《数据库原理与应用 A》
基础训练实验指导书

学　　期：2021 年春季

指导教师：闫蓉

联系方式：csyanr@imu.edu.cn

日　　期：2021 年 2 月 18 日

前言

随着计算机技术与网络通信技术的发展，数据库技术已成为信息社会中对大量数据进行组织与管理的重要技术手段及软件技术，是网络信息化管理系统的基础。它不仅有完整的理论基础，而且随着硬件技术与软件技术的飞快发展，它的应用也越来越广泛。

本实验指导书通过大量的实例，循序渐进地引导学生做好各章的实验。根据实验教学大纲，编排了十一个实验，五个作业和一次上机考试，学时数 36 学时。在实验之前，由教师对实验作一定的讲解后，让学生明确实验目的，并对实验作好预习工作。在实验中，学生根据实验指导中的内容进行验证与总结，然后再去完成实验步骤中安排的任务。实验完成后，学生按要求完成十一个实验的实验报告，并将五个作业提交到服务器中。整个教学和实验中，我们推荐采用 Oracle 数据库管理系统作为实验环境，强调学生切实培养动手实践能力，掌握数据库的建立、维护和使用的方法。

第一部分　实验前期准备和说明

《数据库原理与应用 A》是计算机科学与技术专业、软件工程专业及相近各专业的一门重要基础理论课程。课程概括性的介绍了数据库技术发展历程，详细介绍了数据库系统的基本概念、基本原理、基本方法，阐述了数据库设计、实现的基本过程，同时也介绍了数据库系统的最新进展。通过本课程的学习，使学生牢固掌握数据库系统的基本概念和基本原理，熟悉数据库系统的主要实现方法，能够根据应用需要分析设计实用的数据库管理系统。

《数据库原理与应用 A》是一门实践性很强的课程，通过实验实践，配合课堂教学内容加深对数据库系统的基本概念、基本原理、基本方法的理解，掌握数据库系统设计的基本思想和基本步骤，熟悉关系数据库的标准语言 SQL，并对关系数据库系统的某一典型系统，有较深入的了解，使学生对数据库系统有一个完整的、全面的认识。

1.课程实验的目标：

(1)通过上机操作，加深对数据库系统理论知识的理解。

(2)通过使用具体的数据库管理系统，了解一种实际的数据库管理系统，并掌握其操作技术。

(3)通过实验题目的上机实践，提高学生的实践动手能力，提高学生的分析问题和解决问题的能力，熟悉数据库应用系统开发过程。

2.主要软件实验环境：

访问端操作系统：不限

访问端工具：浏览器或第 3 方通用连接数据库工具

服务端操作系统：Windows 2008 64 位版本

服务端数据库管理系统：Oracle10g 及以上版本。

3、课程实验内容

本实验指导书共有 11 个实验，5 个作业。

4.课程实验的基本要求:

（1）每次实验前，教师需要向学生讲清楚本次实验的目的和基本要求；学生应当先弄清楚相关的理论知识，预习实验内容、方法和步骤，认真准备好实验

程序和数据，避免出现盲目上机的行为。

(2) **实验 1 人 1 组，在规定的时间内，由学生独立完成**。要求学生本人按本实验指导内容，在每次实验结束后撰写实验报告（**命名方式：学号-实验-实验序号.doc，如 2001-实验-1.doc**）并提交到知到平台，并在规定时间内将 5 个作业（**命名方式：学号-作业号.sql，如 2001-1.sql**）提交到知到平台。

5.检查方式

(1)计划实验学时 36 学时，每次作业实验结束后由任课老师检查。

(2) 学生按照相应实验要求在规定时间内完成实验，并提交相关实验的实验报告。**逾期不交的同学，该实验成绩按零分计。**（每次实验结束后两天内提交电子实验报告）

(3) 要求学生在规定时间内完成作业，并提交相关作业。由任课教师检查并给分。**逾期不交的同学，该作业成绩按零分计。**（每次作业结束后两天内提交电子作业）

(4)**严禁拷贝，发现该项实验或作业计零分。**

注：可能有部分实验上机时因无权限而无法使用，请课下自行练习。

6.评分标准

打分方式及标准：

序号	题目	周次号	比重（%）	评分原则				
				不给分	及格	中等	良好	优秀
1	实验一	5-1	0.5	没有	不完整	基本正确	正确	正确，清晰
2	实验二	5-2	0.5	没有	不完整	基本正确	正确	正确，清晰
3	实验三	6-1	0.5	没有	不完整	基本正确	正确	正确，清晰
4	作业一	6-2，7-1	1.5	没有	不完整	基本正确	正确	正确，清晰
5	实验四	7-2	0.5	没有	不完整	基本正确	正确	正确，清晰
6	实验五	8-1	0.5	没有	不完整	基本正确	正确	正确，清晰
7	实验六	8-2	0.5	没有	不完整	基本正确	正确	正确，清晰
8	作业二	9-1，9-2	2.5	没有	不完整	基本正确	正确	正确，清晰
9	实验七	10-1	0.5	没有	不完整	基本正确	正确	正确，清晰
10	作业三	10-2	1.5	没有	不完整	基本正确	正确	正确，清晰
11	实验八	9	0.5	没有	不完整	基本正确	正确	正确，清晰
12	作业四	9	2.5	没有	不完整	基本正确	正确	正确，清晰
13	实验九	9	0.5	没有	不完整	基本正确	正确	正确，清晰
14	作业五	10	1.5	没有	不完整	基本正确	正确	正确，清晰
15	实验十	11	0.5	没有	不完整	基本正确	正确	正确，清晰
16	实验十一	11	0.5	没有	不完整	基本正确	正确	正确，清晰

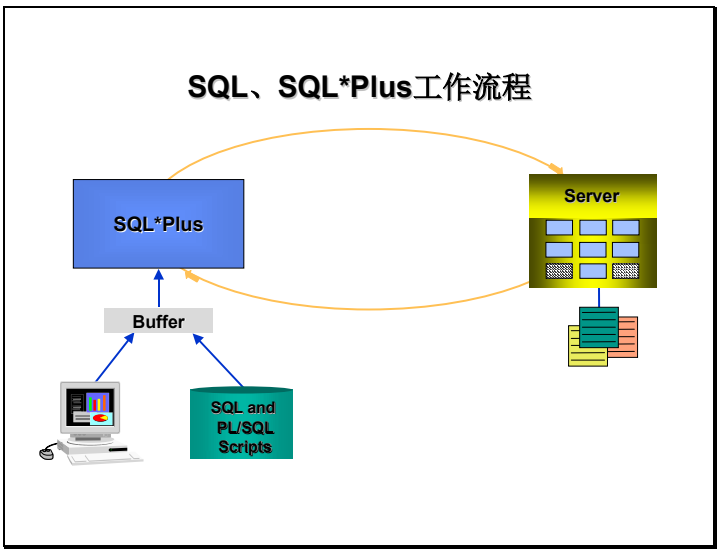
17	上机考试	12	5					
总计		12	20	包括是否按时完成，出勤情况等				
注：具体实验及作业完成以任课教师通知时间完成								

第二部分 基础实践内容

SQL*PLUS 工具软件的使用

SQL*PLUS 是 ORACLE 数据库管理员和普通用户最常用的实用程序之一，它提供一个交互式 SQL 语句、PL/SQL 语句块和 SQL*PLUS 命令的编辑、编译和执行环境。普通用户使用它可以实现各种数据库操作，数据库管理员使用它除能够实现基本的数据库操作之外，还能够完成数据库启动、关闭和恢复等管理工作。

SQL、SQL*PLUS 的工作流程



SQL*PLUS 软件的主要功能包括：

- 登录；
- 描述表结构；
- 执行 SQL 命令；
- 编辑 SQL 命令；
- 将 SQL 命令存入文件；
- 执行文件中的 SQL 命令；
- 将文件中的 SQL 命令调入缓存区；
- 格式化输出结果。

登陆；

- 图形方式：
  点击 SQL*PLUS 图标，输入用户名，口令和主机字符串即可。
- 命令行格式：
  SQLPLUS username/password@’202.207.12.213:1522/rj2’
  其中：网络服务名包括：主机名称（IP 地址），网络访问协议，数据库名称。

SQLPLUS scott/tiger@’202.207.12.213:1522/rj2’

- 描述表结构
  SELECT * FROM cat；显示用户可存取的所有基表清单。
  DESC emp； 描述基表 EMP 的结构。

- 执行 SQL 命令
  SELECT * FROM emp；显示基表 EMP 的所有数据记录。

- 编辑 SQL 命令
  EDIT；编辑存储在 SQL*PLUS 缓存区的内容，即最后一条 SQL 命令。

- 将 SQL 命令存入文件
  格式：SAVE filename
  SAVE c:\select_emp.sql; 将存储在 SQL*PLUS 缓存区的内容，即最后一条 SQL 命令存储为磁盘文件。

- 将文件中的 SQL 命令调入缓存区
  格式：EDIT filename
  EDIT c:\select_emp.sql

- 执行文件中的 SQL 命令
  格式：START filename 或 @filename
  START c:\select_emp.sql
  @c:\select_emp.sql

- 格式化输出结果。
  设置列标题
  COL ename HEADING ‘Employee|Name’ FORMAT A10;
  COL sal JUSTFY LEFT FORMAT $99,999.00;
  COL hiredate FORMAT A8 NULL ‘not hired’;
  COL comm FORMAT $99,999.00 NULL ‘No’

  字符和日期
  A*n* 设置显示的宽度（n）.
  数字
  9 Single zero-suppression digit.
  0 强制为 0.
  $ 美元标识.
  L 当前的货币符号.
  . 小数点位.
  , 千位分隔符.

- 显示列的格式设置

COL ename；

- 清除列的格式设置
  COL ename CLEAR；

- 将输出结果存储为磁盘文件。
  格式：SPOOL [filename [CRE[ATE] | REP[LACE]| APP[END]]
  SPOOL OFF
  SPOOL c:\emp.txt
  SELECT * FROM emp ORDER BY deptno;
  SPOOL OFF

另外，ORACLE 的 SQL*PLUS 工具含有帮助功能：
SQL〉HELP；
SQL〉HELP INDEX；显示所有 SQL*PLUS 可用命令。
SQL〉HELP STARTUP；显示 STARTUP 的功能及用法。

SQL*PLUS 系统常用命令介绍：
假设当前执行命令为：SELECT * FROM tab;

- (A)PPEND 添加文本到缓冲区当前行尾

SQL> A order by tname
结果：SELECT * FROM tab ORDER BY tname; （注：A 后面跟 2 个空格）

- (C)HANGE/old/new 在当前行用新的文本替换旧的文本

SQL> C/*/tname
结果：SELECT tname FROM tab;

- (C)HANGE/text 从当前行删除文本

SQL> C/tab 结果：SELECT tname FROM;

- DEL 删除当前行
- DEL n 删除第 n 行
- (I)NPUT 文本 在当前行之后添加一行
- (L)IST 显示缓冲区中所有行
- (L)IST n 显示缓冲区中第 n 行
- (L)IST m n 显示缓冲区中 m 到 n 行
- RUN 执行当前缓冲区的命令
- / 执行当前缓冲区的命令
- R 执行当前缓冲区的命令
- @文件名 运行调入内存的 sql 文件，如：

- EDIT s<回车>

如果当前目录下不存在 s.sql 文件，则系统自动生成 s.sql 文件，

在其中输入“select * from tab;”，存盘退出。

- @s<回车>

系统会自动查询当前用户下的所有表、视图、同义词。

- @@文件名　　在.sql 文件中调用另一个.sql 文件时使用
- EXIT　　退出 SQL*PLUS
- DESC 表名　　显示表的结构
- CLEAR SCREEN　清空当前屏幕显示

SQL*PLUS 系统环境变量

SHOW 和 SET 命令是两条用于维护 SQL*PLUS 系统变量的命令

SHOW ALL	查看所有 68 个系统变量值
SHOW USER	显示当前连接用户
SHOW ERROR	显示错误信息
SET HEADING OFF	禁止输出列标题，默认值为 ON
SET FEEDBACK OFF	禁止显示最后一行的计数反馈信息，默认值为"对 6 个或更多的记录，回送 ON"
SET TIMING ON	默认为 OFF，设置查询耗时，可用来估计 SQL 语句的执行时间，测试性能
SET SQLPROMPT "SQL> "	设置默认提示符，默认值就是"SQL> "
SET LINESIZE 1000	设置屏幕显示行宽，默认 80
SET AUTOCOMMIT ON	设置是否自动提交，默认为 OFF
SET PAUSE ON	默认为 OFF，设置暂停，会使屏幕显示停止，等待按下 ENTER 键再显示下一页
SET ARRAYSIZE 1	默认为 15
SET LONG 1000	默认为 80

实验一　　表的定义

实验保存为学号-实验-1.doc 文件提交。

【 实验目标】

- 创建一个包含完整性约束的表；
- 确定表命名的规则；
- 描述可以使用的数据类型；
- 利用约束自动创建索引；
- 利用其它表的数据行创建新表。

数据结构

- ORACLE 数据库可以创建多种对象；
  1. 表（Table） 存储数据
  2. 视图（View） 一个或多个表的数据的子集
  3. 序列（Sequence）产生唯一值，用于流水号
  4. 索引（Index） 提高查询的性能
- 在数据库设计阶段定义数据结构。

表结构

- 可在任意时刻创建表
- 不需预先定义表的大小
- 可以在线修改

语法格式：

CREATE TABLE [*schema.*]*table*
(*column datatype* [DEFAULT *expr*]
[*column_constraint*],
...
[*table_constraint*]);

创建者必须具有特定的权限：

- CREATE TABLE
- 存储区域

与其他用户表的关系

- 约束条件只能参照本数据库中的其他表.
- 其他用户的表不会出现在本用户的模式中
- 应该在表名前使用用户名作为前缀

缺省值

- 插入操作时，为某列指定缺省值
- 插入操作中，合法值为数字、表达式和SQL系统变量如：SYSDATE、USER
- 非法值为其他列名
- 缺省值的数据类型必须同列的数据类型相匹配

命名规则

- 以字母开始
- 包含 1-30 个字符长
- Z, a–z, 0–9, _, $, #
- 不能与本用户的其他对象同名
- 不能是保留字

数据类型

数据类型	说　　明
VARCHAR2(size)	变长变量
CHAR(size)	定长变量
NUMBER(有效数字数，小数点左或右保留位数)	浮点数
DATE	日期变量，也存时间
BLOB	音频、视频、图形、图像
CLOB	超过 4000 个字节的大块文本

约束

约束是在数据库水平上定义数据的规则。ORACLE 支持下列这些约束条件：

- 表级规则
- 防止有约束条件的表被删除
- 有效的约束条件
  - NOT NULL
  - UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK

约束的命名规则

- 用户命名或系统自动命名（SYS_Cn）
- 创建约束的时机：
  - 可以在创建表时创建
  - 在表创建之后在创建
- 约束条件有两种类型：列级和表级

约束的语法

- 列级约束

  column [CONSTRAINT constraint_name] constraint_type,

- 表级约束

  column,...

  [CONSTRAINT constraint_name] constraint_type (column, ...),

约束的类型

- 非空约束（NOT NULL）
  - 强制某列不能为空
  - 不允许使用表达式
  - 是列级约束

- 唯一键（UNIQUE KEY）
  - 指明列或列组合的值唯一
  - 当单一列有唯一约束时，可以为空值
  - 可以定义为表级或列级约束
  - 系统会自动建立唯一性索引
- 主键（PRIMARY KEY）
  - 主键约束所表示的列值，其值是唯一识别表中的一条记录
  - 每个表都应该有一个主键，且只能有一个主键
  - 指定列值唯一
  - 主键不可以为空
  - 是表级或列级约束
  - 系统会自动建立唯一性索引
- 外键（FOREIGN KEY）
  - 外键用于连接包括相关信息的两个表，他们大多使用在父子关系中
  - 可以定义在单列或多列上
  - 会同本表或其他表的主键或唯一键建立连接关系
  - 是表级或列级约束
  - 插入值必须同父表匹配或为 NULL
  - FOREIGN KEY：指明子表中的列
  - REFERENCES：指明父级表及相关列
  - ON DELETE CASCADE：允许父表中的删除操作，并且删除子表中的相关记录
- CHECK
  - 制定每条记录必须满足的条件
  - 不允许使用表达式
    - 参照伪列 CURRVAL, NEXTVAL, LEVEL, 或 ROWNUM
    - 调用 SYSDATE, UID, USER, 或 USERENV 函数
    - 与其它行中其它值有关的查询值
  - 是表级或列级约束

例：创建两个数据库基表 department 和 employee;

```
CREATE TABLE department (
      deptno NUMBER(2) NOT NULL,
      dname CHAR(14),
      loc CHAR(13),
      CONSTRAINT department_primary_key PRIMARY KEY (deptno));

CREATE TABLE employee (
      empno NUMBER(4) NOT NULL,
      name CHAR(10),
      job CHAR(9),
      mgr   NUMBER(4) CONSTRAINT emp_self_key
            REFERENCES employee (empno),
      hiredate DATE,
      sal NUMBER(7,2),
      comm NUMBER(7,2),
      deptno NUMBER(2) NOT NULL,
      CONSTRAINT emp_foreign_key FOREIGN KEY (deptno)
            REFERENCES department(deptno),
```

CONSTRAINT emp_primary_key PRIMARY KEY (empno));

使用子查询创建表

- 使用 CREATE TABLE 和 AS 子查询创建表并且插入记录
  CREATE TABLE table [column(, column...)] AS sub-query;
- 指定列应与子查询中的列相匹配
- 为列定义了列名、缺省值和完整性约束

例：

```
CREATE TABLE new_emp_10 AS
    SELECT * FROM emp WHERE deptno=10;
CREATE TABLE new_emp_20 AS
    SELECT empno empid, ename name, sal salary
        FROM emp WHERE deptno=20;
```

例：创建具有表结构但没有任何数据记录的基表

```
CREATE TABLE new_dept AS SELECT * FROM dept WHERE 1=2;
```

利用数据字典查看约束条件

```
SELECT constraint_name, search_condition
    FROM all_constraints WHERE owner='SCOTT';
```

改变表中列的名字

```
CREATE TABLE new_table (new_columns)
    AS SELECT * FROM old_table;
DROP TABLE old_table;
RENAME new_table TO old_table;
```

删除表中的列

```
ALTER TABLE tablename DROP COLUMN columnname;
```

改变一个字段初始定义的 Check 范围

- 从 user_constraints 里查出此约束的名字(constraint_name)：
  SELECT constraint_name, table_name, search_condition
      FROM user_constraints;
- 使用 ALTER TABLE 语句删除旧约束:
  ALTER TABLE tablename DROP CONSTRAINT constraint_name;
- 再创建新约束:
  ALTER TABLE tablename ADD CONSTRAINT constraint_name CHECK();

实验二　操纵数据

实验保存为学号-实验-2.doc 文件提交。

【实验目标】

- 向表中添加记录；
- 修改表中已有记录；
- 删除表中已有记录；
- 事务控制及其重要性。

数据操纵和事务控制命令

Command	description
INSERT	向表中添加记录
UPDATE	修改表中已有记录
DELETE	删除表中已有记录
COMMIT	提交事务
SAVEPOINT	回到一个存储点
ROLLBACK	回滚事务

插入数据

语法格式：

- 使用 INSERT 命令向基表插入数据

  INSERT INTO table[(column [, column …])]
  　　VALUES (value [, value …]);
- 一次只能插入一条记录
- 插入数据时，为每列都插入数据
- 可以在 INSERT 语句中列出列名
- 按照表中列的顺序给出相应数值
- 字符型和日期型数值需使用‘ ’

  INSERT INTO dept VALUES(50,’Administration’, ‘CHICAGO’);

插入空值

- 隐式规则：列表中未列出的列

  INSERT INTO dept(deptno, dname) VALUES(60, ‘MIS’);
- 显式规则：指定值为空值或空字符串

  INSERT INTO dept VALUES(70, ‘DEVELOPMENT’, NULL);

插入特定值

- USER 变量中记录的是当前的用户信息
- SYSDATE 变量记录当前日期和时间

  INSERT INTO emp(empno, ename, hiredate)
  　　VALUES(5555, USER, SYSDATE);

插入时，指定日期格式

- TO_DATE
  - 利用格式串插入指定日期和时间值
- 缺省值

- 缺省时间为当前时间
- 缺省时间为午夜12：00

```
INSERT INTO EMP(empno, ename, hiredate, deptno)
        VALUES (3333, USER, TO_DATE('06-10 月-01','dd-mon-yy'),30);
```

使用变量插入数据

- 使用 SQL*PLUS 中的变量可以实现交互式的插入

```
INSERT INTO emp(empno, ename, deptno)
    VALUES(&no, '&name', &d_no);
```

定制提示

- SQL*PLUS ACCEPT 命令将值存入变量
- SQL*PLUS PROMPT 允许使用特定提示信息

例：

```
ACCEPT dept_id PROMPT 'Please      enter the department number:';
ACCEPT dept_name PROMPT 'Please enter the department name:';
ACCEPT region_name PROMPT 'Please enter the region number:';
INSERT INTO dept (deptno, dname, loc)
    VALUES (&dept_id, '&dept_name', '&region_name');
```

拷贝记录

- 在 INSERT 命令中使用子查询
- 不用 VALUES 子句
- 子查询中的列应和 INSERT 子句中的列相匹配

```
CREATE TABLE new_emp AS
    SELECT * FROM emp WHERE deptno=10;
INSERT INTO new_emp
    SELECT * FROM emp WHERE deptno=20;
```

更新数据

- 使用 UPDATE 命令更新基表中的数据。语法格式：

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

例：部门号为 10 的员工调入部门 40，并且工资均为 2500；

```
UPDATE emp set deptno=40, sal=2500 WHERE deptno=10;
```

- 不使用 WHERE 子句将更新所有记录

例：将所有员工的工资设定为 4000；

```
UPDATE emp set sal=4000;
```

删除数据

- 使用 DELETE 命令删除数据。语法格式：

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

例：删除员工号码为 2222 的员工；

```
DELETE FROM emp WHERE empno=2222;
```

- 不使用 WHERE 子句将删除所有记录

例：删除所有员工的记录；

DELETE FROM emp;

注：若要删除的记录中包含有主键，而它是其它表的外键，则将违反完整性约束错误。

数据库事务

事务是由一系列 SQL 语句组成的单个逻辑工作单元，其中 DML 语句对数据库所做的所有修改被作为一个整体，要么被全部提交，要么被全部回滚。当事务执行失败时，数据库能够自动恢复到它以前的状态。

- 包含下列语句
  - DML 命令
  - DDL 命令
  - DCL 命令
- 第一条可执行 SQL 命令执行时开始

事务提交

结束当前事务，并且向数据库提交，对数据库中数据进行永久性的修改。

语法格式：COMMIT；

事务回滚

结束当前事务，并且放弃所有当前事务中对数据库中数据的修改。

语法格式：ROLLBACK TO <SAVEPOINT> 标记；

事务保存点

在当前事务中加入标记。

语法格式：SAVEPOINT 标记；

事务随下列事件而结束:

- COMMIT or ROLLBACK
- DDL or DCL
- Errors, 退出或系统异常

优越性

- 保证数据一致性
- 可以在确定更新前先预览更新后的结果
- 组操作

隐式控制事务

- 下列情况下系统将自动提交
  - DDL 命令，如 CREATE.
  - DCL 命令，如 GRANT.
  - 正常退出
- 发生系统错误或 SQL*PLUS 异常退出时，系统自动回退

回退至标记

- 使用 SAVEPOINT 命令在当前事务中加入标记
- ROLLBACK 命令可以回退到标记位置.

例：

```
SAVEPOINT A；
INSERT INTO dept VALUES(60, ’SOFTWARE’, ’NEW YORK’);
SAVEPOINT B;
INSERT INTO dept VALUES (70, ’SERVICES’, ’CHICAGO’);
SELECT * FROM dept;
ROLLBACK TO B;
SELECT * FROM dept;
ROLLBACK TO A;
SELECT * FROM dept;
```

在 COMMIT 或 ROLLBACK 之前数据的状态

- 因为数据缓存区的作用，以前的数据状态能够得到恢复；
- 当前的用户通过使用SELECT语句可以预览DML操作之后的结果；
- 其它用户不能浏览当前用户DML操作之后的结果；
- 受影响的行被锁定；其它用户不能修改受影响的行的数据。

COMMIT 之后的数据状态

- 数据的修改被写入数据库中；
- 以前的数据被永久丢失；
- 所有用户都可以浏览数据；
- 受影响行的锁被释放，其它用户都可以操纵这些行；
- 所有的保存点被删除。

ROLLBACK 之后的数据状态

放弃所有未决的修改

- 数据修改没有完成；
- 数据以前的状态被恢复；
- 受影响行的锁被释放。

语句级回退

- 如果一条DML语句在执行期间失败，只有那条语句被回退；
- ORACLE设置隐式保存点；
- 所有其它修改被保留；
- 用户应该通过执行 COMMIT 或 ROLLBACK 显示地终止事务。

实验三　修改基表及其约束

实验保存为学号-实验-3.doc 文件提交。

【实验目标-】

- 增加和修改列；
- 增加、有效、无效或删除约束条件；
- 删除一个基表；
- 保留基表定义，删除所有记录行。
- 改变对象的名称；
- 增加注释并且通过数据字典查询注释。

向基表中增加新列

- 为新列定义一个缺省的新值；
- 指定列必须包含值；
- 新列成为最后的列。

语法格式：

```
ALTER TABLE table
    ADD (column datatype [DEFAULT expr] [NOT NULL]
        [, column datatype]...);
```

例：给 dept 表中增加一个 COMMENTS 列；

```
ALTER TABLE dept    ADD (comments VARCHAR2(20));
```

修改基表中的列

- 修改一个列的数据类型、数据长度、缺省值、和NOT NULL 列约束。

语法格式：

```
ALTER TABLE table
    MODIFY (column datatype [DEFAULT expr] [NOT NULL]
        [, column datatype]...);
```

规则

- 增加一个数字类型的列的宽度或精度；
- 如果列中包含空值活该表中没有记录，可以减少列的宽度；
- 为后来增加的列修改缺省值；
- 如果该列中没有值，可以定义NOT NULL 约束。
- 如果该列包含空值，可以修改该列的数据类型；
- 修改的缺省值对后来增加到表中的记录有效。

例：扩展dept 表中COMMENTS 列的最大长度为30个字符；

```
ALTER TABLE dept
    MODIFY (comments varchar2(30));
```

删除基表中的列

语法格式：

```
ALTER TABLE table
    DROP COLUMN column;
```

例：删除 dept 表中的 COMMENTS 列；

```
ALTER TABLE dept
    DROP COLUMN comments;
```

向基表中增加约束条件

语法格式：

```
ALTER TABLE table
    ADD [ CONSTRAINT constraint ] type (column);
```

- 增加或删除、但不能修改约束条件；
- 使约束条件有效或无效；
- 可以使用 MODIFY 子句增加 NOT NULL 约束条件。

例： 向基表 emp 中增加一个外键约束，指定员工的经理代号时，其经理的记录必须已经在 emp 中存在。

```
ALTER TABLE emp
    ADD CONSTRAINT emp_manager_id_fk
        FOREIGN KEY (mgr) REFERENCES emp(empno);
```

删除基表中的约束条件

语法格式：

```
ALTER TABLE table
    DROP CONSTRAINT constraint ;
```

例：从 emp 表中删除 emp_manager_id_fk 约束;

```
ALTER TABLE emp
    DROP CONSTRAINT emp_manager_id_fk;
```

使约束条件无效

语法格式：

```
ALTER TABLE table
    DISABLE CONSTRAINT constraint [CASCADE];
```

- 执行 ALTER TABLE 命令中 DISABLE 子句能够使完整性约束无效。
- 应用 CASCADE 选项使依靠的完整性约束无效。

```
ALTER TABLE emp
    DISABLE CONSTRAINT pk_emp CASCADE;
```

使约束条件有效

语法格式：

```
ALTER TABLE table
    ENABLE CONSTRAINT constraint ;
```

- 使用 ENABLE 子句可以激活在表中定义的无效的当前完整性约束；
- 如果使唯一性约束和主键约束有效，唯一性索引或主键索引将自动创建。

```
ALTER TABLE emp
    ENABLE CONSTRAINT pk_emp;
```

删除基表

语法格式：

```
DROP TABLE tablename [ CASCADE CONSTRAINTS];
```

- 表中所有数据将被删除；
- 所有未提交的事务全部提交；
- 所有索引被删除；

- CASCADE CONSTRAINTS 选项删除了依赖的完整性约束。
- 命令不能回退。

改变数据对象的名字

语法格式：

RENAME old_table TO new_table;

- 执行 RENAME 命令可以改变基表、视图、序列或同义词的名字；
- 必须是这些数据对象的属主。

RENAME emp TO new_emp;

删除基表中的记录

语法格式：

TRUNCATE TABLE table;

- 删除基表中所有的记录；
- 释放该基表所占用的存储空间。Releases the storage space used by that table;
- 是一条 DDL 命令。
- 使用 TRUNCATE 不能回滚记录。
- 可以选择 DELETE 命令删除记录。

CREATE TABLE new_emp AS SELECT * FROM emp;

TRUNCATE TABLE new_emp;

向基表中增加注释

语法格式：

COMMENT ON TABLE table IS ‘comments’;

- 使用 COMMENT 可以向表或列中增加注释；

COMMENT ON TABLE emp IS ‘Employee Information’;

- 使用空串可以清除注释。

COMMENT ON TABLE emp IS ‘’;

- 可以使用下列数据字典查询注释。
  1. ALL_COL_COMMENTS
  2. USER_COL_COMMENTS
  3. ALL_TAB_COMMENTS
  4. USER_TAB_COMMENTS

作业一 表的定义和操纵

作业保存为学号-1.sql 文件提交。

1.数据库模式建立

（1）熟悉上机环境和 sqlplus 中的各种操作命令。

（2）创建数据库模式：新建立七张表，分别是 EMP（员工表）、DEPT（部门表）、REGIONS（地区表）、LOCATIONS（位置表）、DEPARTMENTS（公寓表）、JOBS（工作表）、EMPLOYEES（职业表）、JOB_HISTORY（工作经历表）和 COUNTRIES（国家表）。

表 1：EMP 表结构如下：

Name	Type	Nullable Default	Comments
EMPNO	NUMBER(4)	NOT NULL	员工号
ENAME	VARCHAR2(10)	NOT NULL	员工姓名
JOB	VARCHAR2(9)	NOT NULL	工作
MGR	NUMBER(4)		上级编号
HIREDATE	DATE	NOT NULL	雇佣日期
SAL	NUMBER(7,2)	NOT NULL	薪金
COMM	NUMBER(7,2)		佣金
DEPTNO	NUMBER(4)	NOT NULL	部门编号

表 2：DEPT 表结构如下：

Name	Type	Nullable Default	Comments
DEPTNO	NUMBER(4)	NOT NULL	部门编号
DNAME	VARCHAR2(14)	NOT NULL	部门名称
LOC	VARCHAR2(13)	NOT NULL	地点

提示：工资 ＝ 薪金 ＋ 佣金

表 3：REGIONS 表结构如下：

名称	是否为空?	类型
REGION_ID	NOT NULL	NUMBER
REGION_NAME		VARCHAR2(25)

表 4：LOCATIONS **表结构如下：**

名称	是否为空?	类型
LOCATION_ID	NOT NULL	NUMBER(4)
STREET_ADDRESS		VARCHAR2(40)
POSTAL_CODE		VARCHAR2(12)
CITY	NOT NULL	VARCHAR2(30)
STATE_PROVINCE		VARCHAR2(25)
COUNTRY_ID		CHAR(2)

表 5：DEPARTMENTS **表结构如下：**

名称	是否为空?	类型
DEPARTMENT_ID	NOT NULL	NUMBER(4)

DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

表 6：JOBS **表结构如下：**

名称	是否为空?	类型
JOB_ID	NOT NULL	VARCHAR2(10)
JOB_TITLE	NOT NULL	VARCHAR2(35)
MIN_SALARY		NUMBER(6)
MAX_SALARY		NUMBER(6)

表 7：EMPLOYEES **表结构如下：**

名称	是否为空?	类型
EMPLOYEE_ID	NOT NULL	NUMBER(4)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

表 8：JOB_HISTORY **表结构如下：**

名称	是否为空?	类型
EMPLOYEE_ID	NOT NULL	NUMBER(4)
START_DATE	NOT NULL	DATE
END_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
DEPARTMENT_ID		NUMBER(4)

表 9：COUNTRIES **表结构如下：**

名称	是否为空?	类型
COUNTRY_ID	NOT NULL	CHAR(2)
COUNTRY_NAME		VARCHAR2(40)
REGION_ID		NUMBER

（3）用“SHOW USER”显示当前用户。

（4）用“SELECT * FROM USER_TABLES” 命令检查当前用户所建表情况是否正确。

（5）用“DESC <TABLE_NAME>” 命令检查所构建表的表结构是否正确。

（6）用“SELECT * FORM USER_CONSTRAINTS;”命令检查所构建关系模式的约束是否正确。熟悉各约束定义，R：foreign key，P:primary key， C：Not Null or Check， U：Unique

2．数据库更新操作

用 SQL 语言数据操作功能将 DEPT（部门表）和 EMP（员工表）的内容设置为如下形式。

表 1：DEPT 表数据形式：

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

表 2：EMP 表数据形式：

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-12 月 -80	800		20
7499	ALLEN	SALESMAN	7698	20-2 月 -81	1600	300	30
7521	WARD	SALESMAN	7698	22-2 月 -81	1250	500	30
7566	JONES	MANAGER	7839	02-4 月 -81	2975		20
7654	MARTIN	SALESMAN	7698	28-9 月 -81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-5 月 -81	2850		30
7782	CLARK	MANAGER	7839	09-6 月 -81	2450		10
7788	SCOTT	ANALYST	7566	19-4 月 -87	3000		20
7839	KING	PRESIDENT		17-11 月 -81	5000		10
7844	TURNER	SALESMAN	7698	08-9 月 -81	1500	0	30
7876	ADAMS	CLERK	7788	23-5 月 -87	1100		20
7900	JAMES	CLERK	7698	03-12 月 -81	950		30
7902	FORD	ANALYST	7566	03-12 月 -81	3000		20
7934	MILLER	CLERK	7782	23-1 月 -82	1300		10

表 3：**REGIONS 表数据形式：**

REGION_ID	REGION_NAME
1	Europe
2	Americas

3	Asia
4	Middle East and Africa

表 4：**LOCATIONS 表数据形式：**

LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY	STATE_PROVINCE	COUNTR
1000	1297 Via Cola di Rie	00989	Roma		IT
1100	93091 Calle della Testa	10934	Venice		IT
1200	2017 Shinjuku-ku	1689	Tokyo	Tokyo Prefecture	JP
1300	9450 Kamiya-cho	6823	Hiroshima		JP
1400	2014 Jabberwocky Rd	26192	Southlake	Texas	US
1500	2011 Interiors Blvd	99236	South San Francisco	California	US
1600	2007 Zagora St	50090	South Brunswick	New Jersey	US
1700	2004 Charade Rd	98199	Seattle	Washington	US
1800	147 Spadina Ave	M5V 2L7	Toronto	Ontario	CA
1900	6092 Boxwood St	YSW 9T2	Whitehorse	Yukon	CA
2000	40-5-12 Laogianggen	190518	Beijing		CN
2100	1298 Vileparle (E)	490231	Bombay	Maharashtra	IN
2200	12-98 Victoria Street	2901	Sydney	New South Wales	AU
2300	198 Clementi North	540198	Singapore		SG
2400	8204 Arthur St		London		UK
2500	Magdalen Centre, The Oxford Science Park	OX9 9ZB	Oxford	Oxford	UK
2600	9702 Chester Road	09629850293	Stretford	Manchester	UK
2700	Schwanthalerstr. 7031	80925	Munich	Bavaria	DE
2800	Rua Frei Caneca 1360	01307-002	Sao Paulo	Sao Paulo	BR
2900	20 Rue des Corps-Saints	1730	Geneva	Geneve	CH
3000	Murtenstrasse 921	3095	Bern	BE	CH

3100	Pieter Breughelstraat 837	3029SK	Utrecht	Utrecht	NL
3200	Mariano Escobedo 9991	11932	Mexico City	Distrito Federal,	MX

表 5：DEPARTMENTS 表数据形式：

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700
110	Accounting	205	1700
120	Treasury		1700
130	Corporate Tax		1700
140	Control And Credit		1700
150	Shareholder Services		1700
160	Benefits		1700
170	Manufacturing		1700
180	Construction		1700
190	Contracting		1700
200	Operations		1700
210	IT Support		1700
220	NOC		1700
230	IT Helpdesk		1700
240	Government Sales		1700
250	Retail Sales		1700
260	Recruiting		1700
270	Payroll		1700

表 6：JOBS 表数据形式：

JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
AD_PRES	President	20000	40000
AD_VP	Administration Vice President	15000	30000
AD_ASST	Administration Assistant	3000	6000
FI_MGR	Finance Manager	8200	16000
FI_ACCOUNT	Accountant	4200	9000
AC_MGR	Accounting Manager	8200	16000
AC_ACCOUNT	Public Accountant	4200	9000
SA_MAN	Sales Manager	10000	20000
SA_REP	Sales Representative	6000	12000
PU_MAN	Purchasing Manager	8000	15000
PU_CLERK	Purchasing Clerk	2500	5500
ST_MAN	Stock Manager	5500	8500
ST_CLERK	Stock Clerk	2000	5000
SH_CLERK	Shipping Clerk	2500	5500
IT_PROG	Programmer	4000	10000
MK_MAN	Marketing Manager	9000	15000
MK_REP	Marketing Representative	4000	9000
HR_REP	Human Resources Representative	4000	9000
PR_REP	Public Relations Representative	4500	10500

表 7：EMPLOYEES 表数据形式：

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
198	Donald	OConnell	DOCONNEL	650.507.9833	21-6月 -07	SH_CLERK	2600		124	50
199	Douglas	Grant	DGRANT	650.507.9844	13-1月 -08	SH_CLERK	2600		124	50
200	Jennifer	Whalen	JWHALEN	515.123.4444	17-9月 -03	AD_ASST	4400		101	10
201	Michael	Hartstein	MHARTSTE	515.123.5555	17-2月 -04	MK_MAN	13000		100	20
202	Pat	Fay	PFAY	603.123.6666	17-8月 -05	MK_REP	6000		201	20
203	Susan	Mavris	SMAVRIS	515.123.7777	07-6月 -02	HR_REP	6500		101	40
204	Hermann	Baer	HBAER	515.123.8888	07-6月 -02	PR_REP	10000		101	70
205	Shelley	Higgins	SHIGGINS	515.123.8080	07-6月 -02	AC_MGR	12008		101	110
206	William	Gietz	WGIETZ	515.123.8181	07-6月 -02	AC_ACCOUNT	8300		205	110
100	Steven	King	SKING	515.123.4567	17-6月 -03	AD_PRES	24000			90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-9月 -05	AD_VP	17000		100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	13-1月 -01	AD_VP	17000		100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-1月 -06	IT_PROG	9000		102	60
104	Bruce	Ernst	BERNST	590.423.4568	21-5月 -07	IT_PROG	6000		103	60
105	David	Austin	DAUSTIN	590.423.4569	25-6月 -05	IT_PROG	4800		103	60
106	Valli	Pataballa	VPATABAL	590.423.4560	05-2月 -06	IT_PROG	4800		103	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-2月 -07	IT_PROG	4200		103	60
108	Nancy	Greenberg	NGREENBE	515.124.4569	17-8月 -02	FI_MGR	12008		101	100
109	Daniel	Faviet	DFAVIET	515.124.4169	16-8月 -02	FI_ACCOUNT	9000		108	100
110	John	Chen	JCHEN	515.124.4269	28-9月 -05	FI_ACCOUNT	8200		108	100
111	Ismael	Sciarra	ISCIARRA	515.124.4369	30-9月 -05	FI_ACCOUNT	7700		108	100
112	Jose Manuel	Urman	JMURMAN	515.124.4469	07-3月 -06	FI_ACCOUNT	7800		108	100

113	Luis	Popp	LPOPP	515.124.4567	07-12月-07	FI_ACCOUNT	6900		108	100
114	Den	Raphaely	DRAPHEAL	515.127.4561	07-12月-02	PU_MAN	11000		100	30
115	Alexander	Khoo	AKHOO	515.127.4562	18-5月 -03	PU_CLERK	3100		114	30
116	Shelli	Baida	SBAIDA	515.127.4563	24-12月-05	PU_CLERK	2900		114	30
117	Sigal	Tobias	STOBIAS	515.127.4564	24-7月 -05	PU_CLERK	2800		114	30
118	Guy	Himuro	GHIMURO	515.127.4565	15-11月-06	PU_CLERK	2600		114	30
119	Karen	Colmenares	KCOLMENA	515.127.4566	10-8月 -07	PU_CLERK	2500		114	30
120	Matthew	Weiss	MWEISS	650.123.1234	18-7月 -04	ST_MAN	8000		100	50
121	Adam	Fripp	AFRIPP	650.123.2234	10-4月 -05	ST_MAN	8200		100	50
122	Payam	Kaufling	PKAUFLIN	650.123.3234	01-5月 -03	ST_MAN	7900		100	50
123	Shanta	Vollman	SVOLLMAN	650.123.4234	10-10月-05	ST_MAN	6500		100	50
124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-11月-07	ST_MAN	5800		100	50
125	Julia	Nayer	JNAYER	650.124.1214	16-7月 -05	ST_CLERK	3200		120	50
126	Irene	Mikkilineni	IMIKKILI	650.124.1224	28-9月 -06	ST_CLERK	2700		120	50
127	James	Landry	JLANDRY	650.124.1334	14-1月 -07	ST_CLERK	2400		120	50
128	Steven	Markle	SMARKLE	650.124.1434	08-3月 -08	ST_CLERK	2200		120	50
129	Laura	Bissot	LBISSOT	650.124.5234	20-8月 -05	ST_CLERK	3300		121	50
130	Mozhe	Atkinson	MATKINSO	650.124.6234	30-10月-05	ST_CLERK	2800		121	50
131	James	Marlow	JAMRLOW	650.124.7234	16-2月 -05	ST_CLERK	2500		121	50
132	TJ	Olson	TJOLSON	650.124.8234	10-4月 -07	ST_CLERK	2100		121	50
133	Jason	Mallin	JMALLIN	650.127.1934	14-6月 -04	ST_CLERK	3300		122	50
134	Michael	Rogers	MROGERS	650.127.1834	26-8月 -06	ST_CLERK	2900		122	50
135	Ki	Gee	KGEE	650.127.1734	12-12月-07	ST_CLERK	2400		122	50

136	Hazel	Philtanker	HPHILTAN	650.127.1634	06-2月 -08	ST_CLERK	2200		122	50
137	Renske	Ladwig	RLADWIG	650.121.1234	14-7月 -03	ST_CLERK	3600		123	50
138	Stephen	Stiles	SSTILES	650.121.2034	26-10月-05	ST_CLERK	3200		123	50
139	John	Seo	JSEO	650.121.2019	12-2月 -06	ST_CLERK	2700		123	50
140	Joshua	Patel	JPATEL	650.121.1834	06-4月 -06	ST_CLERK	2500		123	50
141	Trenna	Rajs	TRAJS	650.121.8009	17-10月-03	ST_CLERK	3500		124	50
142	Curtis	Davies	CDAVIES	650.121.2994	29-1月 -05	ST_CLERK	3100		124	50
143	Randall	Matos	RMATOS	650.121.2874	15-3月 -06	ST_CLERK	2600		124	50
144	Peter	Vargas	PVARGAS	650.121.2004	09-7月 -06	ST_CLERK	2500		124	50
145	John	Russell	JRUSSEL	011.44.1344.429268	01-10月-04	SA_MAN	14000	.4	100	80
146	Karen	Partners	KPARTNER	011.44.1344.467268	05-1月 -05	SA_MAN	13500	.3	100	80
147	Alberto	Errazuriz	AERRAZUR	011.44.1344.429278	10-3月 -05	SA_MAN	12000	.3	100	80
148	Gerald	Cambrault	GCAMBRAU	011.44.1344.619268	15-10月-07	SA_MAN	11000	.3	100	80
149	Eleni	Zlotkey	EZLOTKEY	011.44.1344.429018	29-1月 -08	SA_MAN	10500	.2	100	80
150	Peter	Tucker	PTUCKER	011.44.1344.129268	30-1月 -05	SA_REP	10000	.3	145	80
151	David	Bernstein	DBERNSTE	011.44.1344.345268	24-3月 -05	SA_REP	9500	.25	145	80
152	Peter	Hall	PHALL	011.44.1344.478968	20-8月 -05	SA_REP	9000	.25	145	80
153	Christopher	Olsen	COLSEN	011.44.1344.498718	30-3月 -06	SA_REP	8000	.2	145	80
154	Nanette	Cambrault	NCAMBRAU	011.44.1344.987668	09-12月-06	SA_REP	7500	.2	145	80
155	Oliver	Tuvault	OTUVAULT	011.44.1344.486508	23-11月-07	SA_REP	7000	.15	145	80
156	Janette	King	JKING	011.44.1345.429268	30-1月 -04	SA_REP	10000	.35	146	80
157	Patrick	Sully	PSULLY	011.44.1345.929268	04-3月 -04	SA_REP	9500	.35	146	80

158	Allan	McEwen	AMCEWEN	011.44.1345.829268	01-8月 -04	SA_REP	9000	.35	146	80
159	Lindsey	Smith	LSMITH	011.44.1345.729268	10-3月 -05	SA_REP	8000	.3	146	80
160	Louise	Doran	LDORAN	011.44.1345.629268	15-12月-05	SA_REP	7500	.3	146	80
161	Sarath	Sewall	SSEWALL	011.44.1345.529268	03-11月-06	SA_REP	7000	.25	146	80
162	Clara	Vishney	CVISHNEY	011.44.1346.129268	11-11月-05	SA_REP	10500	.25	147	80
163	Danielle	Greene	DGREENE	011.44.1346.229268	19-3月 -07	SA_REP	9500	.15	147	80
164	Mattea	Marvins	MMARVINS	011.44.1346.329268	24-1月 -08	SA_REP	7200	.1	147	80
165	David	Lee	DLEE	011.44.1346.529268	23-2月 -08	SA_REP	6800	.1	147	80
166	Sundar	Ande	SANDE	011.44.1346.629268	24-3月 -08	SA_REP	6400	.1	147	80
167	Amit	Banda	ABANDA	011.44.1346.729268	21-4月 -08	SA_REP	6200	.1	147	80
168	Lisa	Ozer	LOZER	011.44.1343.929268	11-3月 -05	SA_REP	11500	.25	148	80
169	Harrison	Bloom	HBLOOM	011.44.1343.829268	23-3月 -06	SA_REP	10000	.2	148	80
170	Tayler	Fox	TFOX	011.44.1343.729268	24-1月 -06	SA_REP	9600	.2	148	80
171	William	Smith	WSMITH	011.44.1343.629268	23-2月 -07	SA_REP	7400	.15	148	80
172	Elizabeth	Bates	EBATES	011.44.1343.529268	24-3月 -07	SA_REP	7300	.15	148	80
173	Sundita	Kumar	SKUMAR	011.44.1343.329268	21-4月 -08	SA_REP	6100	.1	148	80
174	Ellen	Abel	EABEL	011.44.1644.429267	11-5月 -04	SA_REP	11000	.3	149	80
175	Alyssa	Hutton	AHUTTON	011.44.1644.429266	19-3月 -05	SA_REP	8800	.25	149	80
176	Jonathon	Taylor	JTAYLOR	011.44.1644.429265	24-3月 -06	SA_REP	8600	.2	149	80
177	Jack	Livingston	JLIVINGS	011.44.1644.429264	23-4月 -06	SA_REP	8400	.2	149	80
178	Kimberely	Grant	KGRANT	011.44.1644.429263	24-5月 -07	SA_REP	7000	.15	149	
179	Charles	Johnson	CJOHNSON	011.44.1644.429262	04-1月 -08	SA_REP	6200	.1	149	80
180	Winston	Taylor	WTAYLOR	650.507.9876	24-1月 -06	SH_CLERK	3200		120	50

181	Jean	Fleaur	JFLEAUR	650.507.9877	23-2月 -06	SH_CLERK	3100		120	50
182	Martha	Sullivan	MSULLIVA	650.507.9878	21-6月 -07	SH_CLERK	2500		120	50
183	Girard	Geoni	GGEONI	650.507.9879	03-2月 -08	SH_CLERK	2800		120	50
184	Nandita	Sarchand	NSARCHAN	650.509.1876	27-1月 -04	SH_CLERK	4200		121	50
185	Alexis	Bull	ABULL	650.509.2876	20-2月 -05	SH_CLERK	4100		121	50
186	Julia	Dellinger	JDELLING	650.509.3876	24-6月 -06	SH_CLERK	3400		121	50
187	Anthony	Cabrio	ACABRIO	650.509.4876	07-2月 -07	SH_CLERK	3000		121	50
188	Kelly	Chung	KCHUNG	650.505.1876	14-6月 -05	SH_CLERK	3800		122	50
189	Jennifer	Dilly	JDILLY	650.505.2876	13-8月 -05	SH_CLERK	3600		122	50
190	Timothy	Gates	TGATES	650.505.3876	11-7月 -06	SH_CLERK	2900		122	50
191	Randall	Perkins	RPERKINS	650.505.4876	19-12月-07	SH_CLERK	2500		122	50
192	Sarah	Bell	SBELL	650.501.1876	04-2月 -04	SH_CLERK	4000		123	50
193	Britney	Everett	BEVERETT	650.501.2876	03-3月 -05	SH_CLERK	3900		123	50
194	Samuel	McCain	SMCCAIN	650.501.3876	01-7月 -06	SH_CLERK	3200		123	50
195	Vance	Jones	VJONES	650.501.4876	17-3月 -07	SH_CLERK	2800		123	50
196	Alana	Walsh	AWALSH	650.507.9811	24-4月 -06	SH_CLERK	3100		124	50
197	Kevin	Feeney	KFEENEY	650.507.9822	23-5月 -06	SH_CLERK	3000		124	50

表 8：JOB_HISTORY 表数据形式：

EMPLOYEE_ID	START_DATE	END_DATE	JOB_ID	DEPARTMENT_ID
102	13-1 月 -93	24-7 月 -98	IT_PROG	60
101	21-9 月 -89	27-10 月-93	AC_ACCOUNT	110
101	28-10 月-93	15-3 月 -97	AC_MGR	110
201	17-2 月 -96	19-12 月-99	MK_REP	20
114	24-3 月 -98	31-12 月-99	ST_CLERK	50
122	01-1 月 -99	31-12 月-99	ST_CLERK	50
200	17-9 月 -87	17-6 月 -93	AD_ASST	90
176	24-3 月 -98	31-12 月-98	SA_REP	80
176	01-1 月 -99	31-12 月-99	SA_MAN	80
200	01-7 月 -94	31-12 月-98	AC_ACCOUNT	90

表 9：COUNTRIES 表数据形式：

CO	COUNTRY_NAME	REGION_ID
AR	Argentina	2
AU	Australia	3
BE	Belgium	1
BR	Brazil	2
CA	Canada	2
CH	Switzerland	1
CN	China	3
DE	Germany	1
DK	Denmark	1
EG	Egypt	4
FR	France	1
HK	HongKong	3
IL	Israel	4
IN	India	3
IT	Italy	1
JP	Japan	3
KW	Kuwait	4
MX	Mexico	2
NG	Nigeria	4
NL	Netherlands	1
SG	Singapore	3
UK	United Kingdom	1
US	United States of America	2
ZM	Zambia	4
ZW	Zimbabwe	4

实验四　　基本查询

实验保存为学号-实验-4.doc 文件提交。

【实验目标】

- 编写 SELECT 语句用于查询数据库数据；
- 完成算术计算；
- 处理 NULL 值；
- 确定使用别名作为列的标题；
- 连接列；

SELECT 语法

SELECT [DISTINCT] { *, column [alias], … }
　　FROM table;

- SELECT 指明查询的列
- FROM 指明查询的表

书写规则：

- 命令可以写在一行或多行。
- 可以使用缩进，提高可读性。
- 关键字不能缩写或拆分。
- 大小写不敏感。
- 命令存入缓冲区。

最简单的查询语句包含如下两个子句：

- SELECT 子句：用 * 代表所有列
- FROM 子句

例：显示 dept 表的所有记录。
　　SELECT * FROM dept;

例：显示 emp 表的所有记录。
　　SELECT * FROM emp ;

查询指定列

- 在 SELECT 子句中列出要查询的列，用，分隔。查询结果中列的排列顺序由 SELECT 子句中位置而定。
　　SELECT empno, ename, sal, deptno FROM emp;
　　SELECT deptno, dname FROM dept;

列标识的缺省对齐模式：

- 日期、字符左对齐
- 数字右对齐
- 缺省的标识显示为大写.

算术表达式

- 可以对数字、日期类型使用算术表达式。
- 算术运算符包括加(+) ， 减(-) ， 乘(*) ， 除(/) 。

例：显示每个雇员的年工资。

SELECT empno, ename, sal*12 FROM emp;

操作符优先级：

- 先乘除后加减。
- 同级算符从左到右。
- ( )改变优先次序。

SELECT ename, 12*sal+100 FROM emp;

SELECT ename, 12*(sal+100) FROM emp;

使用别名改变列标识

- 用于算术表达式。
- 直接跟在列后面，可以在列后面使用关键字 AS。
- 若别名中由空格、特殊字符或大小写敏感，应使用“”。

SELECT empno, ename name, sal*12 “Annual Salary” FROM emp;

连接符（||）

- 连接列或字符串。
- 产生字符表达式。

SELECT ename ||’-----‘||sal||’-----‘||deptno FROM emp;

空值处理

- NULL 表示未定义、不可用且不占用存贮空间。
- NULL 不是 0 或空格。
- 包含 null 数学表达式也为 NULL。

SELECT ename, sal, comm, sal+comm FROM emp;

NVL 函数：将 NULL 转换成其他值.

- 可用于数字、日期、字符.
- 数据类型必须匹配.

NVL (hiredate, '01-JAN-95')

NVL (title, 'No Title Yet')

NVL (salary, 1000)

SELECT ename, sal, comm, sal+NVL(comm, 0) FROM emp;

去重（DISTINCT）

- 缺省的查询结果显示所有记录（包括重复的值）；

SELECT deptno FROM emp;

- 使用 DISTINCT 关键字去掉重复记录；

SELECT DISTINCT deptno FROM emp;

- DISTINCT 可以用于 SELECT 列表中的所有的列。
- DISTINCT 可以用于 SELECT 列表中的多个列, 结果为列组合的去重。

例：SELECT DISTINCT deptno, job FROM emp;

在数据库中，空值用来表示实际值未知或无意义的情况。在一个表中，如果一行中的某列没有值，那么就称它为空值（NULL）。任何数据类型的列,只要没有使用非空（NOT NULL）或主键（PRIMARYKEY）完整性限制，都可以出现空值。在实际应用中，如果忽略空值的存在，将会造成造成不必要的麻烦。

例如，在雇员表（EMP）中，雇员名（ENAME）为 KING 的行，因为 KING 为最高官员（PRESIDENT），他没有主管（MGR），所以其 MGR 为空值。因为不是所有的雇员都有手续费（COMM），所以列 COMM 允许有空值，除 300、500、1400、0 以外的其它各行 COMM 均为空值。

以 EMP 表为例，具体讨论一下空值在日常应用中所具有的一些特性。

空值的生成：

- 如果一列没有非空（NOTNULL）完整性限制，那么其缺省的值为空值，即如果插入一行时未指定该列的值，则其值为空值。
- 使用 SQL 语句 INSERT 插入行，凡未涉及到的列，其值为空值；涉及到的列，如果其值确实为空值，插入时可以用 NULL 来表示（对于字符型的列，也可以用''来表示）。

例：插入一行，其 EMPNO 为 1、ENAME 为'JIA'、SAL 为 10000、job 和 comm 为空值。

```
INSERT INTO emp(empno, ename, job, sal, comm)
        VALUES(1, 'JIA', NULL, 1000, NULL);
SELECT * FROM emp WHERE empno=1;
```

可以看到新插入的一行，除 job 和 comm 为空值外，mgr、hiredate、deptno 三列由于插入时未涉及，也为空值。

- 使用 SQL 语句 UPDATE 来修改数据，空值可用 NULL 来表示（对于字符型的列，也可以用''来表示）。

```
UPDATE emp set ename=NULL, sal=NULL WHERE empno=1;
```

空值的特点：

- 等价于没有任何值。
- 与 0、空字符串或空格不同。
- 在 WHERE 条件中,ORACLE 认为结果为 NULL 的条件为 FALSE，带有这样条件的 SELECT 语句不返回行，并且不返回错误信息。但 NULL 和 FALSE 是不同的。
- 排序时比其他数据都大。
- 空值不能被索引。

空值的测试：

因为空值表示缺少数据，所以空值和其它值没有可比性，即不能用等于、不等于、大于或小于和其它数值比较，当然也包括空值本身（但是在 decode 中例外，两个空值被认为是等价）。测试空值只能用比较操作符 IS NULL 和 IS NOT NULL。如果使用带有其它比较操作符的条件表达式，并且其结果依赖于空值，那么其结果必定是 NULL。在 WHERE 条件中，ORACLE 认为结果为 NULL 的条件为 FALSE，带有这样条件的 SELECT 语句不返回行，也不返回错误信息。

例：查询 EMP 表中 MGR 为 NULL 的行：

SELECT * FROM emp WHERE mgr='';

SELECT * FROM emp WHERE mgr=null;

SELECT * FROM emp WHERE mgr IS NULL;

第 1、2 句写法不妥，WHERE 条件结果为 NULL，不返回行。第三句正确，返回 MGR 为空值的行。

例：

SELECT ename, sal FROM emp WHERE sal>ANY(3000, NULL);

条件"sal>ANY(3000, NULL)"等价于 sal>3000 OR sal>NULL。

例：

SELECT ename, sal FROM emp WHERE sal>ALL(3000, NULL);

条件"sal>all(3000,NULL)"等价于 sal>3000 AND sal>NULL，结果只能为 NULL 或 FALSE，所以不返回行。

实验五　　条件查询

实验保存为学号-实验-5.doc 文件提交。

【实验目标】

- 使用 ORDER BY 子句排序查询数据；
- 使用 WHERE 子句输入查询条件。

ORDER BY 子句：用于查询结果排序。

- ASC – 升序, 缺省；
- DESC – 降序；
- 在 SELECT 命令中，ORDER BY 子句是最后一句；
- 缺省的排列顺序是升序；
- 可以使用关键字 DESC 改变排列顺序；
- 可以对表达式和别名排列顺序；
- 可以对列出现的位置进行排序；
- 可以对多列进行排序；
- 按照关键字 ORDER BY 后面的列表的顺序进行排序；
- 可以对未出现在 SELECT 子句中的列进行排序。

例：

```
SELECT * FROM emp ORDER BY deptno, sal DESC;
SELECT * FROM emp ORDER BY deptno, comm DESC;
SELECT ename, sal*12 FROM emp ORDER BY 2;
```

条件查询

使用 WHERE 子句限制显示的结果，实现条件查询，WHERE 子句跟在 FROM 子句后。条件有下面三部分组成：

- 列、表达式、常数
- 比较运算符
- 结果

字符串、日期的使用

- 字符串、日期应用‘’；
- 字符串大小写敏感；
- 缺省的日期格式为 'DD-MON-YY'。

```
SELECT * FROM emp WHERE ename=’SCOTT’;
SELECT * FROM emp WHERE LOWER(ename)=’scott’;
```

- 比较运算符：= 、> 、>= 、< 、<=；
- SQL 算符：BETWEEN ... AND...、IN(*list*)、LIKE、IS NULL；
- 逻辑算符：AND、OR、NOT。

- 使用 BETWEEN 查找列值在指定范围内的记录。

SELECT * FROM emp WHERE sal between 2000 AND 3000;

- 使用 IN 查找列值在列表内条件的记录。
  SELECT * FROM emp WHERE deptno in (10, 20);

- LIKE 算符
  - 可以使用 LIKE 算符实现通配符匹配；
  - 比较条件中可以含有字符和数字；
  - “%” 代表无字符或多字符；
  - “_” 代表单个字符；
    SELECT * FROM emp WHERE ename LIKE ‘M%’;
  - 使用 LIKE 算符，有时可以替代 BETWEEN 算符；
    SELECT * FROM emp WHERE hiredate LIKE ‘%81’;
  - 可以拼接通配符.
    SELECT * FROM emp WHERE ename LIKE ‘_A%’;
  - 可以使用 ESCAPE 实现通配符的查找.
    INSERT INTO emp(empno,ename) VALUES (88,’bill_gate’);
    SELECT * FROM emp WHERE ename LIKE ‘%ll_g%’ ESCAPE ‘\’;

- IS NULL 算符
  - 测试是否为空值；
  - 不能使用 = 。
  SELECT * FROM emp WHERE comm IS NULL;
  SELECT * FROM emp WHERE comm IS NOT NULL;

否定条件表达式：用于排除不符合条件的记录。

- 逻辑运算符： != 、 <> 、 ^= ；
- SQL 算符：NOT BETWEEN、NOT IN、NOT LIKE、IS NOT NULL。

多条件查询：

- 使用合成的条件；
- 使用 AND、OR 组成多条件；
- AND 要求两个条件都为真；
- OR 要求任一条件为真。
  SELECT * FROM emp WHERE deptno=20 AND sal>2000;
  SELECT * FROM emp WHERE deptno=20 OR sal>2000;

操作符的优先级

- 所有的比较算符.
- AND
- OR

例：列出部门 10，工资大于 1000 和部门 20 的雇员信息。

```
SELECT * FROM emp
    WHERE sal>=1000 AND deptno=10 OR deptno=20;
```

例：列出部门 10 和部门 20，工资大于 1000 的雇员信息。

```
SELECT * FROM emp
    WHERE sal>=1000 AND (deptno=10 OR deptno=20);
```

- 使用变量：在变量名称前加&

```
SELECT * FROM emp WHERE deptno=&d_no;
SELECT * FROM emp WHERE ename=&name;
SELECT * FROM emp WHERE ename='&name';
```

实验六　　单行函数

实验保存为学号-实验-6.doc 文件提交。

【实验目标】

- 解释 SQL 语句中可以使用的各种类型函数；
- 明确使用函数的基本概念；
- 在 SELECT 语句中，包含字符、数字和日期函数的种类；
- 解释转换函数及其用法。

SQL 中函数的功能

- 数值计算.
- 改变数据.
- 控制记录组输出.
- 改变日期显示.
- 类型转换.

SQL 函数的类型

- 单行函数（对每条记录有效）
  - 字符类
  - 数字类
  - 日期类
  - 转换类
- 多行函数（对多条记录有效）
  - 组函数

单行函数的功能：

- 操纵数据
- 根据命令返回数据
- 单行操作，且每行返回值
- 数据类型转换
- 可嵌套

语法格式：

function_name(column | expression , [argument1, argument2 …])

字符格式转换函数

- LOWER　　转换字符串为小写字母；
- UPPER　　转换字符串为大写字母；
- INITCAP　　转换字符串的所有单词，使该单词均以大写字母开头；
- CONCAT　　将两个输入字符串组合成一个；
- SUBSTR 返回一字符串的指定字符串；
- LENGTH　　返回字符串的长度；
- NVL　　需要两个参数，如果第一个参数为空，则返回第二个参数。

函　数	结　果
LOWER('SQL Course')	sql course
UPPER('SQL Course')	SQL COURSE
INITCAP('SQL Course')	Sql Course
CONCAT('Good', 'String')	GoodString
SUBSTR('String',1,3)	Str
LENGTH('String')	6

例：

SELECT * FROM emp WHERE LOWER(ename)='scott';
SELECT empno, ename, NVL(to_char(mgr), 'no manager') manager FROM emp;
SELECT INITCAP('SQL') FROM dual;

数字函数

- ROUND　　将一个数值舍入成第二个参数指定形式的十进制数值；
- TRUNC　　将一个数值截取成第二个参数指定形式的十进制数值；
- MOD　　返回两数相除的余数。

函　数	结　果
ROUND (45.923, 2)	45.92
ROUND (45.923, 0)	46
ROUND (45.923, -1)	50
TRUNC (45.923, 2)	45.92
TRUNC (45.923)	45
TRUNC (45.923, -1)	40
MOD(1600,300)	100

日期格式

- ORACLE 将日期按照内部格式将日期存成以下七个字节
  - 世纪、年、月、日、时、分、秒
- 在 ORACLE 中日期缺省的显示格式为：DD-MON-YY
- 可以使用系统变量 SYSDATE 获取系统时间
- 查看 SYSDATE 时，可以使用虚拟表 DUAL

日期函数

ORACLE 的日期格式：DD-MON-YY

- MONTHS_BETWEEN：返回两个日期之间的月份数；
  格式：MONTHS_BETWEEN(date1,date2)
  SELECT MONTHS_BETWEEN(TO_DATE('29-12 月-1999','DD-MON-YYYY'),
  TO_DATE('29-12 月-1988','DD-MON-YYYY')) FROM dual;

- ADD_MONTHS：返回一个日期加上一制定的月份数；
  格式：ADD_MONTHS(date,months)

SELECT ADD_MONTHS(TO_DATE('29-10 月-1999','DD-MON-YYYY'), 2)
FROM dual;

- NEXT_DAY：返回从实参日期开始，紧随其后的指定星期所对应的日期；
  格式：NEXT_DAY(date,weekday)
  SELECT NEXT_DAY(SYSDATE,'星期一') FROM dual;

- LAST_DAY：返回实参指定日期所对应月份的最后一天；
  格式：LAST_DAY(date)
  SELECT LAST_DAY(SYSDATE) FROM dual;

- ROUND：　将一个日期舍入成第二个参数指定形式的日期；
  SELECT ROUND(TO_DATE('24-11 月-01', 'DD-MON-YY'), 'MONTH')
  FROM dual;
  SELECT ROUND(TO_DATE('1-7 月-01', 'DD-MON-YY'), 'YEAR')
  FROM dual;

- TRUNC：　将一个日期截取成第二个参数指定形式的日期；
  SELECT TRUNC(TO_DATE('24-11 月-01', 'DD-MON-YY'), 'MONTH')
  FROM dual;
  SELECT TRUNC(TO_DATE('1-7 月-01', 'DD-MON-YY'), 'YEAR')
  FROM dual;

转换函数

- 在转换函数中需要使用格式串
- TO_CHAR　　将数字或日期转换为字符串
  TO_CHAR 转换日期：格式：TO_CHAR(date, 'fmt')
  格式串
  - 包含在 '' 中
  - 大小写敏感
  - 可以是任意有效的日期格式
  - 使用 fm 去除占位符
  - 与日期之间使用，分隔

  日期格式串分类
  - YYYY 表示完整的四位数字年
  - YEAR 表示英文拼写的年
  - MM 表示两位数字月
  - MONTH 表示英文全拼的年
  - DY 表示三位缩写的星期
  - DAY 表示英文全拼的日期

  - 时间格式：HH24:MI:SS AM　　15:45:32 PM
  - 在格式串中加入字符串：DD " of " MONTH　　12 of OCTOBER

- 用后缀拼出整个日期：ddspth　　fourteenth

例：

SELECT TO_CHAR(SYSDATE, 'day dd-mon-yyyy') FROM dual;
SELECT TO_CHAR(SYSDATE, 'day dd-mm-yyyy') FROM dual;
SELECT TO_CHAR(SYSDATE, 'ddd yyyy') FROM dual;
--返回从今年开始的天数；
SELECT TO_CHAR(SYSDATE, 'day dd-mon-yyyy HH24:MI:SS AM') FROM dual;
SELECT TO_CHAR(SYSDATE, 'dd "of" month,yyyy') FROM dual;
SELECT TO_CHAR(SYSDATE, 'Year, Month,ddspth') FROM dual;

TO_CHAR 转换函数：使用此函数将数字转换成字符
格式：TO_CHAR(number, 'fmt')
- 9　表示数字
- 0　强制为 0
- $　设置美元符号
- L　使用当前字符集的货币符号
- .　小数点
- ,　千位分隔符

TO_CHAR 转换数字
- 如果出现#，表示格式中没有为数字设定足够的位数
- ORACLE 中可自动按格式将数字四舍五入

- TO_NUMBER　　将字符串转换成数字
- TO_DATE　　将字符串转换成日期

SELECT to_date('01-10 月-01','dd-mon-yyyy') FROM dual;

单行函数的嵌套

- 单行函数可以任意嵌套
- 嵌套函数从最内层的函数开始计算

虚拟表 dual 的用法：

- SELECT USER FROM dual;　查询当前用户；
- SELECT SYSDATE FROM dual;　查询数据库日期；
- SELECT 25*35 FROM dual;　用作计算器；
- CREATE SEQUENCE empse INCREMENT BY 1 START WITH 1;
  SELECT empse.NEXTVAL FROM dual;
  SELECT empse.CURRVAL FROM dual;

作业二 单表查询

作业保存为学号-2.sql 文件提交

在作业一的基础上，完成如下题目。并将每个实验结果均存储为一张视图表。方式如下：

```
create view T2_* [(..)]
        AS
(Select ……);//查询语句
```

1. 在雇员表中查询雇员的编号、姓名、工作。（**其视图名为 T2_1,后面的题目以此类推**）
2. 为题目 1 中查询列取别名（number, NAME,JOB ）。
3. 查询所有的工作。
4. 按照以下的格式进行结果输出，如 NO:7469,Name:SMITH,Job:CLERK。
5. 列出每个雇员的姓名及年薪。
6. 查看每月可以得到奖金的雇员信息。
7. 基本工资大于 1500，同时可以领取奖金的雇员信息。
8. 查询基本工资不大于 1500，同时不可以领取奖金的雇员信息。
9. 查询在 1981 年雇佣的全部雇员信息，BETWEEN .. AND 包含等于的情况。
10. 查询出雇员编号不是 7369、7499 的雇员信息。
11. 查看雇员编号不是 7369 的雇员信息，使用<>或!=。
12. 对雇员的工资由低到高进行排序，升序为默认(ASC)，降序(DESC)。
13. 查看出部门号为 10 的雇员信息，查询的信息按照工资从高到低，若工资相等则按雇用日期从早到晚排列。
14. 将雇员姓名变为开头字母大写，INITCAP。
15. 显示所有雇员的姓名及姓名的后 3 个字符。
16. 查询从雇用日期到今天所有雇员的雇员编号、姓名和月数。
17. 找出佣金高于薪金的 60%的员工。
18. 找出部门 10 中所有经理(MANAGER)和部门 20 中所有办事员(CLERK)的详

细资料。

19. 找出既不是经理又不是办事员但其薪金大于或等于2000的所有员工的资料。
20. 找出有奖金的员工的不同工作。
21. 找出各月倒数第3天受雇的所有员工。
22. 找出早于12年前受雇的员工。
23. 显示刚好为5个字符的员工的姓名。
24. 显示不带有"R"的员工的姓名。
25. 显示员工的姓名和受雇日期，将最老的员工排在最前。
26. 显示所有员工的姓名，加入公司的年份和月份，按受雇日期所在月排序，若月份相同则按年份排序。
27. 找出在2月受聘的员工信息。
28. 以年月日方式显示所有员工服务年限。
29. 找出EMP表中的姓名（ENAME）第三个字母是A 的员工姓名。
30. 找出EMP表员工名字中含有A 和N的员工姓名。
31. 显示工资不在1000 到1500 之间的员工信息：名字、工资，按工资从大到小排序。
32. 显示职位为MANAGER 和SALESMAN，年薪在15000 和20000 之间的员工的信息：名字、职位、年薪。
33. 说明以下两条SQL语句的输出结果：
    1) SELECT EMPNO,COMM FROM EMP WHERE COMM IS NULL;
    2) SELECT EMPNO,COMM FROM EMP WHERE COMM = NULL;

实验七　　多表查询

实验保存为学号-实验-7.doc 文件提交。

【实验目标】

- 编写 SELECT 语句，通过使用相等和非相等连接，用于访问多个表的数据；
- 通过使用外连接，查找不能满足连接条件的数据；
- 自连接。

概念

- 连接用于从多表中查询数据；
- 通过使用公共值将记录行进行连接，最典型的是使用主键和外键值；
- 连接方法：等值连接、非等值连接、外连接、自连接、集合操作符。

笛卡尔乘积

- 笛卡尔乘积的形成：
  - 连接条件被省略；
  - 连接条件无效；
  - 第一个基表的所有行被连接到第二个基表的所有行。
- 为了避免笛卡尔乘积，总是在 WHERE 子句中使用有效的连接条件。

简单连接查询语法格式：

```
SELECT table.column, table.column
    FROM tale1, table2
    WHERE table1.column1=table2.column2;
```

- 在 WHERE 子句中指定连接条件；
- 为了明确，在每个列名前使用表名；
- 当多个表中有相同的列名时，表名必须作为列名的前缀。
- 使用表前缀可以提高性能；
- 对于不同基表中，具有相同名字的列可以使用别名加以区别。

基表别名

- 使用表别名限定列；
- 只有对 SELECT 语句有效；
- 一旦创建了表别名，就使用表别名代替表名限定列。

例：显示所有员工的基本信息，包括部门名称和工作地址

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

例：查询工作地址在 CHICAGO 的员工的情况

```
SELECT empno, ename, job, d.deptno, dname, d.loc
```

```
FROM emp e, dept d
WHERE e.deptno=d.deptno and loc='CHICAGO';
```

非等值连接

- 当一个表中没有列与另一个表中的列相匹配，就导致使用非等值连接；
- 连接条件中包含了除等于号(=)之外的操作符。

例：查询工资高于 2000 的员工及所在部门的情况

```
SELECT ename, sal, e.deptno, d.dname, loc
    FROM emp e, dept d
    WHERE d.deptno=e.deptno AND sal>2500
    ORDER BY e.deptno;
```

例：查询每个员工的工资级别

```
SELECT empno, ename, job, sal, deptno, grade
    FROM emp, salgrade
    WHERE sal BETWEEN losal AND hisal
    ORDER BY deptno;
```

外连接

- 使用外连接查看那些通常不能满足连接条件的记录行；
- 外连接操作符为加号(+)；
- 将操作符放在没有值可以连接的一侧；
- 包括外连接的条件不可以：
  - 使用 IN 操作符；
  - 使用 OR 操作符被连接到另一个条件。

例：查询每个部门的员工人数

```
SELECT d.deptno, count(e.rowid) "Employee Numbers"
    FROM dept d, emp e
    Where d.deptno=e.deptno (+)
    GROUP BY d.deptno
    ORDER BY d.deptno;
```

例：查询暂时没有员工的部门

```
SELECT DISTINCT dept.* FROM dept, emp
    WHERE dept.deptno=emp.deptno(+) AND empno IS NULL;
```

自连接

- 使用自连接来连接同一个基表的不同数据行；
- 一个基表用作自联接时，就如同是两个独立的基表；
- 在 FROM 子句中，至少要为一个基表起别名，以区别联接条件中的列名。

例：查询每个员工的经理名字；

```
SELECT e.ename ||' work   for '|| m.ename
    FROM emp e, emp m
```

WHERE e.mgr=m.empno;

例：查询工资超过其经理工资的员工名字；

```
SELECT e.ename Employee, e.sal “Employee’s Salary”,
        m.ename Manager, m.sal “Manager’s Salary”
FROM emp e, emp m
WHERE e.mgr=m.empno AND e.sal>m.sal;
```

作业三 多表查询

作业保存为学号-3.sql 文件提交

在作业一的基础上，完成如下题目。并将每个实验结果均存储为一张视图表。方式如下：

create view T3_1.* [(..)]

AS

(Select ……);//查询语句

1. 要求查询雇员的编号、姓名、部门编号、部门名称及部门位置。**（其视图名为T3_1,后面的题目以此类推）**
2. 要求查询每个雇员的姓名、工作、雇员的直接上级领导的姓名(表自关联)。
3. 对题目 2 进行扩充，将雇员所在部门名称同时列出。
4. 查询每个雇员的姓名、工资、部门名称，工资在公司的等级(salgrade)，及其领导的姓名所在公司的等级。
5. 让 SELECT TO_CHAR(SALARY,'L99,999.99') FROM HR.EMPLOYEES WHERE ROWNUM < 5 输出结果的货币单位是￥和$。
6. 列出前五位每个员工的名字，工资、涨薪后的的工资（涨幅为 8%），以“元”为单位进行四舍五入。
7. 找出谁是最高领导，将名字按大写形式显示。
8. 找出 First_Name 为 David，Last_Name 为 Austin 的直接领导名字。
9. First_Name 为 Alexander，Last_Name 为 Hunold 领导谁。（谁向 David 报告）。
10. 哪些员工的工资高于他直接上司的工资，列出员工的名字和工资，上司的名字和工资。
11. 哪些员工和 Chen(LAST_NAME)同部门。
12. 哪些员工跟 De Haan(LAST_NAME)做一样职位。
13. 哪些员工跟 Hall(LAST_NAME)不在同一个部门。
14. 哪些员工跟 William（FIRST_NAME）、Smith(LAST_NAME)做不一样的职位。
15. 显示有提成的员工的信息：名字、提成、所在部门名称、所在地区的名称。
16. 显示 Executive 部门有哪些职位。

17. 整个公司中，最高工资和最低工资相差多少。
18. 提成大于 0 的人数。
19. 显示整个公司的最高工资、最低工资、工资总和、平均工资保留到整数位。
20. 整个公司有多少个领导。
21. 列出在同一部门入职日期晚但工资高于其他同事的员工：名字、工资、入职日期。

实验八　　组函数

实验保存为学号-实验-8.doc 文件提交。

【实验目标】

- 确定可以使用的组函数；
- 解释组函数的使用；
- 使用 GROUP BY 子句显示不同组的统计结果；
- 使用 HAVING 子句包含或排除分组的记录行。

语句格式：

```
SELECT column, group-function
    FROM table
    [ WHERE condition ]
    [ GROUP BY group-by-expression ]
    [ HAVING group-condition ]
    [ ORDER BY column ];
```

功能介绍

- 对一组纪录计算，每组返回一个计算值
- 可以在 SELECT、HAVING 子句中使用组函数
- 在查询语句中使用 GROUP BY 子句将记录分成组
- 使用 HAVING 子句作为组函数的条件

组函数（聚集函数）

- AVG (DISTINCT|ALL|*n*)　　平均值
- COUNT (DISTINCT|ALL|*expr*|*)　　计算个数
- MAX (DISTINCT|ALL|*expr*)　　最大值
- MIN (DISTINCT|ALL|*expr*)　　最小值
- STDDEV (DISTINCT|ALL|*n*)　　标准差
- SUM (DISTINCT|ALL|*n*)　　求和
- VARIANCE (DISTINCT|ALL|*n*)　　方差

- COUNT(*) 返回表中的记录数
- COUNT(*expr*) 返回非空记录数

例：

```
SELECT COUNT(*) FROM emp;
SELECT COUNT(comm) FROM emp;
SELECT deptno, COUNT(*) FROM emp GROUP BY deptno;
SELECT deptno, MIN(sal), MAX(sal), AVG(sal), SUM(sal)
    FROM emp GROUP BY deptno;
```

- MAX、MIN 函数可以用在任意类型的列上

SELECT deptno, MIN(job), MAX(job) FROM emp GROUP BY deptno;

注：所有出现在 SELECT 子句中的列必须写在 GROUP BY 子句中，用于分组的列可以不出现在 SELECT 子句中。可以使用 ORDER BY 子句改变结果的排列顺序

多层分组

- 可以按多个分组标准进行分组
- 同时返回主组和从组的结果
- 根据在 GROUP BY 子句中的列的先后次序，决定结果的显示次序

例：

```
SELECT deptno, job, COUNT(*), MAX(sal), MIN(sal),AVG(sal), SUM(sal)
    FROM emp GROUP BY deptno, job;
```

条件分组

- 只能使用 HAVING 子句作为组函数限制条件
- 不能使用 WHERE 子句作为组函数的限制条件

HAVING 子句用来定义满足一定条件的分组，是作为分组函数的限制条件。它与 WHERE 子句类似，但有所不同，WHERE 子句的条件是针对 SELECT 子句的，而 HAVING 子句的条件是针对 GROUP BY 子句的。

例：查询工资总和超过 9000 的部门

```
SELECT deptno, SUM(sal) FROM emp
    GROUP BY deptno HAVING SUM(sal)>9000;
```

例：查询平均工资超过 2500 的工种

```
SELECT job, COUNT(*), AVG(sal) FROM emp
    GROUP BY job HAVING AVG(sal)>=2500;
```

例：除去职员，哪些部门的工资总和超过了 8000，不包括奖金，并且按工资升序显示：

```
SELECT deptno, SUM(sal) FROM emp
    WHERE LOWER(job)!='clerk'
    GROUP BY deptno
    HAVING SUM(sal)>8000
    ORDER BY SUM(sal);
```

作业四 组查询

作业保存为学号-4.sql 文件提交

在作业一的基础上，完成如下题目。并将每个实验结果均存储为一张视图表。方式如下：

```
create view T4_* [(..)]
        AS
(Select ……);//查询语句
```

1. 列出至少有一个员工的所有部门。**（其视图名为 T4_1,后面的题目以此类推）**
2. 列出薪金比“SMITH”多的所有员工。
3. 列出所有员工的姓名及其直接上级的姓名。
4. 列出受雇日期早于其直接上级的所有员工。
5. 列出部门名称和这些部门的员工信息，同时列出那些没有员工的部门。
6. 列出所有“CLERK”（办事员）的姓名及其部门名称。
7. 列出最低薪金大于 1500 的各种工作。
8. 列出在部门“SALES”（销售部）工作的员工的姓名，假定不知道销售部的部门编号。
9. 列出薪金高于公司平均薪金的所有员工。
10. 列出与“SCOTT”从事相同工作的所有员工。
11. 列出薪金等于部门 30 中员工的薪金的所有员工的姓名和薪金。
12. 列出薪金高于在部门 30 工作的所有员工的薪金的员工姓名和薪金。
13. 列出在每个部门工作的员工数量、平均工资和平均服务期限。
14. 列出所有员工的姓名、部门名称和工资。
15. 列出所有部门的详细信息和部门人数。
16. 列出各种工作的最低工资。
17. 列出各个部门的 MANAGER（经理）的最低薪金。
18. 列出所有员工的年工资,按年薪从低到高排序。
19. 各个部门平均、最大、最小工资、人数，按照部门号升序排列。

20. 各个部门中工资大于 5000 的员工人数。

21. 各个部门平均工资和人数，按照部门名字升序排列。

22. 列出每个部门中有同样工资的员工的统计信息，列出他们的部门号，工资，人数。

23. 列出同部门中工资高于 1000 的员工数量超过 2 人的部门，显示部门名字、地区名称。

24. 哪些员工的工资，高于整个公司的平均工资，列出员工的名字和工资（降序）。

25. 哪些员工的工资，介于 50 号 和 80 号部门平均工资之间。

26. 所在部门平均工资高于 5000 的员工名字。

27. 列出各个部门中工资最高的员工的信息：名字、部门号、工资。

28. 最高的部门平均工资是多少。

注：个别题目的排序结果在视图中没办法实现，请再排序的结果无误的情况下，将未排序的结果生成视图提交到服务器中。

实验九　　子查询

实验保存为学号-实验-9.doc 文件提交。

【实验目标】

- 对于未知的条件使用嵌套子查询查询数据；
- 在数据操纵语言中使用子查询；
- 使用子查询排序。

定义：内嵌在其它 SQL 语句中的 SELECT 语句。

子查询语法格式：

```
SELECT select-list
    FROM table-name
    WHERE expression operator
        (SELECT select-list FROM table-name);
```

其中：operator 为=时，要求子查询语句返回一个值；否则，使用 IN。

- 子查询在主查询执行前执行一次
- 主查询利用子查询的结果

子查询的规则

- 包含在其它查询中；
- 子查询可以使用两类操作符：单行(=)及多行(IN)；
- 写在操作符的右侧；
- 可以用在许多 SQL 命令中；
- 子查询不能包含 ORDER BY 子句。

例：查询 SCOTT 的同事，即同一个部门的员工；

```
SELECT ename, deptno FROM emp WHERE deptno=
    (SELECT deptno FROM emp WHERE LOWER(ename)='scott');
```

等价于：

```
SELECT e1.ename, e1.deptno
    FROM emp e1, emp e2
    WHERE e1.deptno=e2.deptno AND LOWER(e2.ename)='scott';
```

例：查询工资低于平均工资的员工；

```
SELECT deptno,ename,job,sal FROM emp
    WHERE sal<(SELECT AVG(sal) FROM emp);
```

例：查询平均工资高于 20 号部门的平均工资的部门；

```
SELECT deptno,AVG(sal) FROM emp GROUP BY deptno
    HAVING AVG(sal)>
        (SELECT AVG(sal) FROM emp WHERE deptno=20);
```

例：查询没有员工的部门；

```
SELECT dept.deptno, dept.dname, dept.loc FROM dept
    WHERE dept.deptno NOT IN
        (SELECT DISTINCT emp. deptno FROM emp);
```

带有子查询的 HAVING 子句

- HAVING 子句可以带有子查询；
- ORACLE 服务器首先执行子查询；
- 将结果带给主查询的 HAVING 子句。

例：查询平均工资大于 20 部门平均工资的部门；

```
SELECT deptno, AVG(sal) FROM emp
    GROUP BY deptno
    HAVING AVG(sal)>
        (SELECT AVG(sal) FROM emp WHERE deptno=20);
```

不相关子查询

是指 WHERE 子句中没有引用父查询的查询列，因此，不相关子查询的求解过程很简单，即可以先求出子查询的结果，再求解父查询。

例：查询 SCOTT 的同事，即同一个部门的员工；

```
SELECT ename, deptno FROM emp WHERE deptno=
    (SELECT deptno FROM emp WHERE LOWER(ename)='scott');
```

相关子查询

是指 WHERE 子句中引用父查询的查询列。

例：查询 SCOTT 的同事，即同一个部门的员工；

```
SELECT ename, deptno FROM emp e1
    WHERE 'SCOTT' IN
        (SELECT ename FROM emp WHERE e1.deptno=deptno);
```

例：查询工资高于本部门平均工资的员工名单；

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

带有 ANY, ALL 的子查询

当子查询的结果不是单值时，不能用比较符，但若和 ANY、ALL 组合使用时，则是可以的。

"> ANY"的语义是大于子查询结果中的任何一个值，而"> ALL"的语义是大于子查询结果中的所有的值，同理对其它比较符也是如此。

例：查询其他部门中比 30 号部门任一员工工资高的员工名单；

```
SELECT ename, deptno, sal FROM emp
        WHERE deptno<>30 AND sal > ANY
            (SELECT sal FROM emp WHERE deptno=30)
```

```
        ORDER BY deptno;
```
等价于：
```
    SELECT ename, deptno, sal FROM emp
        WHERE deptno<>30 AND sal >
            (SELECT MIN(sal) FROM emp WHERE deptno=30)
        ORDER BY deptno;
```

例：查询其他部门中比 30 号部门所有员工工资高的员工名单；
```
    SELECT ename, deptno, sal FROM emp
        WHERE deptno<>30 AND sal > ALL
            (SELECT sal FROM emp WHERE deptno=30)
        ORDER BY deptno;
```
等价于：
```
    SELECT ename, deptno, sal FROM emp
        WHERE deptno<>30 AND sal >
            (SELECT MAX(sal) FROM emp WHERE deptno=30)
        ORDER BY deptno;
```

使用存在量词 EXISTS 的子查询

例：查询与 SCOTT 具有同一经理的员工名单；
```
    SELECT empno, ename, deptno, mgr FROM emp e1
        WHERE EXISTS
            (SELECT * FROM emp
                WHERE e1.mgr=mgr AND ename='SCOTT');
```

其他形式的子查询

例：查询与 SCOTT 具有相同工资和职务的员工姓名、职务和工资；
```
    SELECT ename, job, sal, deptno FROM emp
        WHERE (job, sal)=
            (SELECT job, sal FROM emp WHERE ename='SCOTT');
```

例：查询与 CLARK 具有相同职务或工资比 CLARK 高的员工姓名、职务和工资；
```
    SELECT ename, job, sal, deptno FROM emp
        WHERE job=
            (SELECT job FROM emp WHERE ename='CLARK')
        OR sal>
            (SELECT sal FROM emp WHERE ename='CLARK')
        ORDER BY deptno;
```

例：查询工资比 SCOTT 高且在 NEW YORK 工作的员工姓名、职务和工资；
```
    SELECT ename, job, sal, emp.deptno, loc
        FROM emp, dept
        WHERE emp.deptno=dept.deptno
        AND loc='NEW YORK'
```

```
AND sal>
    (SELECT sal FROM emp WHERE ename='SCOTT')
ORDER BY emp.deptno;
```

子查询小结

子查询是 SQL 语句中极其重要的概念，还可以用在以下语句：

CREATE TABLE AS <子查询>

CREATE VIEW AS <子查询>

SELECT … WHERE <子查询>

HAVING <子查询>

INSERT INTO … <子查询>

UPDATE … SET <子查询> WHERE <子查询>

作业五 子查询

作业保存为学号-5.sql 文件提交

在作业一的基础上，完成如下题目。并将每个实验结果均存储为一张视图表。方式如下：

```
create view T5_* [(..)]
        AS
    (Select ……);//查询语句
```

1. 哪些部门的人数比 90 号部门的人数多。**（其视图名为 T5_1,后面的题目以此类推）**
2. Den(FIRST_NAME)、Raphaely(LAST_NAME)的领导是谁（非关联子查询）。
3. Den(FIRST_NAME)、Raphaely(LAST_NAME) 领导谁（非关联子查询）。
4. Den(FIRST_NAME)、Raphaely(LAST_NAME) 的领导是谁（关联子查询）。
5. Den(FIRST_NAME)、Raphaely(LAST_NAME) 领导谁（关联子查询）。
6. 列出在同一部门共事，入职日期晚但工资高于其他同事的员工：名字、工资、入职日期
7. （关联子查询）。
8. 哪些员工跟 Den(FIRST_NAME)、Raphaely(LAST_NAME)不在同一个部门（非关联子查询）。
9. 哪些员工跟 Den(FIRST_NAME)、Raphaely(LAST_NAME)不在同一个部门（关联子查询）。
10. Finance 部门有哪些职位（非关联子查询）。
11. Finance 部门有哪些职位（关联子查询）。

实验十　创建视图

实验保存为学号-实验-10.doc 文件提交。

【实验目标】

- 解释视图的概念；
- 使用数据字典视图；
- 创建简单和复杂视图；
- 创建带有强制约束选项的视图；
- 修改视图；
- 删除视图。

概念

视图是建立在一个或多个数据基表的逻辑映像。

优越性

- 限制数据访问
- 简化查询
- 增强数据独立性
- 同类数据不同显示

语法格式

- 在 CREATE VIEW 语句中加入子查询语句

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW view
    [(alias[, alias]...)]
    AS sub-query
    [WITH CHECK OPTION [CONSTRAINT constraint ]]
    [WITH READ ONLY]
```

- 子查询可以是复杂的查询语句.
- 子查询中不能含有 ORDER BY 子句.

简单视图和复杂视图的比较

	简单视图	复杂视图
基表数	One	One or More
包含函数	No	Yes
包含组数	No	Yes
视图上的 DML 命令	Yes	No

例： 创建一个视图，包含 emp 表中的 empno, ename, deptno 的列，部门编号为 30;

```
CREATE VIEW emp_view_30 AS
    SELECT empno, ename, deptno
        FROM emp WHERE deptno=30;
```

- 可以使用 SQL*PLUS 中的 DESCRIBE 命令显示视图的结构

DESCRIBE emp_view_30;

- 可以使用 SELECT 命令查询视图中的数据

SELECT * FROM emp_view_30;

- 创建视图时使用子查询中列的别名.

```
CREATE VIEW emp_view_10 AS
    SELECT ename name, sal salary, comm commission
        FROM emp WHERE deptno=10;
```

例：由于在子查询语句中不允许使用 ORDER BY 子句，可以使用 GROUP BY 子句代替。

```
CREATE OR REPLACE VIEW emp_order AS
    SELECT deptno, job, ename FROM emp
        GROUP BY deptno, job, ename;
```

修改视图

使用 CREATE OR REPLACE 命令修改视图

- 使用 CREATE OR REPLACE 修改 emp_view_10 视图，为每个列增加别名。
- 在 CREATE VIEW 子句的别名与子查询中的列名顺序相同。

```
CREATE OR REPLACE VIEW emp_view_10(id, name, salary) AS
    SELECT empno, ename, sal
        FROM emp WHERE deptno=20;
```

创建复杂视图

- 使用组函数创建视图

```
CREATE OR REPLACE VIEW emp_salary_view AS
    SELECT deptno, job, COUNT(*) Total_Employee,
        MAX(sal) Max_salary, MIN(sal) Min_salary,
        AVG(sal) Avg_salary, SUM(sal) Sum_salary
        FROM emp GROUP BY deptno, job;
```

- 创建视图，用于显示两个基表的数据

```
CREATE OR REPLACE VIEW emp_dept_view AS
    SELECT empno, ename, job, e.deptno, dname, loc
        FROM emp e, dept d
        WHERE e.deptno=d.deptno;
```

DML 操作的规则

- 在简单视图上可以进行 DML 操作

若视图中含有下列内容，则不能删除

- 组函数
- GROUP BY 子句
- DISTINCT 命令

含有下列内容时，不能修改数据

- 上述情况

- 表达式定义的列
- ROWID

含有下列内容时，不能插入数据

- 上述情况
- 视图中未选中 NOT NULL 列

- WITH CHECK OPTION 子句确保视图上的 DML 操作只对视图有效.

```
CREATE OR REPLACE VIEW emp_view AS
    SELECT * FROM emp
        WITH CHECK OPTION CONSTRAINT emp_view_ck;
```

- 在有 WITH READ ONLY 选项的视图上，不可以执行 DML 命令

```
CREATE OR REPLACE VIEW emp_view_10(id, name, salary) AS
    SELECT empno, ename, sal
        FROM emp WHERE deptno=10 WITH READ ONLY;
```

数据字典 USER_VIEWS 包含了视图的名称和视图的定义。

```
SELECT view_name, text FROM user_views;
```

删除视图

删除视图，但不会丢失任何数据

```
DROP VIEW emp_view;
```

实验十一　创建索引

实验保存为学号-实验-11.doc 文件提交。

【实验目标】

- 理解自动创建和手工创建索引的区别；
- 确定索引的应用；
- 解释索引结构和提高查询速度的原因；
- 创建非唯一性索引；
- 从数据字典中删除索引；
- 指定创建和使用索引的规则。

概念

- 数据库对象
- 加速查询
- 减少磁盘 I/O
- 与基表相互独立
- 数据库自动使用和维护

创建方式

自动：定义基表时定义的主键和唯一性约束，系统会自动建立唯一性索引；
手动：用户可以建立非唯一性索引以提高访问速度。

索引结构

- ORACLE 中使用 B+树存放索引
- 每个索引都包含列值和指针
- Server 沿着树枝进行搜索

类型

- Unique：保证列值唯一
- Non-unique：加速查询
- Single column：在索引中使用单列
- Concatenated or composite：在索引中使用多列

语法格式：
在一列或多列上创建索引

```
CREATE INDEX index ON table ( column [ , column …];
```

例：　在 emp　表中，通过 ename　列提高数据查询的速度；

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

规则

- 经常在 WHERE 子句中使用的列
- 列值很多

- 列值中含有大量空值
- 两列或多列经常出现在 WHERE 子句中
- 基表数据很多而查询结果通常返回小于 2-4%的记录
- 太多的索引并不能提高速度

确认索引

- 从 USER_INDEXES 中可以查到索引的信息.
- 从 USER_IND_COLUMNS 可以查到索引名，表名和列名

```
SELECT ic.index_name, ic.column_name,
        ic.column_position col_pos, ix.uniqueness
    FROM user_indexes ix, user_ind_columns ic
    WHERE ic.index_name = ix.index_name
    AND ic.table_name = 'EMP';
```

删除索引

- 从数据字典中删除索引

```
DROP INDEX emp_index_ename;
```

- 若要删除索引，必须是索引的创建者或者具有 DROP ANY VIEW 的权限.

基础训练部分上机考试（期中考试）

【考试内容】

在学生将作业 1 至 5 完整提交到服务器的基础上，现场给出考试题目，在规定时间内 15-20 分钟完成，由指导教师现场给出成绩，分值为 0-100 分。

【要求和说明】

考试当场给出题目，1 人 1 题，考试期间不准学生交头接耳或同学间互传题目、用 U 盘拷贝等现象，一经发现，成绩按零分计，无补考机会。学生考试一经结束，迅速离场。