Oracle 第三天

一、使用 DDL 语句管理表

● 创建表空间

表空间？ ORACLE 数据库的逻辑单元。 数据库---表空间 一个表空间可以与多个数据文件(物理结构)关联一个数据库下可以建立多个表空间，一个表空间可以建立多个用户、一个用户下可以建立多个表。

```
create tablespace itcast001
datafile 'c:\itcast001.dbf'
size 100m
autoextend on
next 10m
```

itcast 为表空间名称
datafile 指定表空间对应的数据文件
size 后定义的是表空间的初始大小
autoextend on 自动增长 ，当表空间存储都占满时，自动增长
next 后指定的是一次自动增长的大小。

● 用户

1、创建用户

```
create user itcastuser
identified by itcast
default tablespace itcast001
```

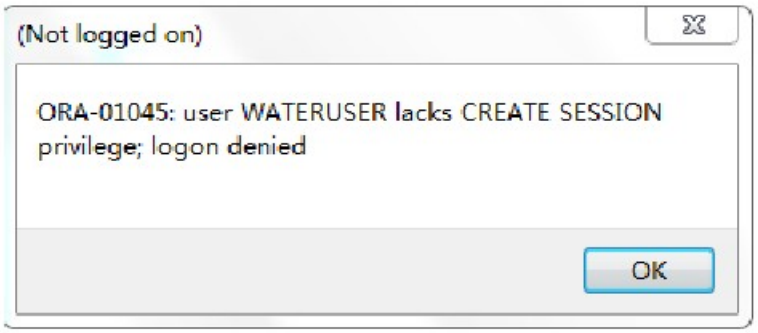
identified by 后边是用户的密码

default tablespace 后边是表空间名称

oracle 数据库与其它数据库产品的区别在于，表和其它的数据库对象都是存储在用户下的。

2、用户赋权限

新创建的用户没有任何权限，登陆后会提示



Oracle 中已存在三个重要的角色：connect 角色，resource 角色，dba 角色。

CONNECT 角色： --是授予最终用户的典型权利，最基本的

ALTER SESSION --修改会话
CREATE CLUSTER --建立聚簇
CREATE DATABASE LINK --建立数据库链接
CREATE SEQUENCE --建立序列
CREATE SESSION --建立会话
CREATE SYNONYM --建立同义词
CREATE VIEW --建立视图

RESOURCE 角色： --是授予开发人员的

CREATE CLUSTER --建立聚簇
CREATE PROCEDURE --建立过程
CREATE SEQUENCE --建立序列
CREATE TABLE --建表
CREATE TRIGGER --建立触发器
CREATE TYPE --建立类型

DBA 角色：拥有全部特权，是系统最高权限，只有 DBA 才可以创建数据库结构，并且系统权限也需要 DBA 授出，且 DBA 用户可以操作全体用户的任意基表，包括删除

```
grant dba to itcastuser
```

进入 system 用户下给用户赋予 dba 权限，否则无法正常登陆

- 创建表

语法：

```
CREATE TABLE [schema.]table
            (column datatype [DEFAULT expr][, ...]);
```

数据的类型：

数据类型	描述
VARCHAR2(size)	可变长字符数据
CHAR(size)	定长字符数据
NUMBER(p,s)	可变长数值数据
DATE	日期型数据
LONG	可变长字符数据，最大可达到2G
CLOB	字符数据，最大可达到4G
RAW and LONG RAW	原始的二进制数据
BLOB	二进制数据，最大可达到4G
BFILE	存储外部文件的二进制数据，最大可达到4G
ROWID	行地址

使用子查询创建表的语法：

```
CREATE TABLE table
       [(column, column...)]
AS subquery;
```

创建表范例：创建 person 表

```
create table person(
      pid     number(10),
      name    varchar2(10),
      gender   number(1)  default 1,
      birthday date
);
insert into person(pid, name, gender, birthday)
values(1, '张三', 1, to_date('1999-12-22', 'yyyy-MM-dd'));
```

● 修改表

在 sql 中使用 alter 可以修改表

- 添加语法：ALTER TABLE 表名称 ADD(列名 1 类型 [DEFAULT 默认值]，列名 1 类型 [DEFAULT 默认值]...)
- 修改语法：ALTER TABLE 表名称 MODIFY(列名 1 类型 [DEFAULT 默认值]，列名 1 类型 [DEFAULT 默认值]...)
- 修改列名：ALTER TABLE 表名称 RENAME COLUMN 列名 1 TO 列名 2

范例：在 person 表中增加列 address

```
alter table person add(address varchar2(10));
```

范例：把 person 表的 address 列的长度修改成 20 长度

```
alter table person modify(address varchar2(20));
```

- 删除表

语法：DROP TABLE 表名

- 约束

在数据库开发中，约束是必不可少，使用约束可以更好的保证数据的完整性。在 Oracle 数据库中，约束的类型包括：

- ✓ 主键约束（Primary Key）
- ✓ 非空约束（Not Null）
- ✓ 唯一约束（Unique）
- ✓ 外键约束（Foreign Key）
- ✓ 检查性约束（Check）

1.主键约束

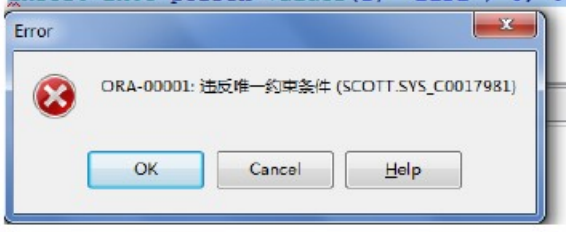
主键约束都是在 id 上使用，而且本身已经默认了内容不能为空，可以在建表的时候指定。

创建一张表，把 pid 作为主键

```
create table person(
      pid      number(10) primary key,
      name     varchar2(10),
      gender   number(1)  default 1,
      birthday date
);
```

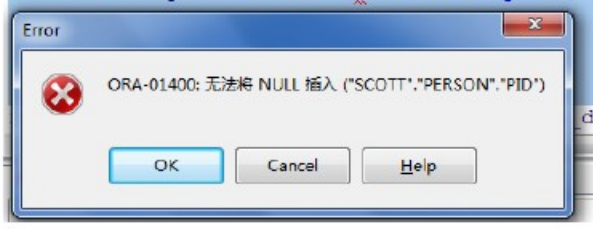
主键不可重复， SCOTT.SYS_C0017981是系统自动分配的约束的名字

```
insert into person values(1, 'zhangsan', 1, to_date('2012-12-12','yyyy-MM-dd'));
insert into person values(1, 'lisi', 0, to_date('2012-12-12','yyyy-MM-dd'));
```



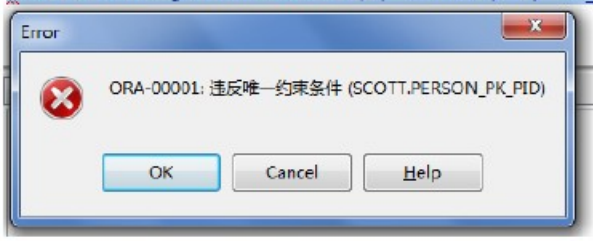
主键不可为空

```
insert into person values(null, 'zhangsan', 1, to_date('2012-12-12','yyyy-MM-dd'));
```



我们可以自己来指定主键约束的名字

```
create table person(
      pid      number(10),
      name     varchar2(10),
      gender   number(1)  default 1,
      birthday date,
      constraint person_pk_pid primary key(pid)
);
insert into person values(1, 'zhangsan', 1, to_date('2012-12-12','yyyy-MM-dd'));
insert into person values(1, 'lisi', 0, to_date('2012-12-12','yyyy-MM-dd'));
```



2.非空约束

使用非空约束，可以使指定的字段不可以为空。

范例：建立一张 **pid** 和 **name** 不可以为空的表

```
create table person(
      pid      number(10) not null,
      name     varchar2(10) not null,
      gender   number(1)  ,
      birthday date,
);
insert into person values(1, null, 1, to_date('2012-12-12','yyyy-MM-dd'));
```

3.唯一约束（unique）

表中的一个字段的内容是唯一的

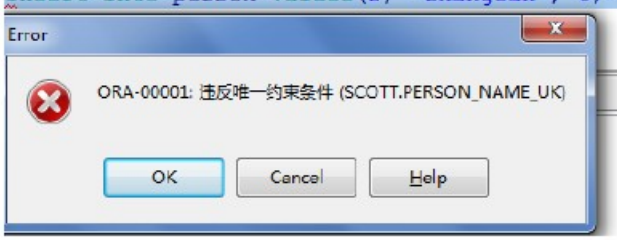
范例：建表一个 **name** 是唯一的表

```
create table person(
      pid      number(10) ,
      name     varchar2(10) unique,
      gender   number(1)  ,
      birthday date
```

```
);
insert into person values(1, 'zhangsan', 1, to_date('2012-12-12','yyyy-MM-dd'));
insert into person values(2, 'zhangsan', 0, to_date('2012-12-12','yyyy-MM-dd'));
```

唯一约束的名字也可以自定义

```
create table person(
      pid      number(10) ,
      name     varchar2(10),
      gender   number(1)  ,
      birthday date,
      constraint person_name_uk unique(name)
);
insert into person values(1, 'zhangsan', 1, to_date('2012-12-12','yyyy-MM-dd'));
insert into person values(2, 'zhangsan', 0, to_date('2012-12-12','yyyy-MM-dd'));
```



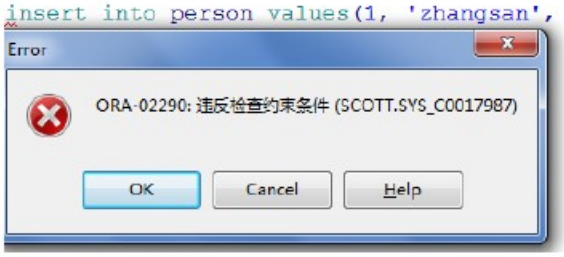
4.检查约束

使用检查约束可以来约束字段值的合法范围。

范例：创建一张表性别只能是 1 或 2

```
create table person(
      pid      number(10) ,
      name     varchar2(10),
      gender   number(1)  check(gender in (1, 2)),
      birthday date
);

insert into person values(1, 'zhangsan', 3, to_date('2012-12-12','yyyy-MM-dd'));
```



检查约束也可以自定义

```
create table person(
      pid      number(10) ,
```

```
        name     varchar2(10),
        gender   number(1),
        birthday date,
        constraint person_gender_ck check(gender in (1,2))
);
insert into person values(1, 'zhangsan', 3, to_date('2012-12-12','yyyy-MM-dd'));
```

5.外键约束

之前所讲的都是单表的约束，外键是两张表的约束，可以保证关联数据的完整性。
范例：创建两张表，一张订单表，一张是订单明细表，订单和明细是一对多的关系

```
create table orders(
        order_id      number(10) ,
        total_price   number(10,2),
        order_time date,
      constraint orders_order_id_pk primary key(order_id)
);

create table order_detail(
        detail_id      number(10) ,
        order_id   number(10),
        item_name  varchar2(10),
        quantity   number(10),
      constraint      order_detail_detail_id_pk      primary
key(detail_id)
);

insert        into        orders        values(1,        200,
to_date('2015-12-12','yyyy-MM-dd'));
insert into order_detail values(1, 2, 'java',1);
```

我们在两张表中插入如上两条数据，我们发现在 **order_detail** 表中插入的 **order_id** 在 **order** 表中并不存在，这样在数据库中就产生了脏数据。此时需要外键来约束它。

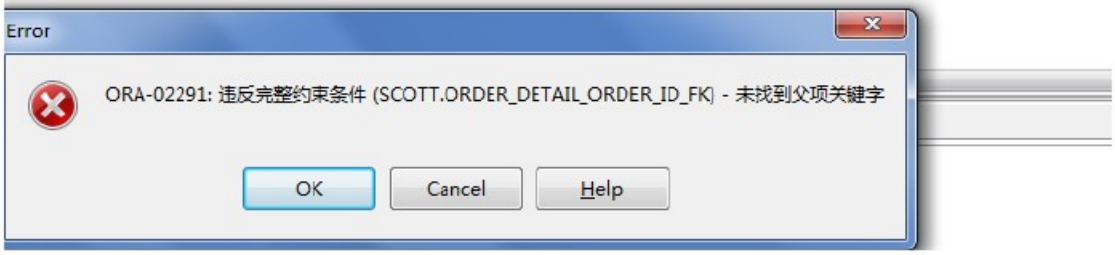
我们再次建表

```
create table orders(
        order_id      number(10) ,
        total_price   number(10,2),
        order_time date,
```

```
    constraint orders_order_id_pk primary key(order_id)
);

create table order_detail(
     detail_id     number(10) ,
     order_id   number(10),
     item_name  varchar2(10),
     quantity   number(10),
    constraint     order_detail_detail_id_pk     primary
key(detail_id),
    constraint     order_detail_order_id_fk     foreign
key(order_id) references orders(order_id)
);
```

```
insert into orders values(1, 200, to_date('2015-12-12','yyyy-MM-dd'));
insert into order_detail values(1, 2, 'java',1);
```



外键关联一定注意：
外键一定是主表的主键
删表时一定先删子表再删主表，如果直接删主表会出现由于约束存在无法删除的问题

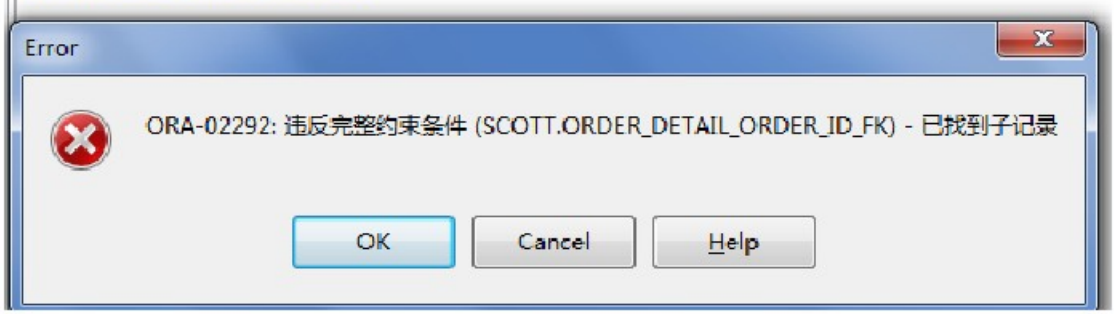
```
SQL> drop table orders;
drop table orders
ORA-02449: 表中的唯一/主键被外键引用
```

但是可以强制删除 drop table orders cascade constraint;(不建议)
删除主表的数据可以先删除子表的关联数据，再删主表，也可以使用级联删除。
级联删除在外键约束上要加上 **on delete cascade** 如

```
constraint order_detail_order_id_fk foreign key(order_id)
     references orders(order_id) on delete cascade
```

这样删除主表数据的时候会把字表的关联数据一同删除

```
delete from orders
```



二、使用 DML 语句处理数据

● 插入数据

语法：INSERT INTO 表名[(列名 1，列名 2，...)]VALUES(值 1，值 2，...)

标准写法

```
insert into person(pid,name,gender,birthday,address)
values(1,'张三',1,'9-5月-1981','北京北七家');
```

简单写法（不建议）

INSERT INTO 表名 VALUES(值 1，值 2，...)

```
insert into person
values(1,'张三',1,'9-5 月-1981','北京北七家');
```

注意：使用简单的写法必须按照表中的字段的顺序来插入值，而且如果有为空的字段使用 null

```
insert into person values(2,'李四',1,null,'北京育新');
```

● 更新数据

全部修改：UPDATE 表名 SET 列名 1=值 1，列名 2=值 2，....

局部修改：UPDATE 表名 SET 列名 1=值 1，列名 2=值 2，....WHERE 修改条件；

在 update 中使用子查询：

例如：给 NEW YORK 地区的所有员工涨 100 员工资

```
update emp set sal=sal+100 where deptno
   in (select deptno from dept where loc='NEW YORK')
```

● 删除数据

语法 :DELETE FROM 表名 WHERE 删除条件;

在删除语句中如果不指定删除条件的话就会删除所有的数据

Truncate table 实现数据删除

比较 truncat 与 delete 实现数据删除？

1. delete 删除的数据可以 rollback,也可以闪回
2. delete 删除可能产生碎片，并且不释放空间
3. truncate 是先摧毁表结构，再重构表结构

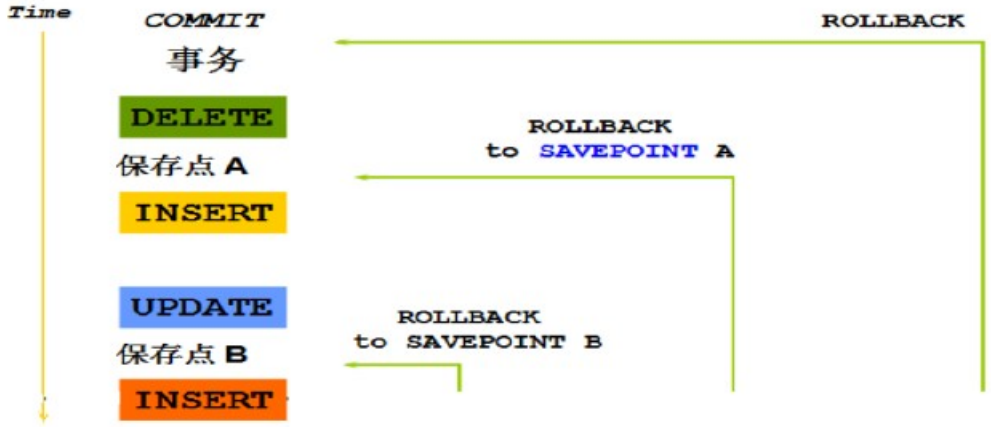
注意：插入、更新和删除会引起数据的变化。我们就必须考虑数据的完整性。

● Oracle 中的事务

这是因为 oracle 的事务对数据库的变更的处理，我们必须做提交事务才能让数据真正的插入到数据库中，在同样在执行完数据库变更的操作后还可以把事务进行回滚，这样就不会插入到数据库。如果事务提交后则不可以再回滚。

提交：commit
回滚：rollback

Oracle 中事务的保存点：



事务的隔离级别和隔离性：

隔离级别	描述
READ UNCOMMITTED (读未提交数据)	允许事务读取未被其他事物提交的变更. 脏读, 不可重复读和幻读的问题都会出现
READ COMMITED (读已提交数据)	只允许事务读取已经被其它事务提交的变更. 可以避免脏读, 但不可重复读和幻读问题仍然可能出现
REPEATABLE READ (可重复读)	确保事务可以多次从一个字段中读取相同的值. 在这个事务持续期间, 禁止其他事物对这个字段进行更新. 可以避免脏读和不可重复读, 但幻读的问题仍然存在.
SERIALIZABLE(串行化)	确保事务可以从一个表中读取相同的行. 在这个事务持续期间, 禁止其他事务对该表执行插入, 更新和删除操作. 所有并发问题都可以避免, 但性能十分低下.

Oracle 支持的 3 种事务隔离级别：READ COMMITED, SERIALIZABLE，READ ONLY.
Oracle 默认的事务隔离级别为: READ COMMITED

三、管理其他数据库对象

● 视图

➢ 什么是视图：

视图就是封装了一条复杂查询的语句。

视图是一个虚表。

最大的优点就是简化复杂的查询。

➢ 创建视图的语法

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

➢ 创建视图示例

范例：建立一个视图，此视图包括了 20 部门的全部员工信息

```
create view empvd20 as select * from emp t where t.deptno = 20
```

视图创建完毕就可以使用视图来查询，查询出来的都是 20 部门的员工

```
select * from EMPVD20 t
```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7369	SMITH	CLERK	7902	1980/12/17	800.00		20
2	7566	JONES	MANAGER	7839	1981/4/2	2975.00		20
3	7788	SCOTT	ANALYST	7566	1987/4/19	3000.00		20
4	7876	ADAMS	CLERK	7788	1987/5/23	1100.00		20
5	7902	FORD	ANALYST	7566	1981/12/3	3000.00		20

语法 2：CREATE OR REPLACE VIEW 视图名称 AS 子查询

如果视图已经存在我们可以使用语法 2 来创建视图，这样已有的视图会被覆盖。

```
create or replace view empvd20 as select * from emp t where t.deptno = 20
```

➢ 不建议通过视图对表中的数据进行修改，因为会受到很多的限制。

● 序列

在很多数据库中都存在一个自动增长的列，如果现在要想在 oracle 中完成自动增长的功能，则只能依靠序列完成，所有的自动增长操作，需要用户手工完成处理。并且 Oracle 将序列值装入内存可以提高访问效率。

语法：

```
CREATE SEQUENCE sequence
       [INCREMENT BY n]
       [START WITH n]
       [{MAXVALUE n | NOMAXVALUE}]
       [{MINVALUE n | NOMINVALUE}]
       [{CYCLE | NOCYCLE}]
       [{CACHE n | NOCACHE}];
```

范例：

```
CREATE SEQUENCE dept_deptid_seq
                INCREMENT BY 10
                START WITH 120
                MAXVALUE 9999
                NOCACHE
                NOCYCLE;
Sequence created.
```

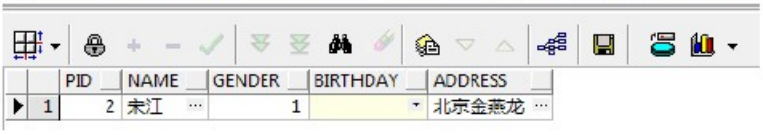
序列创建完成之后，所有的自动增长应该由用户自己处理，所以在序列中提供了以下的两种操作：

nextval ：取得序列的下一个内容

currval ：取得序列的当前内容

在插入数据时需要自增的主键中可以这样使用

```
insert into person
values(seqpersonid.nextval,'宋江',1,null,'北京金燕龙');
```



	PID	NAME	GENDER	BIRTHDAY	ADDRESS
1	2	宋江	1		北京金燕龙

序列可能产生裂缝的原因：

- 回滚
- 系统异常
- 多个表共用一个序列

- 索引

索引是用于加速数据存取的数据对象。合理的使用索引可以大大降低 i/o 次数,从而提高数据访问性能。

1. 单列索引

单列索引是基于单个列所建立的索引，比如:

```
CREATE index 索引名  on 表名(列名)
```

2. 复合索引

复合索引是基于两个列或多个列的索引。在同一张表上可以有多个索引，但是要求列的组合必须不同,比如:

```
Create index emp_idx1 on emp(ename,job);
Create index emp_idx1 on emp(job,ename);
```

范例：给 person 表的 name 建立索引

```
create index pname_index on person(name);
```

范例：给 person 表创建一个 name 和 gender 的索引

```
create index pname_gender_index on person(name, gender);
```

3. 索引测试

创建一个新表，向表中插入 **100W** 或 **500W** 条数据，记录查询一条数据所用时间，之后创建索引，后再查询一条数据，比较两条数据查询的时间。

```
--创建表
create table t_test(
tid number,
tname varchar2(30)
)
--创建序列
create sequence seq_test;
--插入数据
begin
for i in 1..5000000
loop
  insert into t_test values(seq_test.nextval,'测试数据'||i);
end loop;
commit;
end;
```

随便查询一条数据

```
select * from t_test where tname ='测试数据4789889'; 用时：  1.797s

--创建索引
create index index_test on t_test(tname)
select * from t_test where tname ='测试数据 4889889';用时：  0.047
```

哈哈 ~ 效果还是很明显的嘛~

- 同义词

```
CREATE [PUBLIC] SYNONYM synonym
FOR    object;
```

例如：

```
create public synonym emp for scott.emp;

select * from emp;

drop public synonym emp;
```

使用同义词的作用？

1. 可以很方便的访问其它用户的数据库对象
2. 缩短了对象名字的长度

四、数据的导入导出

当我们使用一个数据库时，总希望数据库的内容是可靠的、正确的，但由于计算机系统的故障（硬件故障、软件故障、网络故障、进程故障和系统故障）影响数据库系统的操作，影响数据库中数据的正确性，甚至破坏数据库，使数据库中全部或部分数据丢失。因此当发生上述故障后，希望能重构这个完整的数据库该处理称为数据库恢复，而要进行数据库的恢复必须要有数据库的备份工作。

Oracle 导出的数据文件最常见的有一下几种类型：

第一种是导出为.dmp 的文件格式，.dmp 文件是二进制的，可以跨平台，还能包含权限，效率也很不错，用得最广 。

第二种是导出为.sql 文件的，可用文本编辑器查看，通用性比较好，但效率不如第一种，适合小数据量导入导出。尤其注意的是表中不能有大字段（blob,clob,long），如果有，会提示不能导出(提示如下： table contains one or more LONG columns cannot export in sql

format,user Pl/sql developer format instead)。

第三种是导出为.pde 格式的，.pde 为 Pl/sql developer 自有的文件格式，只能用 Pl/sql developer 自己导入导出，不能用编辑器查看。

如果使用 PL/SQLDEV 工具导出步骤：

1 tools ->export user object 选择选项，导出.sql 文件

2 tools ->export tables-> Oracle Export 选择选项导出.dmp 文件

导入步骤：

1 tools->import tables->SQL Inserts 导入.sql 文件

2 tools->import talbes->Oracle Import 然后再导入 dmp 文件

数据库的导入导出成功

如果没有工具可以使用命令：

● 整库导出与导入

整库导出命令

```
exp system/itcast full=y
```

添加参数 full=y 就是整库导出

```
连接到: Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
已导出 ZHS16GBK 字符集和 AL16UTF16 NCHAR 字符集

即将导出整个数据库...
. 正在导出表空间定义
. 正在导出概要文件
. 正在导出用户定义
. 正在导出角色
. 正在导出资源成本
. 正在导出回退段定义
. 正在导出数据库链接
. 正在导出序号
. 正在导出目录别名
. 正在导出上下文名称空间
. 正在导出外部函数库名
. 导出 PUBLIC 类型同义词
. 正在导出专用类型同义词
. 正在导出对象类型定义
. 正在导出系统过程对象和操作
```

执行命令后会在当前目录下生成一个叫 EXPDAT.DMP，此文件为备份文件。

如果想指定备份文件的名称，则添加 file 参数即可，命令如下

```
exp system/itcast file=文件名  full=y
```

整库导入命令

```
imp system/itcast full=y
```

此命令如果不指定 file 参数，则默认用备份文件 EXPDAT.DMP 进行导入

如果指定 file 参数，则按照 file 指定的备份文件进行恢复

```
imp system/itcast full=y file=itcastdb.dmp
```

● 按用户导出与导入

按用户导出

```
exp itcast001 /itcast file= itcast001.dmp
```

按用户导入(导入到另一个用户中)

```
imp itcast002/itcast02    file= itcast001.dmp full=y
```

● 按表导出与导入

按表导出

```
exp itcast001 /itcast file= itcast001.dmp tables=t_person,t_student
```

用 **tables** 参数指定需要导出的表，如果有多个表用逗号分割即可

按表导入

```
imp itcast001/itcast file= itcast001.dmp tables= t_ person,t_student
```