CityNo	CityName	StateNo	StateName
BJ	北京	(Null)	(Null)
DL	大连	LN	辽宁
GZ	广州	GD	广东
SH	上海	(Null)	(Null)

请编写相应的 SQL 语句。

5. Oracle 关联查询综合示例（提高题，选做）

有科目表，记载学员的学习科目数据，如表 - 11 所示：

表 - 11 t_subject 表（科目表）

subject_id	subject_name
1	语文
2	数学

有学员表，记载学员的信息，如表 - 12 所示：

表 - 12 t_student 表（学员表）

student_id	class_id	student_name
1	1	刘一
2	1	陈二
3	1	张三
4	1	李四
5	2	王五
6	2	赵六

有考核规则表，记载考核规则，如表 - 13 所示：

表 - 13 t_assess_rule 表（考核规则表）

rule_id	class_id	assess_type	scale(%)	esc
1	1	test_score	70	成绩(固定)
2	1	check_in	10	考勤
3	1	task_scale	10	作业完成率
4	1	task_score	10	作业成绩
5	2	test_score	70	成绩(固定)
6	2	check_in	15	考勤
7	2	task_scale	15	作业完成率

有学员成绩表，如表 - 14 所示（注，各科的成绩计算方法根据考核规则中占有的比率计算）：

表 - 14 t_performance 表（成绩表）

performance_id	student_id	subject_id	test_score	check_in	task_scale	task_score
1	1	1	90	100	100	90
2	1	2	67	100	100	78
3	2	1	71	100	100	80
4	2	2	100	100	100	95
5	3	1	85	100	100	90
6	3	2	88	100	100	90
7	4	1	81	100	100	90

8	4	2	78	100	100	88
9	5	1	63	95	100	66
10	5	2	87	95	100	90
11	6	1	84	91	100	82
12	6	2	72	91	100	70

1、如何算出一班(class_id=1)每个学生各科的成绩？

2、请对一班(class_id=1)每个学生各科成绩的总分进行排序。

Oracle 数据库基础

Unit04

知识体系.....Page 111

SQL（高级查询）	子查询	子查询在 WHERE 子句中
		子查询在 HAVING 子句中
		子查询在 FROM 部分
		子查询在 SELECT 部分
	分页查询	ROWNUM
		使用子查询进行分页
		分页与 ORDER BY
	DECODE 函数	DECODE 函数基本语法
		DECODE 函数在分组查询中的应用
	排序函数	ROW_NUMBER
		RANK
		DENSE_RANK
	集合操作	UNION、UNION ALL
		INTERSECT
		MINUS
	高级分组函数	ROLLUP
		CUBE
		GROUPING SETS

经典案例.....Page 126

Oracle 子查询综合示例	子查询在 WHERE 子句中
	子查询在 HAVING 子句中
	子查询在 FROM 部分
	子查询在 SELECT 部分
Oracle 分页查询综合示例	ROWNUM
	使用子查询进行分页
	分页与 ORDER BY
Oracle 高级查询综合示例	DECODE 函数基本语法
	DECODE 函数在分组查询中的应用

达内IT培训集团

	ROW_NUMBER
	RANK
	DENSE_RANK
	UNION、UNION ALL
	INTERSECT
	MINUS
	ROLLUP
	CUBE
	GROUPING SETS

课后作业……Page 134

1. SQL（高级查询）

1.1. 子查询

1.1.1. 【子查询】子查询在 WHERE 子句中

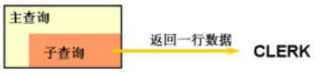
子查询在WHERE子句中

- 在SELECT查询中，在WHERE查询条件中的限制条件不是一个确定的值，而是**来自于另外一个查询的结果**
- 为了给查询提供数据而首先执行的查询语句叫做子查询
- 子查询是嵌入在其它SQL语句中的SELECT语句，大部分时候出现在WHERE子句中
- 子查询嵌入的语句称作主查询或父查询
- 主查询可以是SELECT语句，也可以是其它类型的语句比如DML或DDL语句

解说内容

子查询在WHERE子句中（续1）

- 根据返回结果的不同，子查询可分为单行子查询、多行子查询及多列子查询
  - 单行单列子查询



  - 多行单列子查询

  - 多行多列子查询

解说内容

子查询在WHERE子句中（续2）

```
--查找和SCOTT同职位的员工：
SELECT e.ename, e.job
FROM emp e
WHERE e.job =
(SELECT job FROM emp WHERE ename = 'SCOTT') ;

--查找薪水比整个机构平均薪水高的员工
SELECT deptno, ename, sal
FROM emp e
WHERE sal > (SELECT AVG(sal) FROM emp;
```

先执行子查询，得到结果后再进行主查询

解说内容

子查询在WHERE子句中（续3）

- 如果子查询返回多行，主查询中要使用多行比较操作符
- 多行比较操作符包括IN、ALL、ANY。其中ALL和ANY不能单独使用，需要配合单行比较操作符>、>=、<、<=一起使用

解详识知

--查询出部门中有SALESMAN但职位不是SALESMAN的员工的信息：

```
SELECT empno, ename, job, sal, deptno
FROM emp
WHERE deptno IN
(SELECT deptno FROM emp WHERE job = 'SALESMAN')
AND job <> 'SALESMAN';
```

子查询的结果是多个值

子查询在WHERE子句中（续4）

- 在子查询中需要引用到主查询的字段数据，使用EXISTS关键字
- EXISTS 后边的子查询至少返回一行数据，则整个条件返回TRUE。
- 列出来那些有员工的部门信息

解详识知

```
SELECT deptno, dname FROM dept d
WHERE EXISTS
(SELECT * FROM emp e
 WHERE d.deptno = e.deptno);
```

1.1.2. 【子查询】子查询在 HAVING 子句中

子查询在HAVING子句中

- 查询列出最低薪水高于部门30的最低薪水的部门信息

解详识知

```
SELECT deptno, MIN(sal) min_sal
FROM emp
GROUP BY deptno
HAVING MIN(sal) >
(SELECT MIN(sal) FROM emp WHERE deptno = 30);
```

1.1.3. 【子查询】子查询在 FROM 部分

子查询在FROM部分

- FROM子句用来指定要查询的表
- 如果要在一个子查询的结果中继续查询，则子查询出现在FROM 子句中，这个子查询也称作行内视图或者匿名视图
- 把子查询当作视图对待，但视图没有名字，只能在当前的SQL语句中有效

解说知识

子查询在FROM部分（续1）

- 查询出薪水比本部门平均薪水高的员工信息

```
SELECT e.deptno, e.ename, e.sal
FROM emp e,
(SELECT deptno, AVG(sal) avg_sal FROM emp
 GROUP BY deptno) x
WHERE e.deptno = x.deptno
and e.sal > x.avg_sal
ORDER BY e.deptno;
```

子查询当作表一样使用

解说知识

1.1.4. 【子查询】子查询在 SELECT 部分

子查询在SELECT部分

- 把子查询放在SELECT子句部分，可以认为是外连接的另一种表现形式，使用更灵活

```
SELECT e.ename, e.sal,
       (SELECT d.deptno FROM dept d
        WHERE d.deptno = e.deptno)  deptno
FROM emp e;
```

解说知识

1.2. 分页查询

1.2.1. 【分页查询】ROWNUM

ROWNUM

- 被称作伪列，用于返回标识行数据顺序的数字

```
SELECT ROWNUM, empno, ename, sal
FROM emp;
```

解详识知

- 只能从1计数，不能从结果集中直接截取

```
SELECT ROWNUM, empno, ename, sal
FROM emp
WHERE rownum > 3;
```

查询不到结果

ROWNUM(续1)

- 利用ROWNUM截取结果集中的部分数据，需要用到行内视图

解详识知

```
SELECT  * FROM
(SELECT  ROWNUM rn , e.* FROM emp e )
WHERE rn BETWEEN 8 AND 10;
```

给ROWNUM一个别名

1.2.2. 【分页查询】使用子查询进行分页

使用子查询进行分页

- 分页策略：每次只取一页的数据。每次换页，取下一页的数据。
- 在ORACLE中利用ROWNUM的功能可用来进行分页

解详识知

- 假设结果集共105条，每20条分为一页，则：

Page1： 1 至 20
Page2： 21 至40
…
PageN: (n - 1) * pageSize + 1 至 n * pageSize
共6页

1.2.3. 【分页查询】分页与 ORDER BY

分页与ORDER BY

- 按薪水倒序排列，取出结果集中第8到第10条的记录

```
SELECT  * FROM
  (SELECT  ROWNUM rn , t.* FROM
   (SELECT empno,ename,sal  FROM emp
     ORDER BY sal DESC)  t
  )
 WHERE rn BETWEEN 8 AND 10;
```

- 根据要查看的页数，计算起点值（(n - 1) * pageSize + 1）和终点值（n * pageSize），替换掉BETWEEN和AND的参数，即得到当前页的记录

1.3. DECODE 函数

1.3.1. 【DECODE 函数】DECODE 函数基本语法

DECODE函数基本语法

- DECODE (expr, search1, result1[, search2, result2...][, default])

- DECODE用于比较参数expr的值，如果匹配到哪一个search条件，就返回对应的result结果
- 可以有多组search和result的对应关系，如果任何一个search条件都没有匹配到，则返回最后default的值
- default参数是可选的，如果没有提供default参数值，当没有匹配到时，将返回NULL。

DECODE函数基本语法(续1)

- 查询职员表，根据职员的职位计算奖励金额，当职位分别是‘MANAGER’、‘ANALYST’、‘SALESMAN’时，奖励金额分别是薪水的1.2倍、1.1倍、1.05倍，如果不是这三个职位，则奖励金额取薪水值

```
SELECT ename, job, sal,
    DECODE(job,
            'MANAGER', sal * 1.2,
            'ANALYST', sal * 1.1,
            'SALESMAN', sal * 1.05,
            sal
    ) bonus
FROM emp;
```

DECODE函数基本语法(续2)

解详识知

- 和DECODE函数功能相似的有CASE语句，实现类似于if-else的操作。

```
SELECT ename, job, sal,
   CASE job WHEN 'MANAGER' THEN sal * 1.2
            WHEN 'ANALYST' THEN sal * 1.1
            WHEN 'SALESMAN' THEN sal * 1.05
            ELSE sal END
   bonus
FROM emp;
```

1.3.2. 【DECODE 函数】DECODE 函数在分组查询中的应用

DECODE函数在分组查询中的应用

解详识知

- 按字段内容分组
- 场景：计算职位的人数，analyst/manager属于vip，其余是普通员工operation, 无法用GROUP BY简单实现

```
SELECT DECODE(job,
              'ANALYST', 'VIP',
              'MANAGER', 'VIP',
              'OPERATION') job,
       COUNT(1) job_cnt
FROM emp
GROUP BY DECODE(job, 'ANALYST', 'VIP',
  'MANAGER', 'VIP', 'OPERATION');
```

JOB	JOB_CNT
VIP	5
OPERATION	10

DECODE函数在分组查询中的应用(续1)

解详识知

- 按字段内容排序：
- 场景：

  Dept表中按”OPERATIONS”、“ACCOUNTING”、“SALES”排序，无法按照字面数据排序

```
SELECT deptno, dname, loc
FROM dept
ORDER BY
  DECODE(dname, ' OPERATIONS ',1,'
ACCOUNTING ',2,' SALES ',3);
```

自由定义排序规则

1.4. 排序函数

1.4.1. 【排序函数】ROW_NUMBER

ROW_NUMBER

解详识知

- ROW_NUMBER() OVER(
  PARTITION BY col1 ORDER BY col2)
- 表示根据col1分组，在分组内部根据col2排序
- 此函数计算的值就表示每组内部排序后的顺序编号，组内连续且唯一
- Rownum是伪列， ROW_NUMBER功能更强，可以直接从结果集中取出子集

ROW_NUMBER(续1)

解详识知

- 场景：按照部门编码分组显示，每组内按职员编码排序，并赋予组内编码

```
SELECT deptno, ename, empno,
ROW_NUMBER()
  OVER (PARTITION BY deptno ORDER BY empno)
  AS emp_id
  FROM emp;
```

1.4.2. 【排序函数】RANK

RANK

解详识知

- RANK() OVER(
  PARTITION BY col1 ORDER BY col2)
- 表示根据col1分组，在分组内部根据col2给予等级标识
- 等级标识即排名，相同的数据返回相同排名
- 跳跃排序，如果有相同数据，则排名相同，比如并列第二，则两行数据都标记为2，但下一位将是第四名
- 和ROW_NUMBER的区别是有重复值，而ROW_NUMBER没有

RANK（续1）

- 场景：按照部门编码分组，同组内按薪水倒序排序，相同薪水则按奖金数正序排序，并给予组内等级，用Rank_ID表示

解详识知

```
SELECT deptno, ename, sal, comm,
  RANK() OVER (PARTITION BY deptno
  ORDER BY sal DESC, comm) "Rank_ID"
  FROM emp ;
```

1.4.3. 【排序函数】DENSE_RANK

DENSE_RANK

- DENSE_RANK() OVER(
  PARTITION BY col1 ORDER BY col2)
- 表示根据col1分组，在分组内部根据col2给予等级标识
- 即排名，相同的数据返回相同排名
- 连续排序，如果有并列第二，下一个排序将是三，这一点是和RANK的不同，RANK是跳跃排序

解详识知

DENSE_RANK（续1）

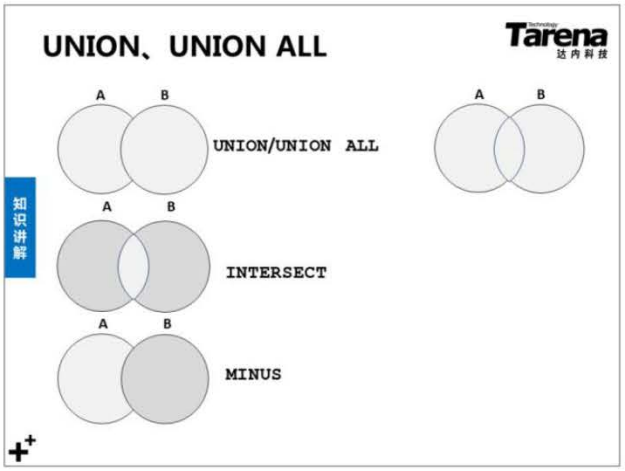
- 场景：关联emp和dept表，按照部门编码分组，每组内按照员工薪水排序，列出员工的部门名字、姓名和薪水

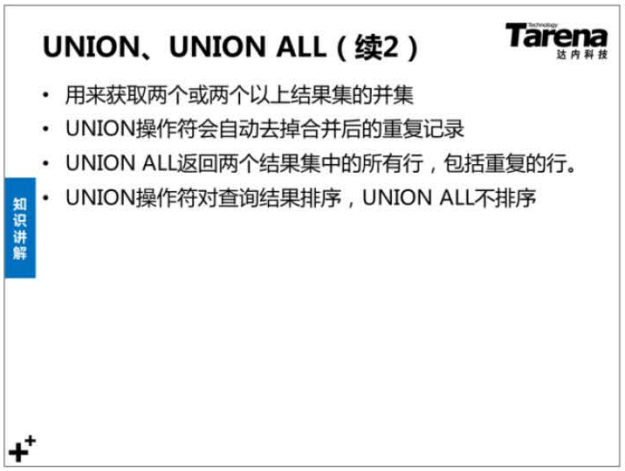
解详识知

```
SELECT d.dname, e.ename, e.sal,
      DENSE_RANK()
    OVER (PARTITION BY e.deptno ORDER BY e.sal)
      AS drank
  FROM emp e join dept d
  on e.deptno = d.deptno;
```

1.5. 集合操作

1.5.1. 【集合操作】UNION、UNION ALL





UNION、UNION ALL（续3）

解详识知

- 合并职位是’MANAGER’的员工和薪水大于2500的员工集合，查看两种方式的结果差别

```
SELECT ename, job, sal FROM emp
WHERE job = 'MANAGER'
UNION
SELECT ename, job, sal FROM emp
WHERE sal > 2500;

SELECT ename, job, sal FROM emp
WHERE job = 'MANAGER'
UNION ALL
SELECT ename, job, sal FROM emp
WHERE sal > 2500;
```

1.5.2. 【集合操作】INTERSECT

INTERSECT

解详识知

- 获得两个结果集的交集，只有同时存在于两个结果集中的数据，才被显示输出
- 使用INTERSECT操作符后的结果集会以第一列的数据作升序排列
- 显示职位是’MANAGER’的员工和薪水大于2500的员工的交集

```
SELECT ename, job, sal FROM emp
WHERE job = 'MANAGER'
INTERSECT
SELECT ename, job, sal FROM emp
WHERE sal > 2500;
```

1.5.3. 【集合操作】MINUS

MINUS

解详识知

- 获取两个结果集的差集
- 只有在第一个结果集中存在，在第二个结果集中不存在的数据，才能够被显示出来。也就是结果集一减去结果集二的结果
- 列出职位是MANAGER但薪水低于2500的员工记录

```
SELECT ename, job, sal FROM emp
WHERE job = 'MANAGER'
MINUS
SELECT ename, job, sal FROM emp
WHERE sal >= 2500;
```

1.6. 高级分组函数

1.6.1. 【高级分组函数】ROLLUP

ROLLUP

解详识知

- ROLLUP、CUBE 和 GROUPING SETS 运算符是 GROUP BY 子句的扩展，可以生成与使用 UNION ALL 来组合单个分组查询时相同的结果集，用来简化和高效的实现统计查询
- GROUP BY ROLLUP(a, b, c)
- GROUP BY CUBE(a, b, c)
- GROUP BY GROUPING SETS ((a), (b))

ROLLUP（续1）

解详识知

- 假设有表test，有a、b、c、d四个列。

```
SELECT a,b,c,SUM(d) FROM test GROUP BY ROLLUP(a,b,c)
等价于：
SELECT a,b,c,SUM(d) FROM test GROUP BY a,b,c
        UNION ALL
SELECT a,b,null,SUM(d) FROM test GROUP BY a,b
        UNION ALL
SELECT a,null,null,SUM(d) FROM test GROUP BY a
        UNION ALL
SELECT null,null,null,sum(d) FROM test
```

- 对ROLLUP的列从右到左以一次少一列的方式进行分组直到所有列都去掉后的分组(也就是全表分组)
- 对于n个参数的ROLLUP，有n+1次分组

ROLLUP（续2）

解详识知

- 准备数据：2010-2011年每月每天的销售额

年	月	日	销售额
2010	1	1	58.82
2010	1	2	47.91
2010	1	3	84.57
……	……	……	……
2011	12	29	76.92
2011	12	30	51.93
2011	12	31	78.11

ROLLUP（续3）

解详识知

```
SQL>DROP TABLE sales_tab;
SQL>CREATE TABLE sales_tab (
  year_id   NUMBER NOT NULL,
  month_id  NUMBER NOT NULL,
  day_id    NUMBER NOT NULL,
  sales_value NUMBER(10,2) NOT NULL
);
```

创建一个测试表

给测试表插入随机数据，2010-2011年，1-12月，1-31天，销售额是小于100的随机浮点数

```
SQL>INSERT INTO sales_tab
SELECT TRUNC(DBMS_RANDOM.value(low => 2010, high => 2012)) AS year_id,
     TRUNC(DBMS_RANDOM.value(low => 1, high => 13)) AS month_id,
     TRUNC(DBMS_RANDOM.value(low => 1, high => 32)) AS day_id,
     ROUND(DBMS_RANDOM.value(low => 1, high => 100), 2) AS sales_value
FROM   dual
CONNECT BY level <= 1000;
SQL>COMMIT;
```

ROLLUP（续4）

解详识知

```
SQL>SELECT SUM(sales_value) AS sales_value FROM   sales_tab;
```

```
SQL>SELECT year_id, COUNT(*) AS num_rows,
     SUM(sales_value) AS sales_value
FROM   sales_tab
GROUP BY year_id
ORDER BY year_id;
```

YEAR_ID	NUM_ROWS	SALES_VALUE
2010	470	23094.63
2011	530	26123.37

复习组函数，查看查询结果

```
SQL>SELECT year_id, month_id,
     COUNT(*) AS num_rows,
     SUM(sales_value) AS sales_value
FROM   sales_tab
GROUP BY year_id, month_id
ORDER BY year_id, month_id;
```

YEAR_ID	MONTH_ID	NUM_ROWS	SALES_VALUE
2010	1	35	1450.55
2010	2	35	1593.23
2010	3	36	1883.68
2010	4	50	2379.55
2010	5	34	1904.79
2010	6	47	1886.48
2010	7	43	2352.98
2010	8	40	1964.24
2010	9	38	1850.47
2010	10	32	1793.25

ROLLUP（续5）

解详识知

```
SELECT year_id, month_id,
SUM(sales_value) AS sales_value
FROM   sales_tab
GROUP BY
ROLLUP (year_id, month_id)
ORDER BY year_id, month_id;
```

YEAR_ID	MONTH_ID	SALES_VALUE
2010	1	1450.55
2010	2	1593.23
2010	3	1883.68
2010	4	2379.55
2010	5	1904.79
2010	6	1886.48
2010	7	2352.98
2010	8	1964.24
2010	9	1850.47
2010	10	1793.25
2010	11	1900.09
2010	12	2135.32
2010	(null)	23094.63
2011	1	2079.73
2011	2	2208.36

计算出2010年的总和

ROLLUP（续6）

解详识知

```
SELECT year_id, month_id, day_id, SUM(sales_value) AS
    sales_value
FROM   sales_tab
GROUP BY ROLLUP (year_id, month_id, day_id)
ORDER BY year_id, month_id, day_id;
```

YEAR_ID	MONT...	DAY_ID	SALES_VALUE
2011	12	27	55.07
2011	12	28	59.09
2011	12	29	11.91
2011	12	30	67.56
2011	12	(null)	2222.13
2011	(null)	(null)	26123.37
(null)	(null)	(null)	49218

2011年12月的总和

2011年的总和

整张表的总和

1.6.2. 【高级分组函数】CUBE

CUBE

解详识知

- GROUP BY CUBE(a, b, c)
- 对cube的每个参数，都可以理解为取值为参与分组和不参与分组两个值的一个维度，所有维度取值组合的集合就是分组后的集合
- 对于n个参数的cube，有2^n次分组
- GROUP BY CUBE(a,b,c)，首先对(a,b,c)进行GROUP BY，然后依次是(a,b)，(a,c)，(a)，(b,c)，(b)，(c)，最后对全表进行GROUP BY操作，一共是2^3=8次分组

CUBE（续1）

解详识知

```
SELECT a,b,c,SUM(d) FROM test GROUP BY CUBE(a,b,c);
等价于
SELECT a,b,c,SUM(d) FROM test GROUP BY a,b,c
UNION ALL
SELECT a,b,NULL,SUM(d) FROM test GROUP BY a,b
UNION ALL
SELECT a,NULL,c,SUM(d) FROM test GROUP BY a,c
UNION ALL
SELECT a,NULL,NULL,SUM(d) FROM test GROUP BY a
UNION ALL
SELECT NULL,b,c,SUM(d) FROM test GROUP BY b,c
UNION ALL
SELECT NULL,b,NULL,SUM(d) FROM test GROUP BY b
UNION ALL
SELECT NULL,NULL,c,SUM(d) FROM test GROUP BY c
UNION ALL
SELECT NULL,NULL,NULL,SUM(d) FROM test ;
```

等价于只是方便理解，其内部运行机制并不相同，其效率远高于UNION ALL

CUBE（续2）

解详识知

```
SELECT year_id, month_id,
       SUM(sales_value) AS sales_value
FROM   sales_tab
GROUP BY CUBE (year_id, month_id)
ORDER BY year_id, month_id;
```

YEAR_ID	MONTH_ID	SALES_VALUE
2011	8	1841.27
2011	9	2873.54
2011	10	1612.09
2011	11	2130.6
2011	12	2222.13
2011	(null)	26123.37
(null)	1	3530.28
(null)	2	3801.59
(null)	3	3904.83

除了计算年度总和，还计算月度总和

CUBE（续3）

解详识知

```
SELECT year_id,  month_id, day_id,
       SUM(sales_value) AS sales_value
FROM   sales_tab
GROUP BY CUBE (year_id, month_id, day_id)
ORDER BY year_id, month_id, day_id;
```

操作前预测下结果？

1.6.3. 【高级分组函数】GROUPING SETS

GROUPING SETS

解详识知

- GROUPING SETS 运算符可以生成与使用单个 GROUP BY、ROLLUP 或 CUBE 运算符所生成的结果集相同的结果集
- 如果不需要获得由完备的 ROLLUP 或 CUBE 运算符生成的全部分组，则可以使用 GROUPING SETS 仅指定所需的分组
- GROUPING SETS 列表可以包含重复的分组

GROUPING SETS（续1）

解详识知

```
SELECT year_id, month_id, SUM(sales_value)
FROM sales_tab
GROUP BY CUBE (year_id,month_id)
order by 1, 2;

SELECT year_id, month_id, SUM(sales_value)
FROM sales_tab
GROUP BY GROUPING SETS ( (year_id), (month_id))
order by 1, 2;
```

只列出感兴趣的数据，从39条降低到14条

GROUPING SETS（续2）

解详识知

- 分组示例：
  - 使用GROUP BY GROUPING SETS(a,b,c)，则对(a),(b),(c)进行GROUP BY
  - 使用GROUP BY GROUPING SETS((a,b),c), 则对(a,b),(c)进行GROUP BY
  - GROUPING BY GROUPING SET(a,a)，则对(a)进行2次GROUP BY, GROUPING SETS的参数允许重复

经典案例

1. Oracle 子查询综合示例

- **问题**

对于 emp 表和 dept 表，完成如下各个案例。

1）执行下列 SQL 语句：

```
SELECT ename,sal FROM emp WHERE sal=
(SELECT sal FROM emp
WHERE ename= 'smith' OR deptno=20);
```

这条语句出错的原因在于（ ）。

A．子查询中不能出现 where 子句。

B．逻辑运算符 OR 不允许出现在 where 子句中。

C．子查询得到多行结果，主查询中使用的是单行比较运算符。

D．子查询得到单行结果，主查询中使用的是多行比较运算符。

2）写 SQL 语句，查询哪个部门的平均工资是最高的，列出部门编码、平均工资。

3）写 SQL 语句，列出各个部门中工资最高的员工的信息：名字、部门号、工资。

4）写 SQL 语句，查询管理者是“KING”的员工姓名（ ename ）和工资（ sal ）。

5）写 SQL 语句，查询部门所在地(loc)为“NEW YORK”的部门的员工的姓名(ename)，部门名称（ dname ）和岗位名称（ job ）。

6）写 SQL 语句，查询工资比公司平均工资高的所有员工的员工号（ empno ），姓名（ ename ）和工资（ sal ）。

7）写 SQL 语句 ,查询姓名中包含字母“u”的员工在相同部门的员工的员工号(empno)和姓名（ ename ）。

8）写 SQL 语句，查询哪些员工的薪水比本部门的平均薪水低。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：子查询得到多行结果，主查询中要求使用多行比较运算符

案例中，子查询 SQL 语句：

```
SELECT sal FROM emp
WHERE ename= 'smith' OR deptno=20;
```

该 SQL 语句返回多条数据，对于这种返回多条数据的情况，Oracle 要求使用多行比较

运算符。本案例中，在主查询中使用了“=”运算符，它属于单行运算符，因此，在此案例中，正确的答案为 C。

步骤二：查询哪个部门的平均工资是最高的

首先，使用 group by 子句配合子查询，查询出平均工资最高的部门，SQL 语句如下所示：

```
select max(avg(sal)) from emp group by deptno;
```

然后，将上述查询的结果作为主查询的条件，进一步查询出平均工资最高的部门的部门编码，SQL 语句如下所示：

```
select deptno, avg(sal) from emp
group by deptno
having avg(sal) = (select max(avg(sal)) from emp group by deptno);
```

上述 SQL 语句中，设置分组的条件使用了 having 子句。

步骤三：列出各个部门中工资最高的员工的信息

首先，使用 group by 子句配合组函数查询出各个部门的最高工资，SQL 语句如下所示：

```
select deptno , max(sal) from emp group by deptno;
```

然后，将上述查询的结果作为 where 条件的值，即使用 in 操作符判断哪些员工与上述查询的结果集中的任一数据相等，SQL 语句如下所示：

```
select ename ,  sal , deptno from emp
where (deptno , sal) in
(select deptno , max(sal) from emp group by deptno);
```

步骤四：查询管理者是“KING”的员工姓名（ename）和工资（sal）

首先，查询员工“KING”的员工编号，SQL 语句如下所示：

```
select empno from emp where ename = 'KING';
```

然后，将上述查询的结果作为 where 条件的值，当员工的管理者 ID（mgr）等于上述查询结果时，该员工的管理者即为“KING”，SQL 语句如下所示：

```
select ename, sal
from emp
where mgr = (select empno from emp where ename = 'KING');
```

步骤五：查询部门所在地（loc）为“NEW YORK”的部门的员工信息

首先，查询部门所在地为“NEW YORK”的部门编号，SQL 语句如下所示：

```
select deptno from dept where loc = 'NEW YORK';
```

然后，将上述查询的结果作为 where 条件的值，查询部门所在地为“NEW YORK”的部门里的所有员工的员工号（empno）、姓名（ename）和工资（sal）。SQL 语句如下所示：

```
select e.ename, d.dname, e.job
from emp e join dept d
on e.deptno = d.deptno
where e.deptno = (select deptno from dept where loc = 'NEW YORK');
```

由于需要显示部门名称（dname），因此在上述 SQL 语句中，使用 join... on 将表 emp 和表 dept 做内连接查询。

步骤六： 查询工资比公司平均工资高的所有员工信息

首先，查询所有员工的平均工资，SQL 语句如下所示：

```
select avg(sal) from emp;
```

然后，将上述查询的结果作为 where 条件的值，当职员表中的某条记录的工资（sal）大于上述查询结果时，该员工的工资即高于公司的平均工资，SQL 语句如下所示：

```
select empno, ename, sal
from emp
where  sal > (select avg(sal) from emp);
```

步骤七： 查询姓名中包含字母“u”的员工在相同部门的员工信息

首先，使用 like 运算符查询姓名中包含字母“u”的员工的员工编号，SQL 语句如下所示：

```
select deptno from emp where ename like '%U%';
```

然后，将上述查询的结果作为 where 条件的值，当职员表中的某条记录的部门编号等于上述查询结果中的任意一个时，该员工所在的部门与姓名中包含字母“u”的员工的部门相同，SQL 语句如下所示：

```
select empno, ename
from emp
where deptno in (select deptno from emp where ename like '%U%');
```

步骤八：查询哪些员工的薪水比本部门的平均薪水低

本题可以使用关联子查询来解决，关联子查询的语法如下所示：

```
SELECT  column1, column2, …
FROM    table1 o
WHERE   column1 operator
                 (SELECT column
```

```
                    FROM table2 i
                    WHERE  i.expr1 = o.expr2)
```

本题中要解决的问题为如何在主查询和子查询中来表示同一部门，SQL 语句如下所示：

```
select avg(sal)
from emp b
where b.deptno = a.deptno ;
```

其中 a.deptno 的 a 为主查询中 emp 表的别名。

将上述查询作为主查询的条件，查询哪些员工的薪水比本部门的平均薪水低，SQL 语句如下所示：

```
select ename, sal, deptno
       from emp a
       where sal < ( select avg(sal)
                     from emp b
                     where b.deptno = a.deptno ) ;
```

2. Oracle 分页查询综合示例

- **问题**

根据上一案例中的表结构和示例数据，完成如下查询：

1）查询 emp 表前 5 条记录

2）查询第 3-5 条记录，无需排序

3）查询公司工资最高的三个人

4）查询公司工资最低的五个人

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：查询 emp 表前 5 条记录

使用 ROWNUM 伪列，查询 emp 表前 5 条记录，SQL 语句如下所示：

```
select * from emp where rownum <= 5;
```

ROWNUM 是一个伪列，对查询返回的行编号即行号，由 1 开始依次递增。注意：Oracle 的 rownum 数值是在获取每行之后才赋予的，因此使用“rownum>数字”是查询不到数据的。

步骤二：查询第 3-5 条记录，无需排序

首先，利用 ROWNUM 截取结果集中的部分数据，需要用到行内视图，SQL 语句如下所示：

```
select rownum num, e.* from emp e;
```

上述 SQL 语句中给 ROWNUM 一个别名 num。

然后，将上述查询的结果作为一张虚表，其中，num 为这张表中的一列。查询第 3-5 条记录，SQL 语句如下所示：

```
select * from (select rownum num, e.* from emp e)
where num >= 3 and num <= 5;
```

步骤三：查询公司工资最高的三个人

首先，按工资降序排列员工数据，SQL 语句如下所示：

```
select * from emp order by sal desc;
```

然后，将上述查询的结果作为一张虚表，再根据 ROWNUM 的特性查询公司工资最高的三个人，SQL 语句如下所示：

```
select * from
(select * from emp order by sal desc)
where rownum <= 3;
```

步骤四：查询公司工资最低的五个人

首先，按工资升序排列员工数据，SQL 语句如下所示：

```
select * from emp order by sal;
```

然后，将上述查询的结果作为一张虚表，再根据 ROWNUM 的特性查询公司工资最低的五个人，SQL 语句如下所示：

```
select * from
(select * from emp order by sal)
where rownum <= 5;
```

3. Oracle 高级查询综合示例

- **问题**

本案例的详细要求如下：

1）计算职位的人数，其中，“ANALYST”和“MANAGER”职位用“VIP”表示，其余是普通员工，职位用“OPERATION”显示。分别计算 VIP 职位和普通职员职位的人数，数据显示效果如图-1 所示：

	JOB	JOB_CNT
1	VIP	5
2	OPERATION	9

图-1

2） 将 dept 表中的数据中按”OPERATIONS”、“ACCOUNTING”、“SALES”进行自定义排序。

3） 按照部门编码分组显示，每组内按职员编码排序，并赋予组内编码，要求使用 ROW_NUMBER 函数完成。

4）将职员表中按照部门编码分组，同组内按薪水降序排列，相同薪水则按奖金数升序排列，并显示等级标识，用 Rank_ID 表示，要求使用 RANK 函数完成。

5） 关联 emp 和 dept 表，按照部门编码分组，每组内按照员工薪水升序排列，列出员工的部门名字、姓名和薪水及等级 drank，要求使用 DENSE_RANK 函数完成。

6） 创建 sales_tab，该表存储了 2010-2011 年每月每天的销售额，创建表的 SQL 语句如下所示：

```
CREATE TABLE sales_tab (
  year_id   NUMBER NOT NULL,
  month_id   NUMBER NOT NULL,
  day_id   NUMBER NOT NULL,
  sales_value NUMBER(10,2) NOT NULL
);
```

向 sales_tab 表中插入测试数据，SQL 语句如下所示：

```
INSERT INTO sales_tab
SELECT TRUNC(DBMS_RANDOM.value(2010, 2012)) AS year_id,
       TRUNC(DBMS_RANDOM.value(1, 13)) AS month_id,
       TRUNC(DBMS_RANDOM.value(1, 32)) AS day_id,
       ROUND(DBMS_RANDOM.value(1, 100), 2) AS sales_value
FROM   dual
CONNECT BY level <= 1000;
```

以上 SQL 语句向 sales_tab 表中插入 1000 条随机数据，年的范围为 2010-2011 年，月范围为 1-12 月，日范围为 1-31 天，销售额的范围为 1 到 100（不包括 100）的随机浮点数。

请统计 2010-2011 年每月每日的销售额，要求使用 ROLLUP 函数来完成。

7） 统计年月的销售额，要求使用 CUBE 函数来完成。

8） 统计年月的销售额，要求使用 GROUPING SETS 函数来完成。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一： 使用 DECODE 函数，按要求显示数据

使用 DECODE 函数，计算职位的人数，“ANALYST”和“MANAGER”职位用“VIP”表示，其余是普通员工，职位用“OPERATION”表示。分别计算 VIP 职位和普通职员职位的人数，SQL 语句如下所示：

```
SELECT DECODE(job,
                    'ANALYST', 'VIP',
                    'MANAGER', 'VIP',
```

```
                    'OPERATION') job,
            COUNT(1) job cnt
FROM emp
GROUP BY DECODE(job, 'ANALYST', 'VIP', 'MANAGER', 'VIP', 'OPERATION');
```

步骤二：使用 DECODE 函数实现自定义排序

使用 DECODE 函数，将 dept 表中的数据按"OPERATIONS"、"ACCOUNTING"、"SALES"进行自定义排序，SQL 语句如下所示：

```
SELECT deptno, dname, loc
FROM dept
ORDER BY
DECODE(dname, 'OPERATIONS',1,'ACCOUNTING',2,'SALES',3);
```

步骤三：使用 ROW_NUMBER 函数实现分组排序

使用 ROW_NUMBER 函数按照部门编码分组显示，每组内按职员编码排序，并赋予组内编码，SQL 语句如下所示：

```
SELECT deptno, ename, empno,
ROW NUMBER()
OVER (PARTITION BY deptno ORDER BY empno)    AS emp id
FROM emp;
```

ROW_NUMBER 函数计算的值就表示每组内部排序后的顺序编号，组内连续且唯一。

步骤四：使用 RANK 函数实现分组排序

使用 RANK 函数将职员表中按照部门编码分组，同组内按薪水降序排列，相同薪水则按奖金数升序排列，并显示等级标识，用 Rank_ID 表示，SQL 语句如下所示：

```
SELECT deptno, ename, sal, comm,
RANK() OVER (PARTITION BY deptno
ORDER BY sal DESC, comm) "Rank_ID"
FROM emp;
```

RANK 函数排序后的等级标识是跳跃的，即如果有相同数据，则排名相同，比如并列第二，则两行数据都标记为 2，但下一位将是第四名。

步骤五： 使用 DENSE_RANK 函数实现分组排序

关联 emp 和 dept 表，使用 DENSE_RANK 函数，按照部门编码分组，每组内按照员工薪水升序排列，列出员工的部门名字、姓名和薪水及等级 drank，SQL 语句如下所示：

```
SELECT d.dname, e.ename, e.sal,
       DENSE_RANK()
  OVER (PARTITION BY e.deptno ORDER BY e.sal)
       AS drank
FROM emp e join dept d
on e.deptno = d.deptno;
```

DENSE_RANK 函数排序后的等级标识是连续的，即如果有并列第二，下一个排序将是三，

这一点是和 RANK 函数不同，RANK 是跳跃排序。

步骤六： 使用 ROLLUP 函数统计销售额

使用 ROLLUP 函数统计 2010-2011 年每月每日的销售额，SQL 语句如下所示：

```
SELECT year_id, month_id, day_id, SUM(sales_value) AS sales_value
FROM   sales_tab
GROUP BY ROLLUP (year_id, month_id, day_id)
ORDER BY year_id, month_id, day_id;
```

上述 SQL 语句会依次按照(year_id, month_id, day_id)分组、(year_id, month_id) 分组、(day_id) 分组以及全表分组。

步骤七： 使用 CUBE 函数统计销售额

使用 CUBE 函数统计年月的销售额，SQL 语句如下所示：

```
SELECT year_id, month_id,
       SUM(sales_value) AS sales_value
FROM   sales_tab
GROUP BY CUBE (year_id, month_id)
ORDER BY year_id, month_id;
```

上述 SQL 语句会依次按照(year_id, month_id)分组、(year_id)分组、(month_id) 分组以及全表分组。

步骤八： 使用 GROUPING SETS 函数统计销售额

使用 GROUPING SETS 函数统计年月的销售额，SQL 语句如下所示：

```
SELECT year_id, month_id, SUM(sales_value)
FROM sales_tab
GROUP BY GROUPING SETS ((year_id), (month_id))
order by 1, 2;
```

上述 SQL 语句会依次按照(year_id)分组、(month_id)分组。

课后作业

1. Oracle 子查询精选面试题

有职员表 emp，表结构表 - 1 所示：

表 - 1 职员表 emp 信息

字段名	类型	描述
Empno	NUMBER(4,0)	员工 ID
Ename	VARCHAR2(10)	员工姓名
Job	VARCHAR2(9)	职位
Mgr	NUMBER(4,0)	员工管理者的 ID
Hiredate	DATE	入职日期
Sal	NUMBER(7,2)	薪资
Comm	NUMBER(7,2)	绩效
Deptno	NUMBER(2,0)	员工所在的部门 ID

职员表 emp 的示例数据如图-1 所示：

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	1980/12/17	800.00		20
7499	ALLEN	SALESMAN	7698	1981/2/20	1600.00	300.00	30
7521	WARD	SALESMAN	7698	1981/2/22	1250.00	500.00	30
7566	JONES	MANAGER	7839	1981/4/2	2975.00		20
7654	MARTIN	SALESMAN	7698	1981/9/28	1250.00	1400.00	30
7698	BLAKE	MANAGER	7839	1981/5/1	2850.00		30
7782	CLARK	MANAGER	7839	1981/6/9	2450.00		10
7788	SCOTT	ANALYST	7566	1987/4/19	3000.00		20
7839	KING	PRESIDENT		1981/11/17	5000.00		10
7844	TURNER	SALESMAN	7698	1981/9/8	1500.00	0.00	30
7876	ADAMS	CLERK	7788	1987/5/23	1100.00		20
7900	JAMES	CLERK	7698	1981/12/3	950.00		30
7902	FORD	ANALYST	7566	1981/12/3	3000.00		20
7934	MILLER	CLERK	7782	1982/1/23	1300.00		10

图-1

有部门表 dept，表结构如表 - 2 所示：

表 - 2 部门表 dept 信息

字段名	类型	描述
Deptno	NUMBER(2,0)	部门 ID
Dname	VARCHAR2(14 BYTE)	部门名称
Loc	VARCHAR2(13 BYTE)	部门所在地

部门表 dept 的示例数据如图-2 所示：

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

图-2

请根据上述表结构和示例数据，完成如下查询。

1. SALES 部门有哪些职位？
2. 哪些人不是别人的经理？
3. 谁的薪水比 FORD 高？如果有多个同名，比任何一个叫 FORD 的人高就行。
4. 谁和 FORD 同部门？列出除了 FORD 之外的员工名字。
5. 哪个部门的人数比部门 20 的人数多？
6. 列出员工名字和职位 ，查询员工所在的部门平均薪水大于 2000 元的员工信息。

2. 简述 where 子句中为什么不能写 rownum>...

3. Oracle 分页查询面试题

news 表的表结构如图-3 所示：

Name	Type
ID	NUMBER(6)
TITLE	CHAR(200)
CONTENT	CHAR(1000)
OCCURTIME	DATE

图-3

写 SQL 语句，按新闻时间倒序排列，即最新的排在最前面，每页输出 5 条，查询出第二页。

4. 下列关于 decode 函数，说法正确的是

A. decode 函数用来实现 IF-ELSE 的逻辑功能
B. decode 函数建议设置默认值，如果未能与任一搜索条件匹配则函数返回默认值
C. decode 函数只能处理字符串数据
D. decode 函数的功能可以用 case 语句替代实现

5. 下面 SQL 语句的输出结果是

```
SELECT ename, job,
DECODE(job,'PRESIDENT','A',
'MANAGER','B',
```

```
    'ANALYST','C',
    'SALESMAN','D',
    'CLERK','E'
    ) AS "Grade"
    FROM EMP;
```

6. Oracle 高级查询经典面试题

1．有学员表 student，表结构如表 - 3 所示：

表 - 3 学员表 student 信息

字段名	类型	描述
Sid	NUMBER(4,0)	学生编码
Sname	VARCHAR2(20)	学生姓名
Score	NUMBER(4,1)	学生成绩
class_id	NUMBER(4,0)	所在专业

学员表 student 的示例数据如图-4 所示：

SID	SNAME	SCORE	CLASS_ID
1001	marry	55.0	101
1002	jerry	66.0	101
1003	tom	74.0	101
1004	jim	88.0	90
1005	allen	100.0	90
1006	terry	66	101

图-4

按照学生所在专业编码分组显示，每一专业组内按学生成绩倒序排序，并赋予组内编码。

2．在上题背景下，学生表中按照所在专业分组，同专业内按成绩倒序排序，成绩相同则按学号正序排序，并给予组内等级，用 Rank_ID 表示。

3．有专业表 class，表结构如表 - 4 所示：

表 - 4 专业表 class 信息

字段名	类型	描述
cid	NUMBER(4,0)	专业编码
Cname	VARCHAR2(20)	专业名称

专业表的示例数据如图-5 所示：

CID	CNAME
101	java
90	php

图-5

学员表的 class_id 数据参照 class 表 cid 列的数据。关联学员表 student 和 class 表，按照 class_id 分组，每组内按照学生成绩倒序排序，相同成绩按照学号正序排列，列出学生所在的专业名字、学生姓名、成绩及等级 drank 。

4．创建 mygroup，该表存储了每组员工的工资情况，创建表的 SQL 语句如下所示：

```
create table mygroup (
  group id number(4),
  job varchar2(10),
  name varchar2(10),
  salary number(10,2)
);
```

向 mygroup 表中插入测试数据，SQL 语句如下所示：

```
insert into mygroup values (10,'Coding',    'Bruce',1000);
insert into mygroup values (10,'Programmer','Clair',1000);
insert into mygroup values (10,'Architect', 'Gideon',1000);
insert into mygroup values (10,'Director',  'Hill',1000);

insert into mygroup values (20,'Coding',    'Jason',2000);
insert into mygroup values (20,'Programmer','Joey',2000);
insert into mygroup values (20,'Architect', 'Martin',2000);
insert into mygroup values (20,'Director',  'Michael',2000);

insert into mygroup values (30,'Coding',    'Rebecca',3000);
insert into mygroup values (30,'Programmer','Rex',3000);
insert into mygroup values (30,'Architect', 'Richard',3000);
insert into mygroup values (30,'Director',  'Sabrina',3000);

insert into mygroup values (40,'Coding',    'Samuel',4000);
insert into mygroup values (40,'Programmer','Susy',4000);
insert into mygroup values (40,'Architect', 'Tina',4000);
insert into mygroup values (40,'Director',  'Wendy',4000);

commit;
```

请统计各组工资的和以及工资总和，要求使用 rollup 函数完成。

5．分别按（group_id,job）、（group_id）、（job）以及全表统计工资和，要求使用 CUBE 函数来完成。

6．分别按（group_id）、（job）统计工资和，要求使用 GROUPING SETS 函数来完成。

Oracle 数据库基础

Unit05

知识体系.....Page 140

视图、序列、索引	视图	什么是视图
		视图的作用
		授权创建视图
		创建简单视图（单表）
		查询视图
		对视图进行 INSERT 操作
		创建具有 CHECK OPTION 约束的视图
		创建具有 READ ONLY 约束的视图
		通过查询 user_views 获取相关信息
		创建复杂视图（多表关联）
		删除视图
	序列	什么是序列
		创建序列
		使用序列
		删除序列
	索引	索引的原理
		创建索引
		创建基于函数的索引
		修改和删除索引
		合理使用索引提升查询效率
约束	约束概述	约束的作用
		约束的类型
	非空约束	建表时添加非空约束
		修改表时添加非空约束
		取消非空约束
	唯一性约束	什么是唯一性约束
		添加唯一性约束
	主键约束	主键的意义
		主键选取的原则

		添加主键约束
	外键约束	外键约束的意义
		添加外键约束
		外键约束对一致性的维护
		外键约束对性能的降低
		关联不一定需要外键约束
	检查约束	什么是检查约束
		添加检查约束

经典案例.....Page 158

Oracle 视图操作综合示例	什么是视图
	视图的作用
	授权创建视图
	创建简单视图（单表）
	查询视图
	对视图进行 INSERT 操作
	创建具有 CHECK OPTION 约束的视图
	创建具有 READ ONLY 约束的视图
	通过查询 user_views 获取相关信息
	创建复杂视图（多表关联）
	删除视图
通过序列实现自动生成主键	什么是序列
	创建序列
	使用序列
	删除序列
外键约束综合示例	外键约束的意义
	添加外键约束
	外键约束对一致性的维护
	外键约束对性能的降低
	关联不一定需要外键约束

课后作业.....Page 166

达内IT培训集团

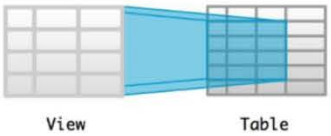
1. 视图、序列、索引

1.1. 视图

1.1.1. 【视图】什么是视图

什么是视图

- 视图(VIEW)也被称作虚表，即虚拟的表，是一组数据的逻辑表示
- 视图对应于一条SELECT语句，结果集被赋予一个名字，即视图名字
- 视图本身并不包含任何数据，它只包含映射到基表的一个查询语句，当基表数据发生变化，视图数据也随之变化



什么是视图（续1）

CREATE [OR REPLACE] VIEW view_name[(alias[, alias...])]
AS subquery；

- 视图创建后，可以像操作表一样操作视图，主要是查询
- Subquery是SELECT查询语句，对应的表被称作基表
- 根据视图所对应的子查询种类分为几种类型
  - SELECT语句是基于单表建立的，且不包含任何函数运算、表达式或分组函数，叫做**简单视图**，此时视图是基表的子集
  - SELECT语句同样是基于单表，但包含了单行函数、表达式、分组函数或GROUP BY子句，叫做**复杂视图**
  - SELECT语句是基于多个表的，叫做**连接视图**

1.1.2. 【视图】视图的作用

视图的作用

- 如果需要经常执行某项复杂查询，可以基于这个复杂查询建立视图，此后查询此视图即可，**简化复杂查询**
- 视图本质上就是一条SELECT语句，所以当访问视图时，只能访问到所对应的SELECT语句中涉及到的列，对基表中的其它列起到安全和保密的作用，**限制数据访问**

1.1.3. 【视图】授权创建视图

授权创建视图

- 创建视图的语句是CREATE VIEW
- 用户必须有CREATE VIEW系统权限，才能创建视图
- 如果没有权限，会提示：**权限不足**
- 管理员可以通过DCL语句授予用户创建视图的权限：

```
GRANT CREATE VIEW  TO tarena;
```

解说识知

权限赋予给谁

GRANT...TO...
DCL命令，
授予权限

授予
何种权限

1.1.4. 【视图】创建简单视图（单表）

创建简单视图（单表）

- 创建一个简单视图V_EMP_10来显示部门10中的员工的编码、姓名和薪水

```
CREATE VIEW v_emp_10
AS
SELECT empno, ename, sal, deptno
FROM emp
WHERE deptno = 10;
```

解说识知

自定义视图名称

查询语句

- 查看视图结构：

```
DESC v_emp_10;
```

创建简单视图（单表）(续1)

- 创建视图时，给列赋予别名
- 可以用OR REPLACE短语修改视图对应的SQL查询语句

```
CREATE OR REPLACE VIEW v_emp_10
AS
SELECT empno id, ename name, sal salary, deptno
FROM emp
WHERE deptno = 10;
```

解说识知

设置列的
别名

- 检查视图结构

```
DESC v_emp_10;
```

1.1.5. 【视图】查询视图

查询视图

- 查询视图和查询表的操作相同：

  SELECT * FROM v_emp_10;

- 此时视图的列名，和创建视图时的列名一致，不一定是原列名：

  SELECT id, name, salary FROM v_emp_10;

解说知识

1.1.6. 【视图】对视图进行 INSERT 操作

对视图进行INSERT操作

- 视图本身并不包含数据，只是基表数据的逻辑映射
- 当对视图执行DML操作时，实际上是对基表的DML操作
- 对视图执行DML操作的基本原则：
  - 简单视图能够执行DML操作，下列情况除外：在基表中定义了非空列，但简单视图对应的SELECT语句并没有包含这个非空列，导致这个非空列对视图不可见，这时无法对视图执行INSERT操作
  - 如果视图定义中包含了函数、表达式、分组语句、DISTINCT关键字或ROWNUM伪列，不允许执行DML操作
  - DML操作不能违反基表的约束条件

解说知识

对视图进行INSERT操作（续1）

- 对简单视图执行INSERT操作，成功插入数据到基表中

  INSERT INTO v_emp_10

  VALUES(1234, 'DOCTOR', 4000, 10);

- 简单视图可以通过DML操作影响到基表数据

解说知识

1.1.7. 【视图】创建具有 CHECK OPTION 约束的视图

创建具有CHECK OPTION约束的视图

```
CREATE [OR REPLACE] VIEW view_name[(alias[, alias...])]
AS subquery
[WITH CHECK OPTION];
```

新手识知

- WITH CHECK OPTION短语表示，通过视图所做的修改，必须在视图的可见范围内
- 假设INSERT，新增的记录在视图仍可查看
- 假设UPDATE，修改后的结果必须能通过视图查看到
- 假设DELETE，只能删除现有视图里能查到的记录

创建具有CHECK OPTION约束的视图（续1）

```
CREATE OR REPLACE VIEW v_emp_10
AS
SELECT empno id, ename name, sal salary, deptno
FROM emp
WHERE deptno = 10
WITH CHECK OPTION;
```

新手识知

- DML操作失败，部门20不在视图可见范围内
  - INSERT INTO v_emp_10
    VALUES(1008, 'donna' ,5500, 20);
  - UPDATE v_emp_10 SET deptno = 20
    WHERE id = 7782;

1.1.8. 【视图】创建具有 READ ONLY 约束的视图

创建具有READ ONLY约束的视图

- 对简单视图进行DML操作是合法的，但是不安全的
- 如果没有在视图上执行 DML 操作的必要，在建立视图时声明为**只读**来避免这种情况，保证视图对应的基表数据不会被非法修改

新手识知

```
CREATE [OR REPLACE] VIEW view_name[(alias[, alias...])]
AS subquery
[WITH READ ONLY];
```

创建具有READ ONLY约束的视图（续1）

解详识知

```
CREATE OR REPLACE VIEW v_emp_10
AS
SELECT empno, ename, sal, deptno FROM emp
WHERE deptno = 10
WITH READ ONLY;
```

- 对只读视图执行DML操作，失败

  INSERT INTO v_emp_10 VALUES(1258, 'DONNA', 3000, 10);

  ERROR 位于第 1 行:

  ORA-01733: 此处不允许虚拟列

  或：ORA-42399: 无法对只读视图执行 DML 操作

1.1.9. 【视图】通过查询 user_views 获取相关信息

通过查询user_views获取相关信息

解详识知

- 和视图相关的数据字典：
  - USER_OBJECTS
  - USER_VIEWS
  - USER_UPDATE_COLUMNS
- 在数据字典USER_OBJECTS中查询所有视图名称

  SELECT object_name FROM **user_objects**

  WHERE object_type = 'VIEW';

通过查询user_views获取相关信息（续1）

解详识知

- 在数据字典USER_VIEWS中查询指定视图

  SELECT text FROM **user_views**

  WHERE view_name = 'V_EMP_10';
- 在数据字典中USER_UPDATE_COLUMNS查询视图

  SELECT column_name, insertable, updatable, deletable

  FROM user_updatable_columns

  WHERE table_name = 'V_EMP_10';

1.1.10. 【视图】创建复杂视图（多表关联）

创建复杂视图（多表关联）

- 复杂视图指在子查询中包含了表达式、单行函数或分组函数的视图
- 必须为子查询中的表达式或函数定义别名
  - 创建一个视图V_EMP_SALARY，把职员表的数据按部门分组，获得每个部门的平均薪水、薪水总和、最高薪水和最低薪水

```
CREATE VIEW v_emp_salary
AS
SELECT d.dname, avg(e.sal) avg_sal, sum(e.sal) sum_sal,
max(e.sal) max_sal, min(e.sal) min_sal
FROM emp e join dept d
ON e.deptno = d.deptno
GROUP BY d.dname;
```

创建复杂视图（多表关联）（续1）

- 查询复杂视图

```
SELECT * FROM v_emp_salary;
```

- 复杂视图不允许DML操作

1.1.11. 【视图】删除视图

删除视图

- 当不再需要视图的定义，可以使用DROP VIEW语句删除视图

```
DROP VIEW view_name;
```

- 删除视图v_emp_10：

```
DROP VIEW v_emp_10;
```

- 视图虽然是存放在数据字典中的独立对象，但视图仅仅是基于表的一个查询定义，所以对视图的删除不会导致基表数据的丢失，不会影响基表数据

1.2. 序列

1.2.1. 【序列】什么是序列

什么是序列

解详识知

- 序列(SEQUENCE)是一种用来生成唯一数字值的数据库对象
- 序列的值由Oracle程序按递增或递减顺序自动生成，通常用来自动产生表的主键值，是一种高效率获得唯一键值的途径
- 序列是独立的数据库对象，和表是独立的对象，序列并不依附于表
- 通常情况下，一个序列为一个表提供主键值，但一个序列也可以为多个表提供主键值

1.2.2. 【序列】创建序列

创建序列

解详识知

```
CREATE SEQUENCE [schema.]sequence_name
[ START WITH i ] [ INCREMENT BY j ]
[ MAXVALUE m | NOMAXVALUE ]
[ MINVALUE n | NOMINVALUE ]
[ CYCLE | NOCYCLE ][ CACHE p | NOCACHE ]
```

- sequence_name是序列名，将创建在schema方案下
- 序列的第一个序列值是i，步进是j
- 如果j是正数，表示递增，如果是负数，表示递减

创建序列（续1）

解详识知

- 序列可生成的最大值是m，最小值是n
- 如果没有设置任何可选参数，序列的第一个值是1，步进是1
- CYCLE表示在递增至最大值或递减至最小值之后是否继续生成序列号，默认是NOCYCLE
- CACHE用来指定先预取p个数据在缓存中，以提高序列值的生成效率，默认是20

1.2.3. 【序列】使用序列

使用序列

- 创建一个序列，起始数据是100，步进是10
- 当序列被创建后，第一个序列值将是100，将要生成的序列号分别是110、120、130等

解说知识

```
CREATE SEQUENCE emp_seq
START WITH 100
INCREMENT BY 10;
```

使用序列（续1）

- 序列中有两个伪列
  - NEXTVAL：获取序列的下个值
  - CURRVAL：获取序列的当前值

解说知识

- 当序列创建以后，必须先执行一次NEXTVAL，之后才能使用CURRVAL
- 获取序列的第一个值，并且使用序列值为EMP表插入新的记录。

```
SELECT emp_seq.NEXTVAL FROM DUAL;
```

```
INSERT INTO emp(empno, ename)
        VALUES(emp_seq.NEXTVAL, 'donna');
```

使用序列（续2）

- 查询刚刚生成的记录，主键值将是110

```
SELECT empno, ename FROM emp
 WHERE ename = 'DONNA';
```

解说知识

- 此时查询序列的当前值，会得到110的数字。

```
SELECT emp_seq.CURRVAL FROM DUAL;
```

- 在序列的使用过程中，比如执行了一条SELECT emp_seq.NEXTVAL FROM DUAL语句，则浪费了一个序列值，会导致表的主键值不连续。而CURRVAL的使用不会导致序列值的递进

1.2.4. 【序列】删除序列

删除序列

- 删除序列的语法如下：
  - DROP SEQUENCE sequence_name;
- 删除序列emp_seq。
  - DROP SEQUENCE emp_seq;

解说识知

1.3. 索引

1.3.1. 【索引】索引的原理

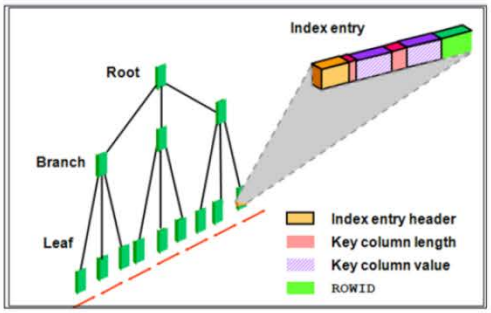
索引的原理

- 索引是一种允许直接访问数据表中某一数据行的树型结构，为了提高查询效率而引入，是独立于表的对象，可以存放在与表不同的表空间（TABLESPACE）中
- 索引记录中存有**索引关键字**和**指向表中数据的指针**（地址）
- 对索引进行的I/O操作比对表进行操作要少很多
- 索引一旦被建立就将被Oracle系统自动维护，查询语句中不用指定使用哪个索引
- 索引是一种提高查询效率的机制

解说识知

索引的原理(续1)

- Oracle B-tree索引的结构
- ROWID: 伪列，唯一标识一条数据记录，可理解为行地址



解说识知

1.3.2. 【索引】创建索引

创建索引

- 创建索引的语法：

CREATE [UNIQUE] INDEX index_name
ON table(column[, column...]);

解释说明

- index_name表示索引名称
- table表示表名
- column表示列名，可以建立单列索引或复合索引
- UNIQUE表示唯一索引

- 在EMP表的ENAME列上建立索引

```
CREATE INDEX idx_emp_ename ON emp(ename);
```

创建索引（续1）

- 复合索引也叫多列索引，是基于多个列的索引

解释说明

- 如果经常在ORDER BY子句中使用job和sal作为排序依据，可以建立复合索引：

```
CREATE INDEX idx_emp_job_sal ON emp(job, sal);
```

- 当做下面的查询时，会自动应用索引idx_emp_job_sal

```
SELECT empno, ename, sal, job FROM emp
ORDER BY job, sal;
```

1.3.3. 【索引】创建基于函数的索引

创建基于函数的索引

- 如果需要在emp表的ename列上执行大小写无关搜索，可以在此列上建立一个基于UPPER函数的索引:

解释说明

```
CREATE INDEX emp_ename_upper_idx
ON emp(UPPER(ename));
```

- 当做下面的查询时，会自动应用刚刚建立的索引:

```
SELECT * FROM emp
  WHERE UPPER(ename) = 'KING';
```

1.3.4. 【索引】修改和删除索引

修改和删除索引

- 如果经常在索引列上执行DML操作，需要定期重建索引，提高索引的空间利用率：

  ALTER INDEX index_name REBUILD;

- 重建索引idx_emp_ename

  ALTER INDEX idx_emp_ename **REBUILD**;

- 当一个表上有不合理的索引，会导致操作性能下降，删除索引的语法：

  DROP INDEX index_name;

- 删除索引idx_emp_ename

  DROP INDEX idx_emp_ename;

1.3.5. 【索引】合理使用索引提升查询效率

合理使用索引提升查询效率

- 为经常出现在WHERE子句中的列创建索引
- 为经常出现在ORDER BY、DISTINCT后面的字段建立索引。如果建立的是复合索引，索引的字段顺序要和这些关键字后面的字段顺序一致
- 为经常作为表的连接条件的列上创建索引
- 不要在经常做DML操作的表上建立索引
- 不要在小表上建立索引
- 限制表上的索引数目，索引并不是越多越好
- 删除很少被使用的、不合理的索引

2. 约束

2.1. 约束概述

2.1.1. 【约束概述】约束的作用

约束的作用

- 约束（CONSTRAINT）的全称是约束条件，也称作完整性约束条件
- 约束是在数据表上强制执行的一些数据校验规则，当执行DML操作时，数据必须符合这些规则，如果不符合则无法执行
- 约束条件可以保证表中数据的完整性，保证数据间的商业逻辑

2.1.2. 【约束概述】约束的类型

约束的类型

- 约束条件包括：
  - 非空约束(Not Null)，简称NN
  - 唯一性约束(Unique)，简称UK
  - 主键约束(Primary Key)，简称PK
  - 外键约束(Foreign Key)，简称FK
  - 检查约束(Check)，简称CK

解说识知

2.2. 非空约束

2.2.1. 【非空约束】建表时添加非空约束

建表时添加非空约束

- 非空约束用于确保字段值不为空
- 默认情况下，任何列都允许有空值，但业务逻辑可能会要求某些列不能取空值
- 当某个字段被设置了非空约束条件，这个字段中必须存在有效值，即：
  - 当执行INSERT操作时，必须提供这个列的数据
  - 当执行UPDATE操作时，不能给这个列的值设置为NULL

解说识知

建表时添加非空约束（续1）

```
CREATE TABLE employees (
eid NUMBER(6),
name VARCHAR2(30) NOT NULL,
salary NUMBER(7, 2),
hiredate DATE
    CONSTRAINT employees_hiredate_nn NOT
  NULL
);
```

解说识知

2.2.2. 【非空约束】修改表时添加非空约束

修改表时添加非空约束

- 可以在建表之后，通过修改表的定义，添加非空约束：

```
ALTER TABLE employees
MODIFY (eid NUMBER(6) NOT NULL);
```

2.2.3. 【非空约束】取消非空约束

取消非空约束

- 如果业务要求取消某列的非空约束，可以采用重建表或者修改表的方式：

```
ALTER TABLE employees
MODIFY (eid NUMBER(6) null);
```

2.3. 唯一性约束

2.3.1. 【唯一性约束】什么是唯一性约束

什么是唯一性约束

- 唯一性(Unique)约束条件用于保证字段或者字段的组合不出现重复值
- 当给表的某个列定义了唯一约束条件，该列的值不允许重复，但允许是NULL值
- 唯一性约束条件可以在建表同时建立，也可以在建表以后再建立

2.3.2. 【唯一性约束】添加唯一性约束

添加唯一性约束

- 在建表employees的同时，在eid、email列上创建唯一约束条件，并在建表后在name列上建立一个名为employees_name_uk的唯一约束条件

解释说明

```
DROP TABLE employees；--将表删掉重新创建
CREATE TABLE employees（
eid NUMBER(6) UNIQUE,
name VARCHAR2(30),
email VARCHAR2(50),
salary NUMBER(7, 2),
hiredate DATE,
CONSTRAINT employees_email_uk UNIQUE(email)
);
```

添加唯一性约束（续1）

- 在建表之后增加唯一性约束条件：

```
ALTER TABLE employees
ADD CONSTRAINT employees_name_uk
  UNIQUE(name);
```

解释说明

2.4. 主键约束

2.4.1. 【主键约束】主键的意义

主键的意义

- 主键(Primary Key)约束条件从功能上看相当于非空（NUT NULL）且唯一（UNIQUE）的组合
- 主键字段可以是单字段或多字段组合，即：在主键约束下的单字段或者多字段组合上不允许有空值，也不允许有重复值
- 主键可以用来在表中唯一的确定一行数据
- 一个表上只允许建立一个主键，而其它约束条件则没有明确的个数限制

解释说明

2.4.2. 【主键约束】主键选取的原则

主键选取的原则

- 主键应是对系统无意义的数据
- 永远也不要更新主键，让主键除了唯一标识一行之外，再无其他的用途
- 主键不应包含动态变化的数据，如时间戳
- 主键应自动生成，不要人为干预，以免使它带有除了唯一标识一行以外的意义
- 主键尽量建立在单列上

2.4.3. 【主键约束】添加主键约束

添加主键约束

- 在建表时添加主键约束条件：

```
CREATE TABLE employees2 (
eid NUMBER(6) PRIMARY KEY,
name VARCHAR2(30),
email VARCHAR2(50),
salary NUMBER(7, 2),
hiredate DATE
);
```

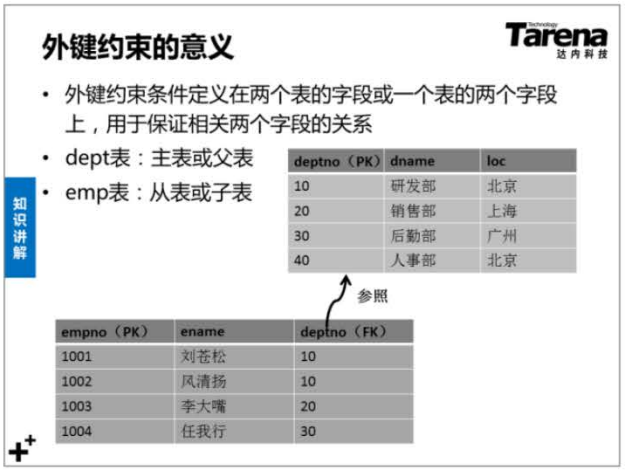
添加主键约束（续1）

- 建表后创建主键约束条件，并自定义约束条件名称

```
CREATE TABLE employees3 (
eid NUMBER(6),
name VARCHAR2(30),
email VARCHAR2(50),
salary NUMBER(7, 2),
hiredate DATE
);
ALTER TABLE employees3
ADD CONSTRAINT
        employees3_eid_pk PRIMARY KEY (eid);
```

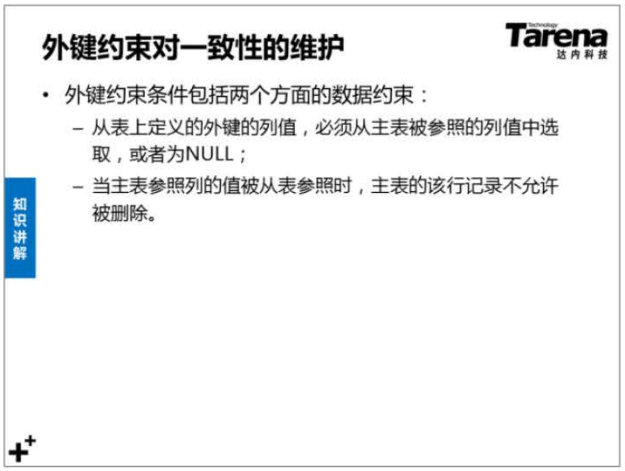
2.5. 外键约束

2.5.1. 【外键约束】外键约束的意义



2.5.2. 【外键约束】添加外键约束

2.5.3. 【外键约束】外键约束对一致性的维护



外键约束对一致性的维护（续1）

解详识知

```
INSERT INTO employees4(eid, name, deptno)
  VALUES(1234, 'rose tyler', 40);--成功

INSERT INTO employees4(eid, name, deptno)
  VALUES(1235, 'martha jones', NULL); --成功

INSERT INTO  employees4(eid, name, deptno)
  VALUES(1236, 'donna noble', 50);
  --失败，不存在部门50

DELETE FROM dept WHERE deptno  = 40;
  --失败，40被参照，不允许删除
```

2.5.4. 【外键约束】外键约束对性能的降低

外键约束对性能的降低

解详识知

- 如果在一个频繁DML操作的表上建立外键，每次DML操作，都将导致数据库自动对外键所关联的对应表做检查，产生开销，如果已在程序中控制逻辑，这些判断将增加额外负担，可以省去
- 外键确定了主从表的先后生成关系，有时会影响业务逻辑

2.5.5. 【外键约束】关联不一定需要外键约束

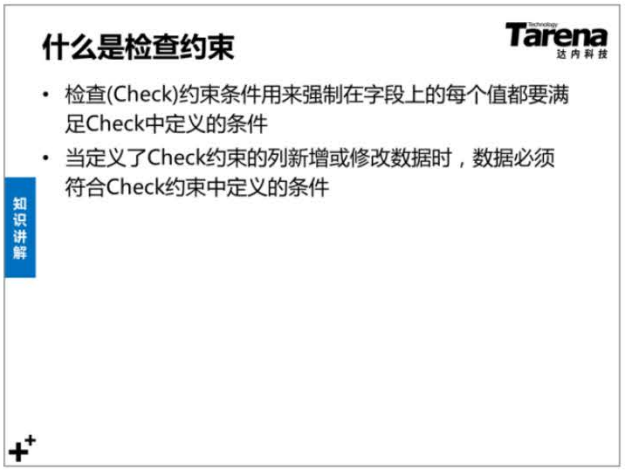
关联不一定需要外键约束

解详识知

- 保证数据完整性可由程序或触发器控制
- 简化开发，维护数据时不用考虑外键约束
- 大量数据DML操作时不需考虑外键耗费时间

2.6. 检查约束

2.6.1. 【检查约束】什么是检查约束



2.6.2. 【检查约束】添加检查约束

经典案例

1. Oracle 视图操作综合示例

- **问题**

有学员表 student，表结构如表 - 1 所示：

表 - 1 学员表 student 信息

字段名	类型	描述
sid	NUMBER(4,0)	学生编码
sname	VARCHAR2(20)	学生姓名
score	NUMBER(4,1)	学生成绩
class_id	NUMBER(4,0)	所在专业

学员表 student 的示例数据如图-1 所示：

SID	SNAME	SCORE	CLASS_ID
1001	allen	89.0	101
1002	marry	80.0	101
1003	jerry	81.0	101
1004	tom	74.0	90
1005	jim	67.0	90
1234	rose	90.0	

图-1

请根据上述表结构和示例数据，完成如下各个案例。

1）基于student表创建视图v_student_101，该视图包含student表的数据的子集，即所有专业为 101 的学生的数据为视图 v_student_101 的数据。

2）查询视图 v_student_101 中的所有数据。

3）对视图 v_student_101 执行插入操作，插入的数据信息为 sid、sname、score 的数据分别为 1234、rose、90，并检查插入操作的结果。

4）基于 student 表创建了一个视图 v_student，查询学生的平均成绩，该视图的数据显示如图-2 所示：

CLASS_ID	AVG_SCORE
	90
90	70.5
101	83.3333333333333

图-2

5）在视图 v_student 中，执行如下 SQL 语句：

```
update v_student set avg_score =80 where class_id=101;
```

执行上述 SQL 语句，将产生的结果是什么？

6）定义视图 v_stu，在视图 v_student 现有列的基础上，加入人数一列,并查询。

7）从 v_stu 视图中查找数据的 SQL 语句为：

```
SELECT * FORM View v_student;
```

上述查询视图的 SQL 语句是否正确？为什么？

8）删除视图 v_student 以及 v_stu。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：创建视图 v_student_101

使用 create view 语句基于 student 表创建名为 v_student_101 的视图，该视图包含 student 表的数据的子集，即所有专业为 101 的学生的数据为视图 v_student_101 的数据，SQL 语句如下所示：

```
CREATE VIEW v student 101
 AS
 SELECT sid,  sname, score
 FROM student
where class_id = 101;
```

步骤二：查询视图中的数据

使用 select 语句查询视图 v_student_101 中的所有数据，SQL 语句如下所示：

```
select * from v_student_101;
```

查询的结果如图-3 所示：

SID	SNAME	SCORE
1001	allen	89.0
1002	marry	80.0
1003	jerry	81.0

图-3

从查询结果中的数据可以看出，只包含所在专业为 101 的学员信息。

步骤三：对视图执行插入操作

使用 insert 语句对视图 v_student_101 执行插入操作，插入的数据信息为 sid、sname、score 的数据分别为 1234、rose、90，SQL 语句如下所示：

```
insert into v_student_101 values(1234, 'rose' ,90);
```

```
commit;
```

使用如下 SQL 语句查询 v_student_101 视图的数据：

```
select * from v_student_101;
```

会发现查询不到刚插入的数据。接着，使用下列 SQL 语句去查询表 student：

```
select * from student;
```

会发现 student 表多了一条记录，即为刚刚插入的那条数据。视图本身并不包含数据，只是基表数据的逻辑映射。当对视图(v_student_101)执行 DML 操作时，实际上是对基表（student）的 DML 操作。

步骤四：创建复杂视图 v_student

使用 create view 语句基于 student 表创建了一个视图 v_student，查询学生的平均成绩，SQL 语句如下所示：

```
CREATE VIEW v student
AS
SELECT class id, avg(score) avg score
FROM student
GROUP BY class_id;
```

复杂视图指在子查询中包含了表达式、单行函数或分组函数的视图，上述视图属于负载视图。复杂视图必须为子查询中的表达式或函数定义别名。

步骤五：对复杂视图 v_student 执行 DML 操作

在视图 v_student 中，执行如下 SQL 语句：

```
update v_student set avg_score =80 where class_id=101;
```

执行上述 SQL 语句，产生的结果如图-4 所示：

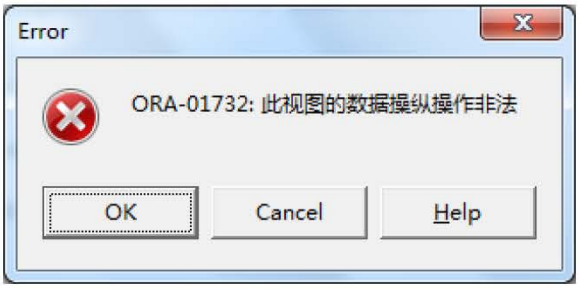


图-4

从图-4 可以看出复杂视图不允许 DML 操作。

步骤六：创建复杂视图 v_stu

使用 create view 语句创建复杂视图 v_stu，该视图在 v_student 视图现有列的基础上，加入人数一列，SQL 语句如下所示：

```
CREATE VIEW v_stu
 AS
 SELECT class_id, avg(score) avg_score, count(*) cnt
 FROM student
 GROUP BY class_id;
```

查询该视图的 SQL 语句如下所示：

```
select * from v_stu;
```

查询结果如图-5 所示：

CLASS_ID	AVG_SCORE	CNT
	90	1
90	70.5	2
101	83.3333333333333	3

图-5

步骤七：查询视图

从 v_student 视图中查找数据的 SQL 语句为：

```
SELECT * FORM View v_student;
```

上述查询视图的 SQL 语句是不正确的。查询视图和查询表的 SQL 语句类似，只需将表名的位置更换为视图名字即可，正确的写法为：

```
SELECT * FORM v_student;
```

步骤八：删除视图 v_student 以及 v_stu

使用 drop view 语句删除视图 v_student 以及 v_stu，SQL 语句如下所示：

```
DROP VIEW v_student;
DROP VIEW v_stu;
```

执行上述 SQL 语句将视图 v_student 以及 v_stu 删除掉，但是创建这两个视图的基表 student 的数据并不受影响。这是因为，视图虽然是存放在数据字典中的独立对象，但视图仅仅是基于表的一个查询定义，所以对视图的删除不会导致基表数据的丢失，不会影响基表数据。

2. 通过序列实现自动生成主键

- **问题**

本案例的详细要求如下：

1）创建序列 emp_seq，该序列的起始值为 100，步进为 10。

```
CREATE SEQUENCE emp_seq
START WITH 100
INCREMENT BY 10;
```

2）查询序列的下一个值。

```
SELECT emp_seq.NEXTVAL FROM DUAL;
```

3）利用序列 emp_seq 为 emp 表生成主键。向 emp 表插入的数据为 empno 由序列生成、ename 为“donna”。

```
INSERT INTO emp(empno, ename)
VALUES(emp_seq.NEXTVAL, 'donna');
```

4）查询插入的数据是否插入成功。

```
SELECT empno, ename FROM emp WHERE ename = 'donna';
```

5）查询序列 emp_seq 的当前值。

```
SELECT emp_seq.CURRVAL FROM DUAL;
```

6）再次利用序列 emp_seq 为 EMP 表生成主键，数据为 empno 由序列生成、ename 为“donna”，SQL 语句如下所示：

```
INSERT INTO emp(empno, ename)
VALUES(emp_seq.NEXTVAL, 'donna');
```

7）查询上一步骤中插入的数据是否插入成功。

```
SELECT empno, ename FROM emp;
```

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：创建序列 emp_seq

使用 create sequence 创建序列 emp_seq，该序列的起始值为 100，步进为 10,SQL 语句如下所示：

```
CREATE SEQUENCE emp_seq
START WITH 100
INCREMENT BY 10;
```

步骤二：查询序列的下一个值

使用查询 NEXTVAL 获取序列 emp_seq 的下一个值，SQL 语句如下所示：

```
SELECT emp_seq.NEXTVAL FROM DUAL;
```

步骤三：使用序列作为表的主键

利用序列 emp_seq 为 emp 表生成主键，向 emp 表插入的为 empno 由序列生成、ename 为“donna”，SQL 语句如下所示：

```
INSERT INTO emp(empno, ename)
VALUES(emp_seq.NEXTVAL, 'donna');
```

步骤四：查询插入的数据是否插入成功

查询上一步骤中向 emp 表插入的数据是否插入成功，SQL 语句如下所示：

```
SELECT empno, ename FROM emp WHERE ename = 'donna';
```

执行上述 SQL 语句 emp 表的数据如图-6 所示：

EMPNO	ENAME
110	donna

图-6

从图-6 可以看出该记录中的 empno 列的数据为 110，在之前 100 的基础上步进了 10。

步骤五：查询序列的当前值

查询序列 emp_seq 的当前值,SQL 语句如下所示：

```
SELECT emp_seq.CURRVAL FROM DUAL;
```

查询后，序列的当前值为 110。

步骤六：再次利用序列作为主键插入数据

再次利用序列 emp_seq 为 EMP 表生成主键，数据为 empno 由序列生成、ename 为“donna”，SQL 语句如下所示：

```
INSERT INTO emp(empno, ename)
VALUES(emp_seq.NEXTVAL, 'donna');
```

步骤七：查询上一步骤中插入的数据是否插入成功

```
SELECT empno, ename FROM emp;
```

执行上述 SQL 语句，查询的结果如图-7 所示：

EMPNO	ENAME
7369	SMITH
7499	ALLEN
7521	WARD
7566	JONES
7654	MARTIN
7698	BLAKE
7782	CLARK
7788	SCOTT
7839	KING
7844	TURNER
7876	ADAMS
7900	JAMES
7902	FORD
7934	MILLER
110	donna
120	donna

图-7

从图-7 的查询结果可以看出，序列的当前值为 120。

3. 外键约束综合示例

- **问题**

本案例的详细要求如下：

1）请看如下 SQL 语句：

```
delete product where p_id = 1011;
```

执行上述 SQL 语句，提示错误“已找到子记录日志”，错误的原因什么？

2）student 表结构如表-2 所示：

表 - 2 学员表 student 信息

字段名	类型	描述
Id	NUMBER(4)	学员 ID
Name	VARCHAR2(25)	学员姓名
Score	NUMBER(7,2)	学员分数
Birth	VARCHAR2(9)	职位
graduate_time	NUMBER(4)	毕业时间

student 表的如图-8 所示：

ID	NAME	SCORE	BIRTH	GRADUATE_TIME
1011	marry	100.00	07-1998	1002
1012	jerry	78	07-1997	1001

图-8

GraduateTime 表结构如表-3 所示：

表 - 3 GraduateTime 表信息

字段名	类型	描述
Id	NUMBER(4)	学员 ID（PK）
Name	VARCHAR2(25)	学员姓名

GraduateTime 表的数据如图-9 所示：

ID	NAME
1001	08-2007
1002	08-2008
1003	08-2009
1004	08-2010

图-9

其中，student 表和 GraduateTime 表存在外键关系，即毕业时间 graduate_time 列作为外键（Foreign Key）关联到 GraduateTime 表的主键列（id）。

下列 SQL 语句会引发异常“ORA-02291：违反完整约束条件 - 未找到父项关键字”的是：( )。

A . update student set graduate_time =1003 where id=1011;

B . update student set name= 'smith', graduate_time = 1005 where id=1011;

C . update student set name= 'smith', graduate_time = 1004 where id=1011;

D . update student set id=null, score= 85 where graduate_time = 1005;

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一： 问题一解答

本题中提示错误“已找到子记录日志“说明 product 表中，编码为 1011 的商品被其它表中的数据参照，该表中的该行记录不允许删除。如果解除外键关联关系，再实施删除则会成功。

步骤二：问题二解答

问题二的正确答案为 B。外键约束条件包括两个方面的数据约束：

1）从表上定义的外键的列值，必须从主表被参照的列值中选取，或者为 NULL；

2）当主表参照列的值被从表参照时，主表的该行记录不允许被删除。

本题属于上述描述中的第一条。B 选项中的更新语句，欲将学员 ID 为 1011 的毕业时间（graduate_time）更新为 1005，列 graduate_time 为外键，其数据要参照主表（GraduateTime）id 列的数据，由于在 GraduateTime 表不存在 id 列为 1005 的数据，因此会引发异常“ORA-02291：违反完整约束条件 - 未找到父项关键字”。

课后作业

1. 简述视图的意义

2. 创建一个视图 v_emp_salesman，内容是所有职位是 SALESMAN 的员工

3. 下列语句运行的结果是

```
insert into v_emp_salesman values(1234, 'tina', 3500, 20);
```

4. 下列关于视图说法正确的是

A．对复杂视图可以执行插入数据操作

B．为了禁止在视图上执行 DML 操作，可以在建立视图时设置 READ ONLY 约束

C． CHECK OPTION 约束表示，通过视图所做的修改，必须在视图的可见范围内

D．对视图的操作,最终都会转化成对基本表的操作

5. 创建一个视图 v_mgr_salary，列出每个主管的名字，以及他的下属中的最高薪水和最低薪水。

视图中数据显示的形式如图-1 所示：

ENAME	MAX_SAL	MIN_SAL
JONES	3000	3000
FORD	800	800
CLARK	1300	1300
SCOTT	1100	1100
BLAKE	1600	950
KING	2975	2450

图-1

6. 关于序列的说法，下列正确的是

A．序列是 Oracle 提供的可用于产生一系列唯一数字的数据库对象 。

B．通常情况下，一个序列为一张表提供主键值，但一个序列也可以为多个表提供主键值。

C．使用序列时，伪列 NEXTVAL 返回序列生成的下一个值。

D．任何时候都可以使用伪列 CURRVAL 返回当前序列值。

7. 创建序列 stu_seq，然后，对 student 表进行插入，插入数据时，使用该序列生成主键

有学员表 student，表结构如表 - 1 所示：

表 - 1 学员表 student 信息

字段名	类型	描述
id (PK)	NUMBER(4)	学生编码
name (not null)	VARCHAR2(20)	学生姓名
Birthday	DATE	学生生日
Score	NUMBER(9,2)	学生总成绩

1. 创建一个序列 stu_seq，初始值是 1000，步进是 2。

2. 构造 SQL 语句，向 student 数据表中插入一条记录，其信息如下：学号使用上一步创建的序列生成，学生姓名为张无忌，生日为 1987-11-17，总成绩为 639 分。

8. 简述索引的原理及创建索引的意义

9. 需要经常在 emp 表的 ename 列上执行大小写无关搜索，在此列上建立一个基于 UPPER 函数的索引

10. 列举需要创建索引以及不适合创建索引的场合

11. 简述主键选取的原则

12. 已知 student 表、course 表和 sc 表，为 sc 表添加外键约束

已知学生表 student，该表两个字段学号（id）、姓名（name），其中 id 列为主键列，创建该表的 SQL 语句如下所示：

```
create table student(
  id number(10) primary key,
  name varchar2(20)
 );
```

有课程表 course ，该表有两个字段编号（id）、课程名称（name），其中 id 列为主键列，创建该表的 SQL 语句如下所示：

```
create table course(
 id number(10) primary key,
 name varchar2(20)
);
```

有选课表 sc，该表有三个字段学号（sid）、课程编号（cid）、成绩（score），创建该表的 SQL 语句如下所示：

```
create table sc(
 sid number(10),
 cid number(10),
 score number(5,2)
);
```

本案例要求建立下列外键关联关系：

1.sc 表的学号列（sid），外键关联学生表（student）的学号列（id）。

2.sc 表的课程编号列（cid），外键关联课程表（course）的编号列（id）。

请写出建立上述关联关系的 SQL 语句。

13. 简述外键约束的意义

14. 简述外键约束对性能的影响，以及应对策略