



大型数据库应用技术

04-SQL和PL/SQL

授课教师：王欢

准备知识



注册表编辑器

文件(E) 编辑(E) 查看(V) 收藏夹(A) 帮助(H)

计算机\HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\KEY_OraDB12Home1

	名称	类型	数据
> DefaultUserEnviron ^			
> Google	ab MSHELP_TOOLS	REG_SZ	D:\app\oracle12c\product\12.1.0\dbhome_1\M...
> HaoZip	ab NLS_LANG	REG_SZ	SIMPLIFIED CHINESE_CHINA.ZHS16GBK
> IM Providers	ab OLEDB	REG_SZ	D:\app\oracle12c\product\12.1.0\dbhome_1\ol...
> Intel	ab OMTSRECO_PO...	REG_EXPAND_SZ	2030
> Khronos	ab ORA_ORCL_AU...	REG_EXPAND_SZ	TRUE
> Macromedia	ab ORA_ORCL_SH...	REG_EXPAND_SZ	TRUE
> Microsoft	ab ORA_ORCL_SH...	REG_EXPAND_SZ	90
> MozillaPlugins	ab ORA_ORCL_SH...	REG_EXPAND_SZ	immediate
> Nuance	ab ORA_SSS_AUT...	REG_EXPAND_SZ	TRUE
> ODBC	ab ORA_SSS_SHUT...	REG_EXPAND_SZ	TRUE
> OEM	ab ORA_SSS_SHUT...	REG_EXPAND_SZ	90
▼ ORACLE	ab ORA_SSS_SHUT...	REG_EXPAND_SZ	immediate
> KEY_OraDB12Ho	ab ORACLE_BASE	REG_SZ	D:\app\oracle12c
> ODP.NET	ab ORACLE_BUND...	REG_SZ	Enterprise
OracleMTSRecov	ab ORACLE_GROU...	REG_SZ	Oracle - OraDB12Home1
> Partner	ab ORACLE_HOME	REG_SZ	D:\app\oracle12c\product\12.1.0\dbhome_1
> Policies	ab ORACLE_HOME...	REG_SZ	SOFTWARE\ORACLE\KEY_OraDB12Home1
> Primax	ab ORACLE_HOME...	REG_SZ	OraDB12Home1
> Realtek	ab ORACLE_HOME...	REG_SZ	1
RegisteredApplicat	ab ORACLE_SID	REG_SZ	orcl
SonicFocus	ab ORACLE_HOME...	REG_SZ	1
SoundResearch	ab ORACLE_SVCUS...	REG_SZ	oracle12c



□ 启用 SCOTT 和 HR 用户

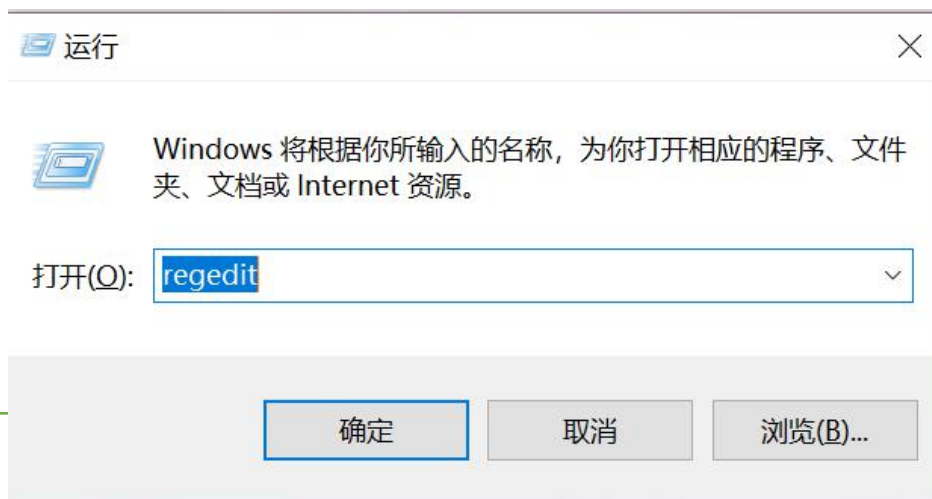
更改数据库当前实例

1 Windows 命令行

```
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 10.0.17763.737]
(c) 2018 Microsoft Corporation。保留所有权利。

C:\Users\hp>set ORACLE_SID=orcl
C:\Users\hp>
```

2 注册表





□ 四个用户

- **SCOTT** : Oracle 便于用户学习所提供的内置用户。
- **HR** : 一个示例用户，是在安装Oracle，创建数据库时选中“示例数据库”后产生的，实际就是模拟一个人力资源部的数据库。
- **SYS**: 所有 Oracle 的数据字典的基表和视图都存放在 sys 用户中，这些基表和视图对于 Oracle 的运行至关重要，由数据库自己维护，任何用户都不能手动更改。sys用户拥有 dba, sysdba, sysoper 等角色或权限，是 Oracle 权限最高的用户。
- **SYSTEM**: 用于存放次一级的内部数据，如 Oracle 的一些特性或工具的管理信息。system 用户拥有普通 dba 角色权限。



□ 启用 SCOTT 和 HR 用户

Oracle12c 中，SCOTT 和 HR 用户在默认安装的 ORCL 数据库自带的 PDB (PDBORCL或ORCLPDB) 中。

1. 使用 SYS 用户连接到 PDB (先打开，再连接)；
2. 查看 SCOTT 和 HR 的状态，默认为密码过期和锁定状态；

```
select username, account_status  
from dba_users;
```

	USERNAME	ACCOUNT_STATUS
6	HR	EXPIRED & LOCKED
7	SCOTT	EXPIRED & LOCKED

3. 解锁账号，设置密码；

```
alter user scott(hr) account unlock;  
alter user scott(hr) identified by newpassword;
```

4. 使用用户名和新密码连接数据库。



C:\Windows\system32\cmd.exe - sqlplus / as sysdba

C:\Users\hp>set ORACLE_SID=orcl

C:\Users\hp>sqlplus / as sysdba

SQL*Plus: Release 12.1.0.2.0 Production on 星期日 10月 27 22:16:19 2019

Copyright (c) 1982, 2014, Oracle. All rights reserved.

连接到:

Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing option

SQL> show pdbs

CON_ID	CON_NAME	OPEN MODE	RESTRICTED
2	PDB\$SEED	READ ONLY	NO
3	PDBORCL	MOUNTED	

SQL> alter pluggable database pdborcl open;

插接式数据库已变更。

SQL>



SQL> alter session set container=pdborcl; **SQL*PLUS只能连接一个数据库**
会话已更改。

SQL> select username, account_status from dba_users where username='SCOTT';

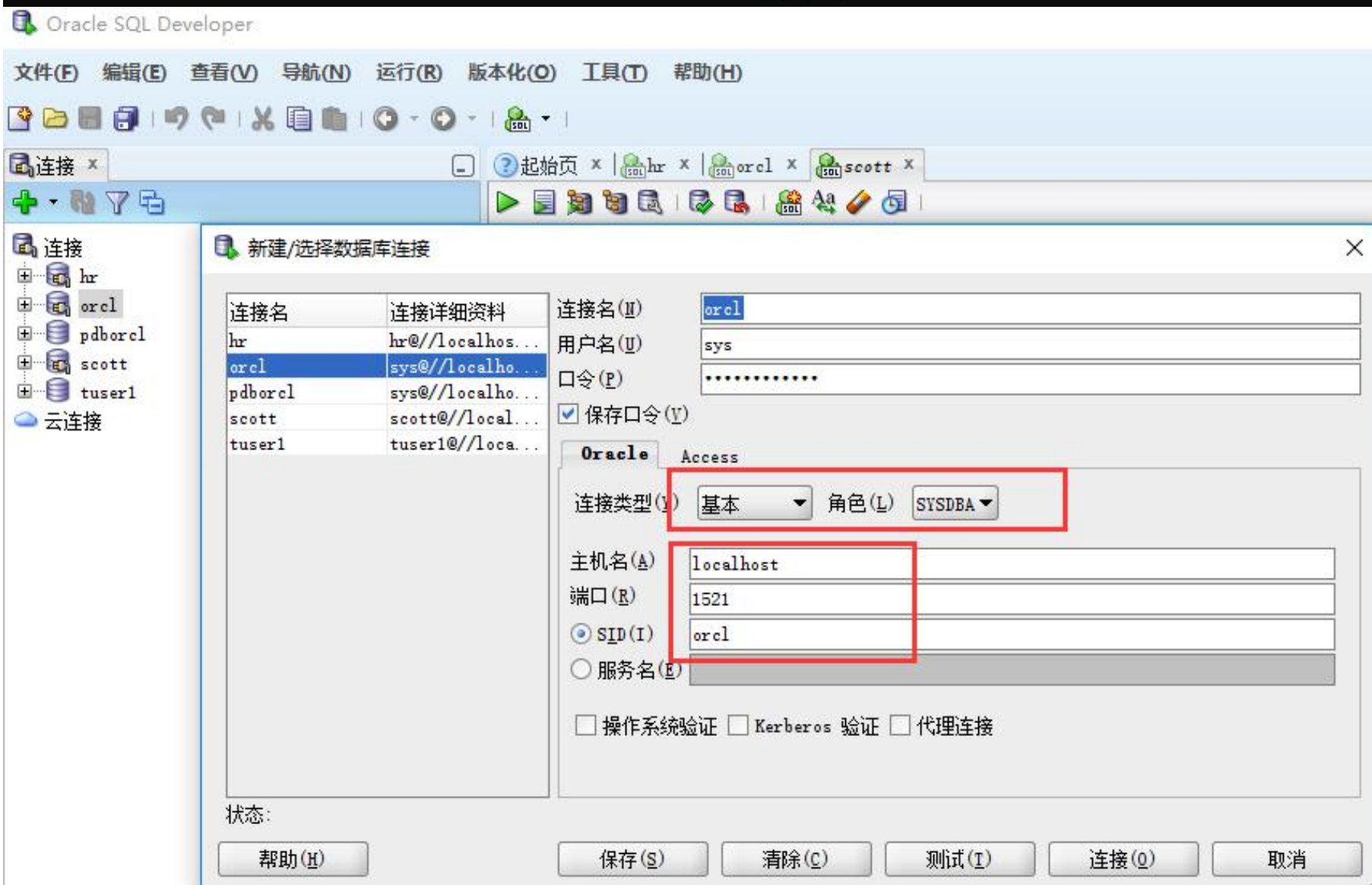
USERNAME

ACCOUNT_STATUS

SCOTT
OPEN

SQL> alter session set container=pdborcl;
会话已更改。

SQL*PLUS只能连接一个数据库





- 连接
- HR
- ORCL
- PDBORCL
- SCOTT
- 云连接

新建/选择数据库连接

连接名	连接详细资料
HR	hr@//localhos...
ORCL	sys@//localho...
PDBORCL	sys@//localho...
SCOTT	scott@//local...

连接名(N) SCOTT

用户名(U) scott

口令(P)

☐ 保存口令(V)

Oracle Access

连接类型(T) 基本 角色(L) 默认值

主机名(A) localhost

端口(R) 1521

☐ SID(I)

☒ 服务名(E) pdborcl

☐ 操作系统验证 ☐ Kerberos 验证 ☐ 代理连接

状态:

帮助(H)

保存(S)

清除(C)

测试(T)

连接(O)






取消



□ SCOTT 用户






部门表 DEPT

部门编号
部门名称
部门位置

 COLUMN_NAME	 DATA_TYPE	 NULLABLE	DATA_DEFAULT	 COLUMN_ID	 COMMENTS
DEPTNO	NUMBER(2,0)	No	(null)	1 (null)	
DNAME	VARCHAR2(14 BYTE)	Yes	(null)	2 (null)	
LOC	VARCHAR2(13 BYTE)	Yes	(null)	3 (null)	

雇员表 EMP

雇员编号
雇员姓名
工作岗位
领导编号
雇佣日期
基本工资
奖金
部门编号

 COLUMN_NAME	 DATA_TYPE	 NULLABLE	DATA_DEFAULT	 COLUMN_ID	 COMMENTS
EMPNO	NUMBER(4,0)	No	(null)	1 (null)	
ENAME	VARCHAR2(10 BYTE)	Yes	(null)	2 (null)	
JOB	VARCHAR2(9 BYTE)	Yes	(null)	3 (null)	
MGR	NUMBER(4,0)	Yes	(null)	4 (null)	
HIREDATE	DATE	Yes	(null)	5 (null)	
SAL	NUMBER(7,2)	Yes	(null)	6 (null)	
COMM	NUMBER(7,2)	Yes	(null)	7 (null)	
DEPTNO	NUMBER(2,0)	Yes	(null)	8 (null)	



□ SCOTT 用户

工资等级表 SALGRADE

工作等级
等级内最低工资
等级内最高工资

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
GRADE	NUMBER	Yes	(null)	1	(null)
LOSAL	NUMBER	Yes	(null)	2	(null)
HISAL	NUMBER	Yes	(null)	3	(null)

工资表 BONUS

雇员姓名
工作岗位
基本工资
奖金

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
ENAME	VARCHAR2(10 BYTE)	Yes	(null)	1	(null)
JOB	VARCHAR2(9 BYTE)	Yes	(null)	2	(null)
SAL	NUMBER	Yes	(null)	3	(null)
COMM	NUMBER	Yes	(null)	4	(null)



□ HR 用户

- **国家表 COUNTRIES**: 国家编号、国家名称、地区编号。
- **部门表 DEPARTMENTS**: 部门编号、部门名称、经理编号、地点编号。
- **雇员表 EMPLOYEES**: 雇员编号、姓氏、名字、电子邮件、电话号码、雇佣日期、工作编号、工资、佣金比、经理编号、部门编号。
- **地区表 REGIONS**: 地区编号、地区名称。
- **工作表 JOBS**: 工作编号、职称、最低工资、最高工资。
- **工作历史表 JOB_HISTORY**: 雇员编号、开始时间、结束时间、工作编号、部门编号。
- **地点表 LOCATIONS**: 地点编号、街道地址、邮政编码、城市、州省、国家编号。
- **员工详细视图 EMP_DETAILS_VIEW**: 雇员编号、工作编号、经理编号、部门编号、地点编号、国家编号、姓氏、名字、工资、佣金比、部门名称、职称、城市、州省、国家名称、地区名称。



□ SQL

□ SQL概述

□ 数据定义

□ 数据查询

□ 数据操纵

□ 数据控制



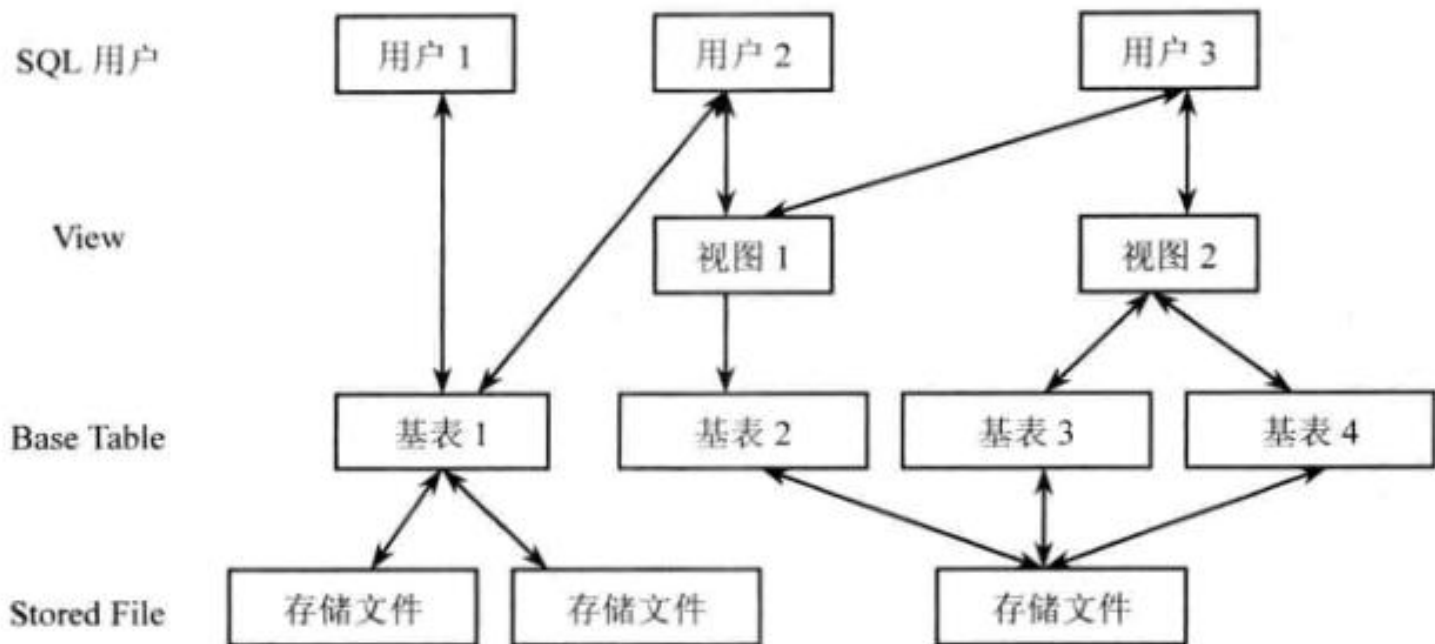
□ SQL 概述

SQL（Structure Query Language，结构化查询语言）是数据库操作的国际标准语言，也是所有的数据库产品均要支持的语言。

- 1974 年由美国 IBM 公司的 Boyce 和 Chamberlin 提出。
- 1975~1979 年在关系数据库的管理系统原型 System R 初次实现。
- 1986 年美国国家标准局（ANSI）批准 SQL 作为美国标准，公布了 SQL_86。1987 年国际标准化组织（ISO）将其采纳为国际标准。
- 1989 年——SQL_89； 1992 年——SQL_92（也称为 SQL2）； 1999 年——SQL_99（也称为 SQL3）。
- 不同的数据库管理系统厂商开发的 SQL 并不完全相同，均在标准 SQL 的基础上进行了扩展，增强了一些功能；Oracle——PL/SQL；Microsoft SQL Server——Transact-SQL。



SQL 支持三级模式结构



- 基表 (Base Table) 是本身独立存在的。
- 视图和部分基表构成了数据库的**外模式**，视图是一个虚表，是从基表或其他视图中导出的表；用户看来，视图和基表是一样的，都是关系。
- 数据库的存储文件及其索引文件构成了关系数据库的**内模式**。



□ SQL 编写规则

- 关键字、对象名、列名不区分大小写。
- 字符值和日期值区分大小写。

`select * from dba_users where username='scott';` 无结果
`select * from dba_users where username='SCOTT';` 有结果

- 可以将语句文本分布到多行，可以通过使用跳格和缩进提高可读性。

□ Oracle 使用 SQL

- SQL * Plus
- SQL Developer





□ SQL 分类——根据SQL指令完成的数据库操作不同

- **数据定义**：（Data Definition Language, **DDL**），用于完成对数据库对象（数据库、表、视图、索引等）的创建、修改和删除操作，分别对应 CREATE、ALTER 和 DROP 三条语句。**DDL语句执行后自动提交。**
- **数据查询**：（Data Query Language, **DQL**），用于对数据库中的各种数据对象进行查询。查询语句（SELECT）可以由许多子句组成，用以进行查询、统计、分组、排序等操作。
- **数据操纵**：（Data Manipulation Language, **DML**），用于改变数据库中的数据，数据操纵包括插入、删除和修改三种操作，分别对应 INSERT、DELETE 和 UPDATE 三条语句。**DCL语句执行后不会自动提交。**
- **数据控制**：（Data Control Language, **DCL**），用于对基表和视图的授权、完整性规则的描述以及事务开始和结束等控制语句，对应的语句有 GRANT、REVOKE、COMMIT和ROLLBACK等。**DCL执行后自动提交。**



□ SQL

□ SQL概述

□ 数据定义

□ 数据查询

□ 数据操纵

□ 数据控制



□ 数据定义 DDL

操作对象	操作方式		
	创建	删除	修改
表	CREATE TABLE	DROP TABLE	ALTER TABLE
视图	CREATE VIEW	DROP VIEW	
索引	CREATE INDEX	DROP INDEX	

CREATE TABLE (<表名> (<列名> <数据类型> [列级完整性约束条件]
[, <列名> <数据类型> [列级完整性约束条件]]...)
[, <表级完整性约束条件>]);

CREATE VIEW <视图名> [(<列名> [, <列名>]...)]
AS <子查询>
[WITH CHECK OPTION]

CREATE [UNIQUE] [CLUSTER] **INDEX** <索引名>
ON <表名> (<列名> [<次序>] [, <列名> [<次序>]]...);



□ 数据定义

- DROP

DROP TABLE <表名>;

DROP VIEW <视图名>;

DROP INDEX <索引名>;

- ALTER

ALTER TABLE <表名>

[ADD <新列名> <数据类型> [完整性约束]]

[DROP <完整性约束名>]

[MODIFY <列名> <数据类型>];



【例1】HR模式中，创建一个名为 IT_EMPLOYEES 的表，它由编号 EMPLOYEE_ID、名 FIRST_NAME、姓 LAST_NAME、邮箱 EMAIL、电话号码 PHONE_NUMBER、部门编号 JOB_ID、薪资 SALARY 和部门经理编号 MANAGER_ID 八个字段组成，其中 EMPLOYEE_ID 不能为空，值是唯一的。

```
CREATE TABLE IT_EMPLOYEES
```

```
(  
    EMPLOYEE_ID NUMBER(6) not null unique,  
    FIRST_NAME VARCHAR2(20),  
    LAST_NAME VARCHAR2(25),  
    EMAIL VARCHAR2(25),  
    PHONE_NUMBER VARCHAR(20),  
    JOB_ID VARCHAR2(10),  
    SALARY NUMBER(8,2),  
    MANAGER_ID NUMBER(6)  
);
```



【例2】 HR模式中，建立程序员视图 PROG_EMPLOYEES (JOB_ID 为 IT_PROG) 。

```
CREATE VIEW prog_employees  
as  
select employee_id, first_name, last_name, email, phone_number,  
salary, manager_id  
from it_employees  
where job_id = 'IT_PROG';
```

【例3】 HR模式中，向 IT_EMPLOYEES 表中增加“雇员生日”列，其数据类型为日期型。

```
ALTER TABLE IT_EMPLOYEES ADD BIRTHDATE DATE;
```

【例4】 HR模式中，删除 IT_EMPLOYEES 表 EMPLOYEE_ID 字段的 UNIQUE 约束。

```
ALTER TABLE IT_EMPLOYEES DROP UNIQUE(EMPLOYEE_ID);
```



□ SQL

- SQL概述

- 数据定义

- 数据查询

- 数据操纵

- 数据控制



□ 数据查询 DQL

用于从存储在数据表中的数据中查询满足特定条件的数据记录。在 SQL 语句中，数据查询语句 SELECT 是使用频率最高、用途最广的语句。

SELECT 目标列名序列

--需要哪些列

FROM 数据表

--来自于哪些表

WHERE 检索条件

--根据什么条件

GROUP BY 分组依据列

HAVING 组提取条件

ORDER BY 排序依据列



□ 数据查询

- **简单查询**：使用 FROM 子句指定表、指定列、使用算术表达式、DISTINCT 关键字去重。
- **WHERE**：筛选从 FROM 子句中返回的值，可使用条件表达式、连接运算符,不可使用聚合函数。
- **ORDER BY**：对结果集进行排序。
- **GROUP BY**：在查询结果集中对记录进行分组，可使用统计函数。
- **HAVING**：通常与 GROUP BY 子句一起使用，在完成对分组结果统计后，可以使用 HAVING 子句对分组的结果做进一步的筛选。
- **多表连接查询**：简单连接、JOIN连接（内、外、自）。
- **集合操作**：UNION、UNION ALL、INTERSECT、MINUS。
- **子查询**：IN、EXIST、比较运算符。



□ 数据查询语句示例

【例1】 HR 模式中，查询出所有属于 IT 部门（DEPARTMENT_ID=60），并且薪金值大于 2000 的雇员，并按薪金值升序排序。

【例2】 HR 模式中，列出职位平均薪金值大于 10000 的统计信息（平均薪金值、薪金总和、最大薪金、人数）。



□ 数据查询语句示例

【例1】HR 模式中，查询出所有属于 IT 部门（DEPARTMENT_ID=60），并且薪金值大于 2000 的雇员，并按薪金值升序排序。

```
select department_id, employee_id, first_name, last_name, salary from
employees
where department_id=60 and salary>2000
order by salary;
```

	DEPARTMENT_ID	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY
1	60	107	Diana	Lorentz	4200
2	60	105	David	Austin	4800
3	60	106	Valli	Pataballa	4800
4	60	104	Bruce	Ernst	6000
5	60	103	Alexander	Hunold	9000

【例2】HR 模式中，列出职位平均薪金值大于 10000 的统计信息（平均薪金值、薪金总和、最大薪金、人数）。

```
select job_id, avg(salary), sum(salary),
       max(salary), count(*)
from employees
group by job_id having avg(salary)>10000
```

	JOB_ID	AVG(SALARY)	SUM(SALARY)	MAX(SALARY)	COUNT(*)
1	AC_MGR	12008	12008	12008	1
2	PU_MAN	11000	11000	11000	1
3	AD_VP	17000	34000	17000	2
4	FI_MGR	12008	12008	12008	1
5	SA_MAN	12200	61000	14000	5
6	MK_MAN	13000	13000	13000	1
7	AD PRES	24000	24000	24000	1



□ 数据查询语句示例

【例3】HR 模式中，使用内连接查询 job_id 为 IT_PROG 的雇员的编号、姓氏、部门名称、职称。

- 国家表 **COUNTRIES**：国家编号、国家名称、地区编号。
- 部门表 **DEPARTMENTS**：部门编号、部门名称、经理编号、地点编号。
- 雇员表 **EMPLOYEES**：雇员编号、姓氏、名字、电子邮件、电话号码、雇佣日期、工作编号、工资、佣金比、经理编号、部门编号。
- 地区表 **REGIONS**：地区编号、地区名称。
- 工作表 **JOBS**：工作编号、职称、最低工资、最高工资。
- 工作历史表 **JOB_HISTORY**：雇员编号、开始时间、结束时间、工作编号、部门编号。
- 地点表 **LOCATIONS**：地点编号、街道地址、邮政编码、城市、州省、国家编号。
- 员工详细视图 **EMP_DETAILS_VIEW**：雇员编号、工作编号、经理编号、部门编号、地点编号、国家编号、姓氏、名字、工资、佣金比、部门名称、职称、城市、州省、国家名称、地区名称。



□ 数据查询语句示例

【例3】HR 模式中，使用内连接查询 job_id 为 IT_PROG 的雇员的编号、姓氏、部门名称、职称。

```
select employees.employee_id, employees.last_name,  
departments.department_name, jobs.job_title  
from employees inner join jobs  
on employees.job_id=jobs.job_id  
inner join departments  
on employees.department_id=departments.department_id  
where employees.job_id='IT_PROG';
```

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_NAME	JOB_TITLE
1	107	Lorentz	IT	Programmer
2	106	Pataballa	IT	Programmer
3	105	Austin	IT	Programmer
4	104	Ernst	IT	Programmer
5	103	Hunold	IT	Programmer



□ 数据查询语句示例

【例4】HR 模式中，解释下面语句的含义。

```
select employee_id,last_name
from employees
where last_name like 'C%' or last_name like 'S%'
minus      -- 差集
select employee_id,last_name
from employees
where last_name like 'S%' or last_name like 'T%';
```



□ 数据查询语句示例

【例4】HR 模式中，解释下面语句的含义。

```
select employee_id,last_name
from employees
  where last_name like 'C%'or last_name like 'S%'
minus
select employee_id,last_name
from employees
whefe last_name like 'S%' or last_name like 'T%';
```

第一个查询会返回所有 LAST_NAME 以 C 或 S 开头的雇员，而第二个查询会返回所有 LAST_NAME 以 S 和 T 开头的雇员。因此，两个查询结果集的 MINUS 操作将返回 LAST_NAME 以 C 开头的那些雇员。

	EMPLOYEE_ID	LAST_NAME
1	110	Chen
2	119	Colmenares
3	148	Cambrault
4	154	Cambrault
5	187	Cabrio
6	188	Chung



□ 数据查询语句示例

【例5】HR 模式中，使用子查询查看所有部门在某一地区（1700）的雇员信息。

```
select employee_id, last_name, department_id
from employees
where department_id in (
    select department_id
    from departments
    where location_id=1700);
```

- 部门表 **DEPARTMENTS**: 部门编号、部门名称、经理编号、地点编号。
- 雇员表 **EMPLOYEES**: 雇员编号、姓氏、名字、电子邮件、电话号码、雇佣日期、工作编号、工资、佣金比、经理编号、部门编号。

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	100	King	90
2	101	Kochhar	90
3	102	De Haan	90
4	108	Greenberg	100
5	109	Faviet	100
6	110	Chen	100
7	111	Sciarra	100
8	112	Urman	100
9	113	Popp	100
10	114	Raphaely	30
11	115	Khoo	30
12	116	Baida	30
13	117	Tobias	30
14	118	Himuro	30
15	119	Colmenares	30
16	200	Whalen	10
17	205	Higgins	110
18	206	Gietz	110



- SQL回顾
 - SQL概述
 - 数据定义
 - 数据查询
 - 数据操纵
 - 数据控制



□ 数据操纵 DML

- INSERT: 用于向数据表中插入记录。

- 一般 INSERT 语句

```
INSERT INTO [user.]table [@db_link] [ (column1 [ ,column2]... )]  
VALUES (express1 [ ,express2 ] ... )
```

- 批量 INSERT 语句

```
INSERT INTO [user.]table [@db_link] [ (column1 [ ,column2]... )]  
Subquery
```

- UPDATE: 用于修改数据表中一列或多列的值。

```
UPDATE table_name  
SET { column 1 = express1 [ column2 = express2 ]  
( column1 [ ,column2 ] ) = ( select query ) }  
[WHERE condition]
```



□ 数据操纵 DML

- DELETE：用于从数据表中删除记录。

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

区别：TRUNCATE：不带回滚（即不可撤销）的删除。

```
TRUNCATE FROM table_name  
[ WHERE condition ]
```

当你不再需要该表时，用 **drop**；当你仍要保留该表，但要删除所有记录时，用 **truncate**；当你要删除部分记录时（**always with a WHERE clause**），用 **delete**。



□ 数据操纵语句示例

【例1】HR模式中，我们建立了一个名为IT_EMPLOYEES的表，下面的示例将从EMPLOYEES 表提取 department_id 等于 IT 的雇员信息，并保存到IT_EMPLOYEES 中。

```
insert into IT_EMPLOYEES( employee_id, first_name, last_name,
email, phone_number, job_id, salary, manager_id)
select em.employee_id, em.first_name, em.last_name, em.email,
       em.phone_number, em.job_id, em.salary, em.manager_id
from employees em, departments dep                --隐式内连接
where em.department_id=dep.department_id
and dep.department_name='IT';
```

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	JOB_ID	SALARY	MANAGER_ID
1	103	Alexander	Hunold	AHUNOLD	590.423.4567	IT_PROG	9000	102
2	104	Bruce	Ernst	BERNST	590.423.4568	IT_PROG	6000	103
3	105	David	Austin	DAUSTIN	590.423.4569	IT_PROG	4800	103
4	106	Valli	Pataballa	VPATABAL	590.423.4560	IT_PROG	4800	103
5	107	Diana	Lorentz	DLORENTZ	590.423.5567	IT_PROG	4200	103



□ 数据操纵语句示例

【例2】HR模式中，使用 UPDATE 语句更新编号为 104 的雇员薪金，调整后的薪金为 IT 程序员的平均薪金。

查看104雇员的工资和平均工资

```
select salary from employees where employee_id=104; 6000
```

```
select avg(salary) from employees where job_id='IT_PROG'; 5760
```

```
update employees set salary=
(select avg(salary) from employees where job_id='IT_PROG')
where employee_id=104;
```

验证

```
select salary from employees where employee_id=104; 5760
```

【例3】HR模式中，从 IT_EMPLOYEES 表中删除一条记录 employee_id=107。

```
delete from it_employees where employee_id=107;
```

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	JOB_ID	SALARY	MANAGER_ID
1	103	Alexander	Hunold	AHUNOLD	590.423.4567	IT_PROG	9000	102
2	104	Bruce	Ernst	BERNST	590.423.4568	IT_PROG	6000	103
3	105	David	Austin	DAUSTIN	590.423.4569	IT_PROG	4800	103
4	106	Valli	Pataballa	VPATABAL	590.423.4560	IT_PROG	4800	103



□ SQL

□ SQL概述

□ 数据定义

□ 数据查询

□ 数据操纵

□ 数据控制



□ 数据控制 DCL

- GRANT/REVOKE: 用于向用户授予操作权限。

GRANT/REVOKE <权限> [, <权限>]...

[ON <对象类型> <对象名>]

TO/FROM <用户> [, <用户>]...

[WITH GRANT OPTION]

对象	对象类型	操作权限
属性列	TABLE	SELECT、INSERT、UPDATE、DELETE、ALL PRIVILEGES
视图	TABLE	SELECT、INSERT、UPDATE、DELETE、ALL PRIVILEGES
基表	TABLE	SELECT、INSERT、UPDATE、DELETE、ALTER、INDEX、ALL PRIVILEGES
数据库	DATABASE	CREATETAB



□ 数据控制语句示例

【例1】HR模式中，把查询 IT_EMPLOYEES 表的权限授给用户 SCOTT，再回收。

1 授权之前，用SCOTT执行 `select * from HR.IT_EMPLOYEES;`

```
ORA-00942: 表或视图不存在
00942. 00000 - "table or view does not exist"
```

2 换管理员（HR）身份执行 `grant SELECT on HR.IT_EMPLOYEES to SCOTT;`

3 用SCOTT再执行 `select * from HR.IT_EMPLOYEES;`

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	JOB_ID	SALARY	MANAGER_ID
1	103	Alexander	Hunold	AHUNOLD	590.423.4567	IT_PROG	9000	102
2	104	Bruce	Ernst	BERNST	590.423.4568	IT_PROG	6000	103
3	105	David	Austin	DAUSTIN	590.423.4569	IT_PROG	4800	103
4	106	Valli	Pataballa	VPATABAL	590.423.4560	IT_PROG	4800	103

4 换管理员（HR）身份执行 `revoke SELECT on HR.IT_EMPLOYEES from SCOTT;`

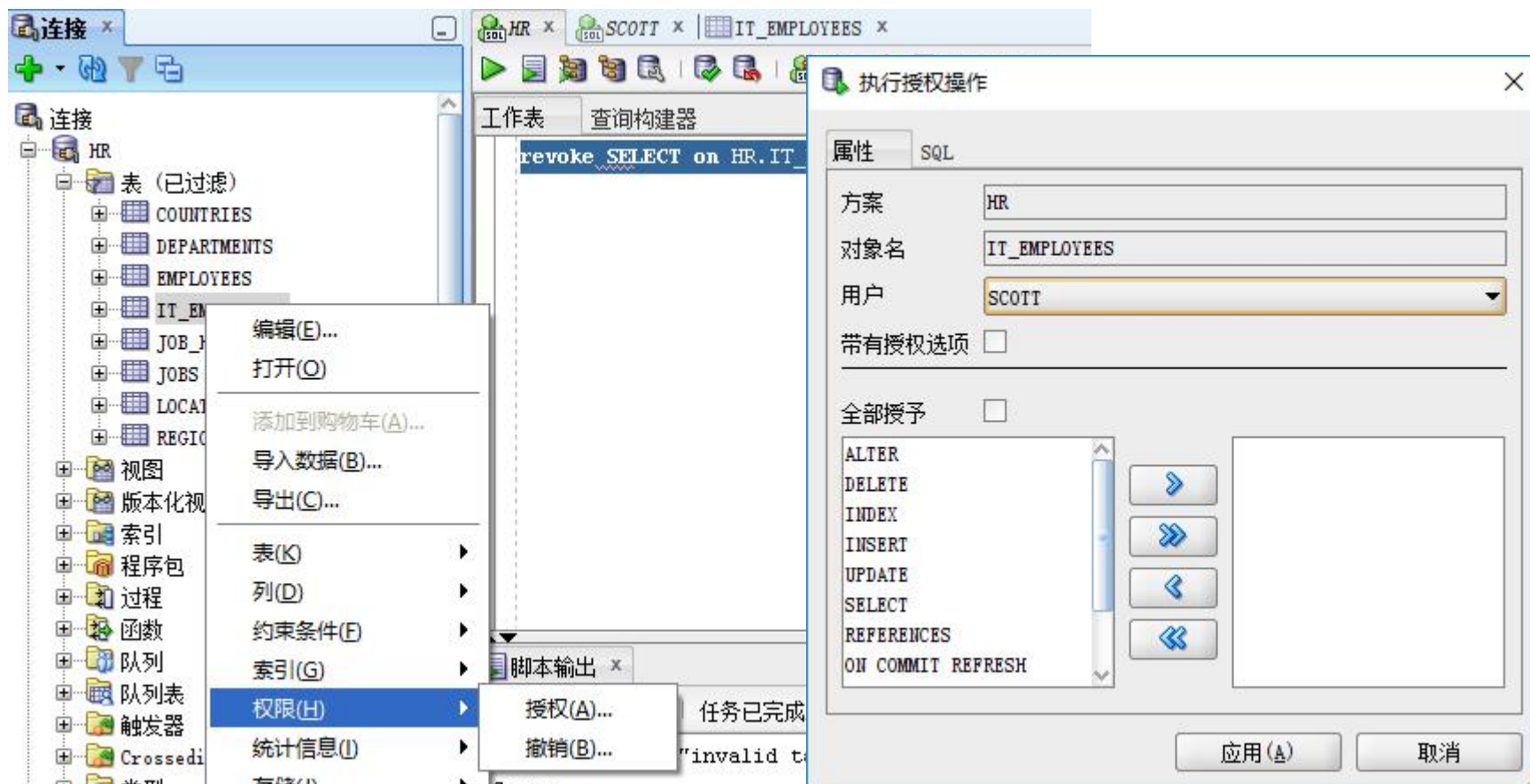
5 用SCOTT再执行 `select * from HR.IT_EMPLOYEES;`

```
ORA-00942: 表或视图不存在
00942. 00000 - "table or view does not exist"
```



□ 数据控制

【例2】可使用SQL Developer 界面化操作进行授权和回收。





□ PL/SQL编程

□ PL/SQL概述

□ 字符集和数据类型

□ 程序结构和语句

□ 过程和函数

□ 游标

□ 程序包和触发器



□ PL/SQL概述

PL/SQL (Procedural Language / SQL) 是 Oracle 在标准 SQL 上扩展后的程序设计语言，是 Oracle 数据库特有的、支持应用开发的语言。

PL/SQL程序块的结构：

[DECLARE]——声明部分，**可选**

声明一些变量、常量、用户定义的数据类型以及游标等

BEGIN——执行部分，**必须**

主程序体，可以加入各种合法语句

[EXCEPTION]——异常处理部分，**可选**

异常处理程序，当程序中出现错误时执行这一部分

END ;——主程序体结束



□ PL/SQL概述

一个PL/SQL程序包含一个或多个块，这些块可以独立存在，也可以嵌套在另一个块中，PL/SQL程序有三种类型的块组成，分别为匿名块、过程和函数。

匿名块↵	过程↵	函数↵
[DECLARE]↵	PROCEDURE name↵	FUNCTION name↵
↵	IS↵	RETURN datatype↵
↵	↵	IS↵
BEGIN↵	BEGIN↵	BEGIN↵
-statements↵	-statements↵	-statements↵
↵	↵	↵
EXCEPTION↵	EXCEPTION↵	EXCEPTION↵
↵	↵	↵
END;↵	END;↵	END;↵



□ PL/SQL概述

- 匿名块是未命名的块，在运行时被执行，匿名块不存储在数据库中，如果要再次执行相同的块，则必须重写该块，并在执行时重新进行编译。由于块是匿名的，每次执行后该块就不再存在，开发者不能调用以前写过的匿名块。
- 过程和函数都称作子程序，它们是匿名块的有益补充，是存储在数据库中被命名的PL/SQL块。由于子程序已经命名，且存储在数据库中，开发者可以随时调用它们，可以将子程序声明为过程或函数，通常使用过程执行一些操作，而使用函数进行计算并返回一个值。
- 函数和过程类似，不同的是函数必须有返回值，而过程没有。



□ PL/SQL编程

□ PL/SQL概述

□ 数据类型

□ 程序结构和语句

□ 过程和函数

□ 游标

□ 程序包和触发器



□ 变量和常量

- **变量**是指其值在程序运行过程中**可以改变**的数据存储结构，定义变量必须的元素就是变量名和数据类型，另外还有可选择的初始值。

<变量名> <数据类型> [:= <初始值>];

- PL/SQL定义了一个未初始化变量应该存放的内容，被赋值为NULL。
- **常量**是指其值在程序运行过程中**不可改变**的数据存储结构，定义常量必须的元素包括常量名、数据类型、**常量值**和 constant 关键字。

<常量名> constant <数据类型> := <常量值>;

```
Birthday DATE ;  
emp_count SMALLINT :=0;  
credit_limit CONSTANT REAL:=5000.00;  
pi REAL:=3.14159;  
radius REAL:=1;  
area REAL:= pi*radius**2;
```

--变量和常量示例



□ 字符集

● 合法字符

- 所有的大写和小写英文字母，数字0~9；
- 符号 ()、+、-、*、/、<、>、=、!、~、;、:、..、'、@、%、,、"、#、^、&、_、{、}、?、[、]。

● 运算符

- 算术运算符：+（加）、-（减）、*（乘）、/（除）、**（指数）和 ||（连接字符）；
- 关系运算符：比较运算符、BETWEEN ... AND ...、IN、LIKE、IS NULL / IS NOT NULL；
- 逻辑运算符：AND、OR、NOT。

● 其它符号：列表分隔、项分离、字符串界定符、注释符等。



□ 标识符

- 标识符用于定义PL/SQL变量、常量、异常、游标名称、游标变量、参数、子程序名称和其他的程序单元名称等。
- 在PL/SQL程序中，标识符是**以字母开头**的，后边可以跟字母、数字、美元符号（\$）、井号（#）或下划线（_），其最大长度为30个字符。
- 标识符中**不能包含减号“-”和空格**，不能使SQL保留字
 - 例如，X，v_empno，v_\$等都是有效的标识符，而X-y，_temp则是非法的标识符。
- **注意：**一般不要把变量名声明与表中字段名完全一样,如果这样可能得到不正确的结果。



□ 类型概览

● 系统预定义类型

- 标量型：标量数据类型的变量**只有一个值**，且内部没有分量。包括数值型，字符型，日期/时间型和布尔型。
- 复合型：含有能够被单独操作的内部组件，包括 **record**，**table** 和 **cursor** 型。
- LOB 型：专门用于存储大对象的数据，包括大文本、图像图像、视频剪辑等。分为内部 LOB 和外部 LOB。
- 引用类型：引用数据类型是 PL/SQL 程序语言特有的数据类型，是用来引用数据库当中的某一行或者某个字段作为数据类型的声明。

● 用户自定义子类型：

```
SUBTYPE subtype_name IS base_type [ (constraint) ] [ NOT NULL ];
```

SUBTYPE *subtype_name* **IS** *base_type*[(constraint)] [NOT NULL];

□ 用户定义子类型示例

SET serveroutput ON --显示服务器输出信息

DECLARE

SUBTYPE name IS CHAR(20);

--定义了一个name类型

SUBTYPE MESSAGE IS VARCHAR2(100);

--定义了一个MESSAGE类型

salutation name;

--定义了一个name类型的变量

greetings MESSAGE;

--定义了一个MESSAGE类型的变量

BEGIN

salutation := 'Reader';

greetings := 'Welcome to the World

dbms_output.put_line('Hello ' || sal

END;

注意

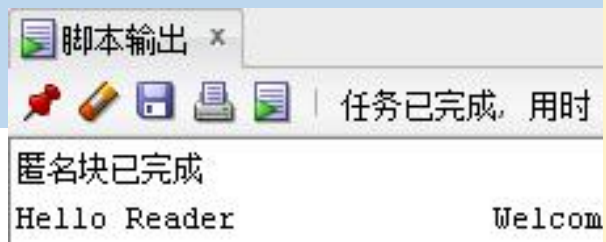
若要在SQL*Plus环境中看到

DBMS_OUTPUT.PUT_LINE方法

的输出结果，必须将环境变量

SERVEROUTPUT设置为ON。

SET SERVEROUTPUT ON





□ 标量型——数值类型和子类型

序号	类型	描述
1	PLS_INTEGER	带符号整数：-2,147,483,648至2,147,483,647，以32位表示
2	BINARY_INTEGER	带符号整数：-2,147,483,648至2,147,483,647，以32位表示
3	BINARY_FLOAT	单精度IEEE 754格式浮点数
4	BINARY_DOUBLE	双精度IEEE 754格式浮点数
5	NUMBER(prec, scale)	在1E-130到(但不包括)1.0E126范围内的绝对值的定点或浮点数。 NUMBER变量也可以表示0
6	DEC(prec, scale)	ANSI特定定点类型，最大精度为38位十进制数字
7	DECIMAL(prec, scale)	IBM具体定点类型，最大精度为38位十进制数字
8	NUMERIC(prec, scale)	浮点型，最大精度为38位十进制数
9	DOUBLE PRECISION	ANSI特定浮点类型，最大精度为126位二进制数字
10	FLOAT	ANSI和IBM特定浮点类型，最大精度为126位二进制数字
11	INT	ANSI特定整数类型，最大精度为38位十进制数
12	INTEGER	ANSI和IBM特定整数类型，最大精度为38位十进制数
13	SMALLINT	ANSI和IBM特定整数类型，最大精度为38位十进制数
14	REAL	浮点型，最大精度为63位二进制数字(约十八位数)



□ 标量型——字符类型和子类型

序号	类型	描述
1	CHAR	固定长度字符串，最大大小为32,767字节
2	VARCHAR2	最大大小为32,767字节的可变长度字符串
3	RAW	最大大小为32,767字节的可变长度二进制或字节字符串，不由PL/SQL解释
4	NCHAR	固定长度的国家字符串，最大大小为32,767字节
5	NVARCHAR2	可变长度的国家字符串，最大大小为32,767字节
6	LONG	最大长度为32,760字节的可变长度字符串
7	LONG RAW	最大大小为32,760字节的可变长度二进制或字节字符串，不由PL/SQL解释
8	ROWID	物理行标识符，普通表中的行的地址
9	UROWID	通用行标识符(物理，逻辑或外部行标识符)



□ 标量型——日期/时间类型

序号	类型	描述
1	DATE	7字节的定宽日期/时间数据类型。包含7个属性，包括：世纪、世纪中的年、月份、月份中的天、小时、分钟和秒。 示例：2005-12-05 12:30:43 存储的是 120, 105, 12, 5, 12, 31, 44。 解释：世纪和年份分别加100，分钟和秒钟分别加1。
2	TIMESTAMP	与DATE的区别是不仅可以保存日期和时间，还能保存小数秒，小数位数可以指定为0-9，默认为6位，所以最高精度可以到ns(纳秒)，数据库内部用7或者11个字节存储，如果精度为0，则用7字节存储，与DATE类型功能相同，如果精度大于0则用11字节存储。
3	INTERVAL	INTERVAL YEAR TO MONTH: 单位为年和月的时间间隔； INTERVAL DAY TO SECOND: 单位为天和秒的时间间隔；

INTERVAL YEAR [(year_precision)] TO MONTH

INTERVAL DAY [(day_precision)] TO SECOND [(fractional_seconds_precision)]

year_precision、day_precision、fractional_seconds_precision: 精度位数，默认年和日是两位数，小数秒是六位数。

示例：INTERVAL '123 2:25:45.1247' DAY(3) TO SECOND(4)

说明：时间间隔为123天2小时25分45.1247秒；天的精度是3位数字，小数秒精度是4位数字。



□ 复合型

序号	类型	描述
1	RECORD	<p>记录类型，使用该类型的变量可以存储由多个列值组成的一行数据。在声明记录类型变量之前，首先需要定义记录类型，然后才可以声明记录类型的变量。</p> <pre>TYPE record类型名称 is record(属性 数据类型, 属性 数据类型 ...);</pre>
2	TABLE	<p>使用记录类型变量只能保存一行数据，这限制了SELECT语句的返回行数，如果SELECT语句返回多行就会错。Oracle提供了另外一种自定义类型，也就是表类型，它是对记录类型的扩展，允许处理多行数据，类似于表。</p> <pre>TYPE table类型名称 IS TABLE OF data_type [NOT NULL] INDEX BY BINARY_INTEGER;</pre>
3	CURSOR	游标 后续详细介绍。



- **RECORD类型**：“记录类型”，使用该类型的变量可以存储由多个列值组成的一行数据。在声明记录类型变量之前，首先需要定义记录类型，然后才可以声明记录类型的变量。

```
TYPE record_name IS RECORD(  
    field1 data_type1 [:=value1 ],  
    field 2 data_type2 [:= value ],  
    .....  
    field n data_typen [:= valuen ] );
```

```
type emp_type is record  
(    var_ename varchar2(20),  
    var_job varchar2(20),  
    var_sal number );  
empinfo emp_type;
```

注意：

- 相同记录类型的变量可以相互赋值；
- 不同记录类型的变量，即使成员完全相同也不能相互赋值；



RECORD类型使用示例

【例】SCOTT 模式中，声明一个记录类型，然后使用该类型的变量存储 emp 表中的一条记录信息并输出。

```
--SET serveroutput ON
```

```
DECLARE
```

```
type emp_type IS record
```

--声明record类型emp_type

```
(  
    var_ename VARCHAR2(20),  
    var_job    VARCHAR2(20),  
    var_sal    NUMBER );
```

--定义字段/成员变量

```
empinfo emp_type;
```

```
BEGIN
```

```
SELECT ename, job, sal INTO empinfo FROM emp WHERE empno=7369;
```

/*输出雇员信息*/

```
dbms_output.put_line('雇员'||empinfo.var_ename||'的职务是'  
'||empinfo.var_job||'、工资是'||empinfo.var_sal);
```

```
END;
```

PL/SQL是面向过程语言和SQL语言的结合，可使用SELECT-INTO语句将SQL语言查询结果存入变量。





□ LOB 型 大对象数据类型

序号	类型	描述
1	BFILE	外部LOB类型，二进制文件LOB。与其说是一个数据库存储实体，不如说是一个指针。带BFILE列的数据库存储的只是某个文件的一个指针。这个文件在数据库之外维护，不是数据库的一部分。BFILE提供了对文件内容的只读访问。
2	BLOB	二进制LOB。用于存储大量的二进制信息，如文档、图像和任何其他数据。它不会执行字符集转换。应用向BLOB写入什么位和字节，其就会返回什么位和字节。
3	CLOB	字符LOB。这种类型用于存储大量的文本信息，如xml或者只是纯文本。这个数据类型需要进行字符集转换，也就是说，在获取时，这个字段中的字符会从数据库的字符集转换为客户的字符集，而在修改时会总客户的字符集转换为数据库的字符集。
4	NCLOB	存储在这一列中的数据所采用的字符集是数据库的国家字符集，而不是数据库的默认字符集。



□ 引用型

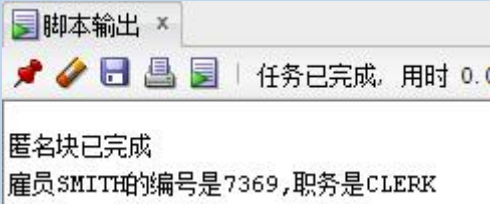
序号	类型	描述
1	%TYPE	引用数据表的某列的类型作为某变量的数据类型，也可以直接引用PL/SQL程序中某个变量作为新变量的数据类型。 示例： <code>pename emp.ename%type;</code> 解释：创建一个变量 <code>pename</code> ，其类型与 <code>emp</code> 表 <code>ename</code> 列类型相同；
2	%ROWTYPE	引用数据库表中的一行作为数据类型，即 RECORD 类型表示一条数据记录。 示例： <code>rowVar_emp emp%rowtype;</code> 解释：创建一个变量 <code>rowVar_emp</code> （Record 类型），其能够存储 <code>emp</code> 表的一行数据；

DECLARE

```
rowVar_emp emp%rowtype;
```

BEGIN

```
SELECT * INTO rowVar_emp FROM emp WHERE empno=7369;  
dbms_output.put_line('雇员'||rowVar_emp.ename||'的编号是'  
'||rowVar_emp.empno||',职务是'||rowVar_emp.job);  
END;
```





- **%TYPE类型**: 使用 %TYPE 关键字可以声明一个与指定列名称相同的数据类型，它通常紧跟在指定列名的后面。

变量名 表名.字段名% **%type**
v_job emp.job **%type**

使用%type的优点:

- 所引用的数据库列的数据类型可以不必知道;
- 所引用的数据库列的数据类型可以实时改变。



- **%ROWTYPE 类型**：可以根据数据表中行的结构定义一种特殊的数据类型，用来存储从数据表中检索到的一行数据。

记录变量名 表名 **%rowtype**

rowVar_emp emp **%rowtype**

使用%ROWTYPE特性的优点：

- 所引用的数据库中列的个数和数据类型可以不必知道；
- 所引用的数据库中列的个数和数据类型可以实时改变。

%ROWTYPE类型



DECLARE

v_emp emp%rowtype;

BEGIN

select * into v_emp from emp where empno=&eno;

dbms_output.put_line('员工名字:'||v_emp.ename);

dbms_output.put_line('员工工资:'||v_emp.sal);

dbms_output.put_line('员工职位'||v_emp.job);

END;

PL/SQL是面向过程语言和SQL语言的结合，可使用SELECT-INTO语句将SQL语言查询结果存入变量。

匿名块已完成
员工名字: SMITH
员工工资: 800
员工职位 CLERK



1. 复习sql
2. SCOTT模式中，查找职员入工时间（hiredate）排名倒数第三的员工所有信息。（提示：可使用limit）
3. 学生选课三张表，S学生表、C课程表、 S-C选课表，
 - a.查询出被2至4名学生选修的所有课程信息，及选课人数
 - b.查询选修课程超过5门的学生姓名和所选修课程的数目