* 学习目标

 * 能够掌握Oracle的锁

   * S,X

   * DML锁

      * TX

      * TM

         * S

         * X

         * RS（SS）

         * RX（SX）

         * SRX（SSX）

     * 死锁

 * 能够掌握Oracle常用的运算符

      * ||

      * union

      * union all

      * interserct

      * minus

 * 能够掌握Oracle常用的函数

      * to_char

      * to_date

      * to_number

      * rank() over(order by emp_age)

      * dense_rank() over(....)

      * row_numer over(...)

      * partition by(分区进行排名)

---

* 回顾

  * Oracle 表空间

    * dba_data_files

    * system账户登录

    * create tablespace lgdata datafiles 'C:/oracle-data/lg.dbf' size 20M next 10M

    * drop tablespace lgdata including contents and datafiles

    * create user lg identified by 123123 default tablespace lgdata

    * grant,revoke

  * 事务：确保数据库数据一致性，一组dml语句组成，要么一起成功，要么一起失败

    * '转'

  * 事务的四大特性：ACID

    * A：原子性

    * C：一致性

    * D：持久性

    * I（isolation）：隔离性

      * 隔离级别

        * ReadUncommited: 脏读，不可重复读，虚读

        * ReadCommited：避免脏读，不可重复读，虚读

        * Repeatable：避免脏读，避免不可重复读，虚读

        * Serializable:避免脏读，避免不可重复读，避免虚读

      * MySql:四种隔离级别都支持，默认隔离级别Repeatable

        * Oracle支持ReadCommited， Serializable，还提供Read only,默认隔离级别ReadCommited

    * JDBC 操作事务

      * Connection:setAutoCommited(false),commit,rollback(..,sp),savepoint

* 能够掌握Oracle的锁

  * 锁的概述

    * 锁是一种防止在访问共享数据的事务之间进行破坏性交互（即错误地更新数据或错误地

改变底层数据结构的交互）的机制。锁在维护数据库并发性和一致性方面起着至关重要的作用。

* 锁行为的摘要

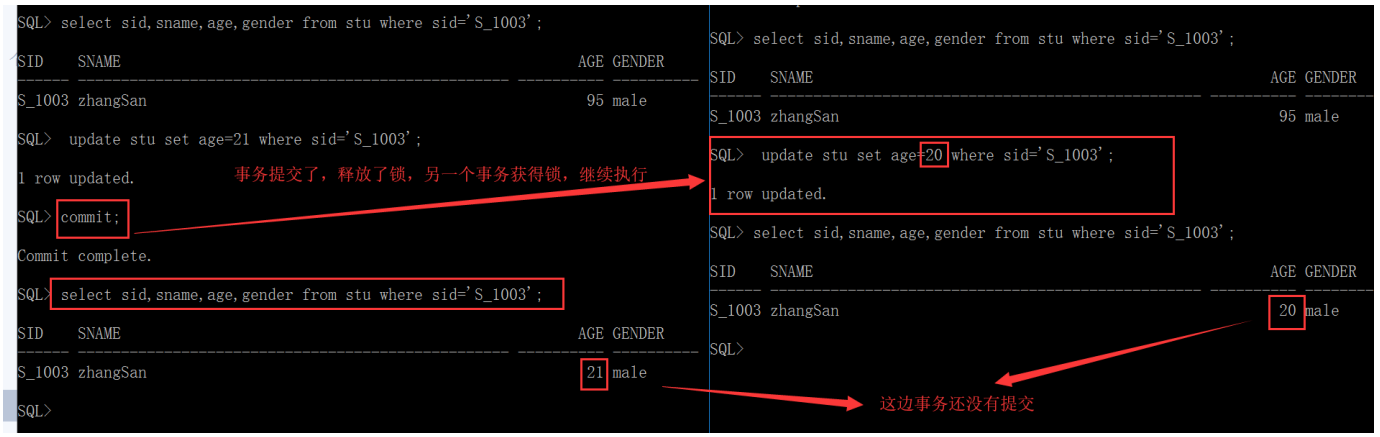
```
* 数据库通常使用两种类型的锁：排他锁和共享锁。
* 一个资源（如行或表）只能获得一个排他锁，但一个资源可以获得多个共享锁。
* 锁会影响readers和writers的交互。reader是对资源的查询，而writer是修改资源的语句。
* 以下规则总结了Oracle数据库对readers和writers的锁行为：
  * 当一个writer更新一行时，行才会被锁定，事务仅获取该行的锁。
    * 在正常情况下，数据库不会将行锁升级到块或表级别。
  * writer阻塞同一行的并发writer。
    * 如果一个事务正在修改行，则行锁将阻止不同事务同时修改同一行。
  * 一个reader从不阻塞一个writer。
   * 因为行的reader不会锁定它，所以writer可以修改此行。唯一的例外是select…for update
  * 一个writer从不阻塞reader。
   * 当一行被一个writer修改时，数据库将使用undo data数据为reader提供行的一致视图（避
```

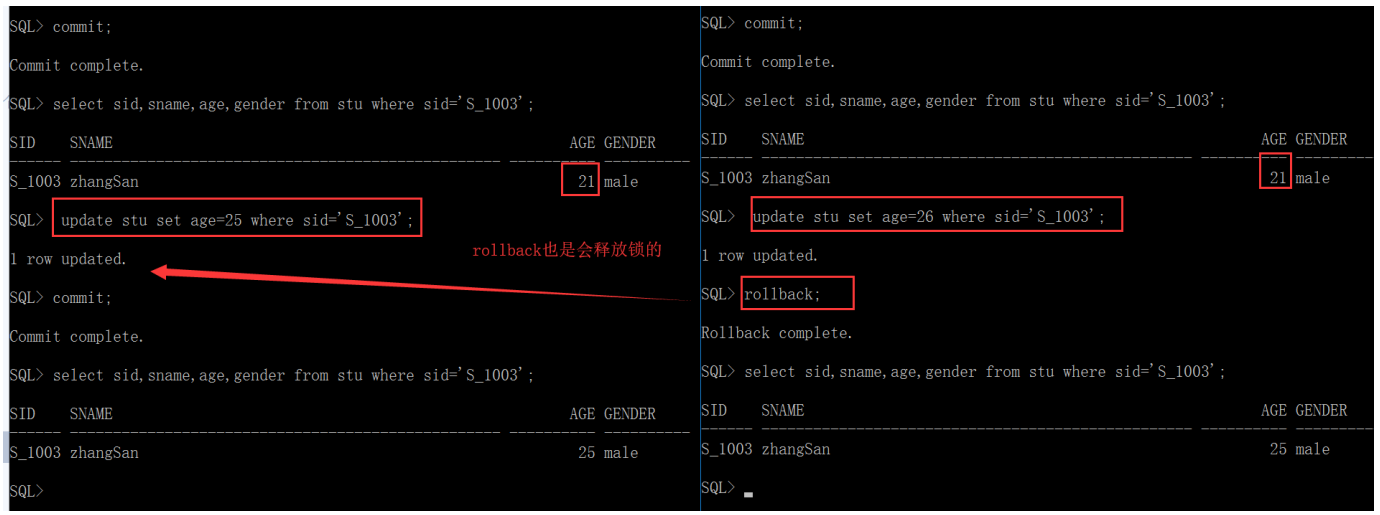
* 使用锁

* 锁是自动执行的，不需要用户操作。

```
* 在单用户数据库中，由于只有一个用户在修改信息，因此不需要锁定。
* 当多个用户访问和修改数据时，数据库必须提供一种防止同时修改同一数据的方法。
* 锁满足以下重要的数据库要求：
  * 一致性（Consistency）
   * 在用户完成之前，会话正在查看或更改数据确保数据不能被其他会话更改。
  * 完整性（Integrity）
   * 数据和结构必须以正确的顺序反映对它们所做的所有更改。
* oracle数据库通过其锁机制提供事务之间的数据并发性、一致性和完整性。
测试：
  * 关键sql
   * select sid,sname,age,gender from stu where sid='S_1003';
   * update stu set age=21 where sid='S_1003';
   * commit
   * rollback
```

* 使用锁效果截图



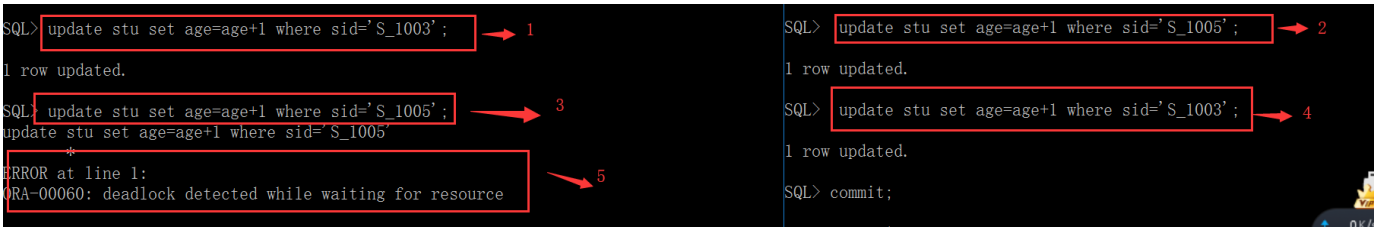


* 死锁

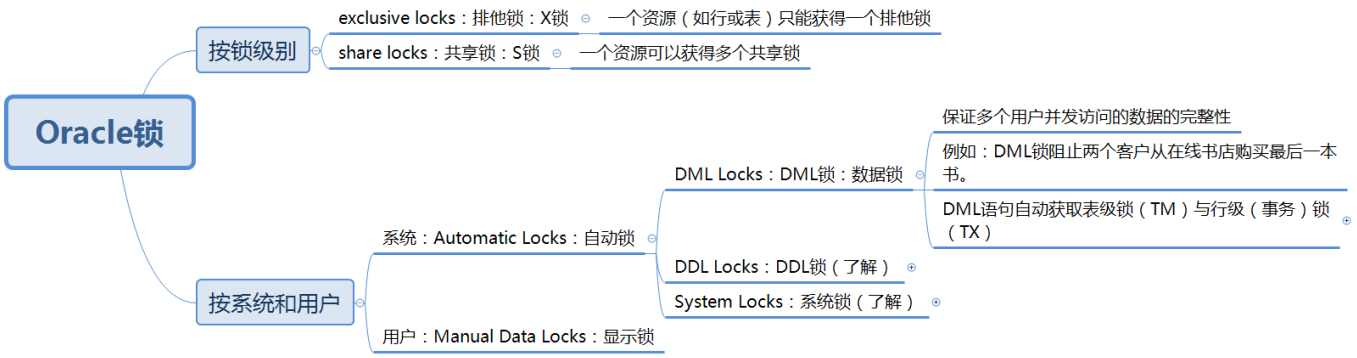
```
* 死锁是指两个或多个用户等待彼此锁定的数据的情况。
* 死锁会阻止某些事务继续工作。
* oracle数据库自动检测死锁并通过回滚死锁中涉及的一条语句来解决它们，释放一组冲突的行锁

* 演示死锁
  * 窗口1
    update stu set age=age+1 where sid='S_1003';
  * 窗口2
     update stu set age=age+1 where sid='S_1005';
  * 窗口1
    update stu set age=age+1 where sid='S_1005';
```

```
  * 窗口2
     update stu set age=age+1 where sid='S_1003';

温馨提醒：
* mysql 更新的时候，锁表的
```

* 死锁演示效果图



* 锁的分类



* 参考网络：https://docs.oracle.com/cd/E11882_01/server.112/e40540/consist.htm#CNCPT221

* 自动锁

  * 当进行一项数据库操作时，缺省情况下，系统自动为此数据库操作获得所有有必要的锁

```
* 自动锁的分类（DML锁，DDL锁，系统锁）
  * DML锁
    * 保证多个用户并发访问的数据的完整性，例如：DML锁阻止两个客户从在线书店购买最后一本
    * DML语句自动获取表级锁（TM）与行级（事务）锁（TX）
     * Row Locks：TX 锁：行级锁：事务锁
     * Table Locks :TM锁：表级锁（S，X，RS（SS），RX（SX），SRX）
```

```
      * Share Table Lock (S)：共享表锁
      * Exclusive Table Lock (X)：排他表锁
      * Row Share (RS)：a subshare table lock (SS)：行级共享锁
      * Row Exclusive Table Lock (RX)：a subexclusive table lock (SX)：行级排他锁
      * Share Row Exclusive Table Lock (SRX)：share-subexclusive table lock (SSX)
  * DDL锁(了解)
    * 保护模式对象的结构，例如表和视图的字典定义
  * 系统锁（了解）
    * 保护内部数据库结构，如数据文件。

* Row Locks：TX 锁：行级锁：事务锁
  * 锁一行
  * 获取行锁的方式
    * insert
    * update
    * delete
    * merge
    * select .... for update
  * 释放行锁的方式
    * commit
    * rollback
  * 行锁的机制
    * 行锁主要用作队列机制，以防止两个事务修改同一行。
    * 用了排他锁（X锁）：确保持有锁的事务提交或回滚之前，其他事务无法修改该行。
    * 行锁定提供了最好的粒度锁定，因此提供了最好的并发性和吞吐量。
  * 温馨提示：
    * 如果事务获取行的锁，则该事务还获取包含该行的表的锁。
      表锁防止发生冲突的ddl操作，这些操作将覆盖当前事务中的数据更改。
  * 演示案例（各自操作不影响的情况）
    窗口1
     select * from stu where sid in('S_1002','S_1005');
    窗口2
     select * from stu where sid in('S_1002','S_1005');
    窗口3
     select * from stu where sid in('S_1002','S_1005');
----------------------------------------------------------------------
    窗口1
     update stu set age=22 where sid='S_1002';
     select * from stu where sid in('S_1002','S_1005');
    窗口2
```

```
     select * from stu where sid in('S_1002','S_1005');
    窗口3
     select * from stu where sid in('S_1002','S_1005');
--------------------------------------------------------------------------------
    窗口2
     update stu set age=22 where sid='S_1005';
     select * from stu where sid in('S_1002','S_1005');
    窗口1
     select * from stu where sid in('S_1002','S_1005');
    窗口3
     select * from stu where sid in('S_1002','S_1005');

* Table Locks :TM锁：表级锁（S，X，RS，RX，，SRX）
  * 获取表锁的方式
    * insert
    * update
    * delete
    * merge
    * select .... for update
    * ...
  * 表锁可以在以下任何模式下保持
     * Share Table Lock (S)：共享表锁
      * 事务持有的共享表锁允许其他事务查询表（不使用select…for update），但仅当单个事
        由于多个事务可能同时持有共享表锁，因此持有此锁不足以确保事务可以修改表。
      * create index
     * Exclusive Table Lock (X)：排他表锁
       * 这个锁是最严格的，禁止其他事务执行任何类型的dml语句或将任何类型的锁放在表上。
       * alter table，drop table，drop index，trancute table
       * 表示持有表锁的事务已锁定表中的行并打算更新它们。
     * Row Share (RS)：a subshare table lock (SS)：行级共享锁
         行共享锁是表锁中限制最少的模式，为表提供最高程度的并发性。
     * Row Exclusive Table Lock (RX)：a subexclusive table lock (SX)：行级排他锁
       * 通常表示持有该锁的事务已经更新了表行或者已经发出select ... for update.
         RX锁允许其他事务同时查询、插入、更新、删除或锁定同一表中的行。
       * insert,update,delete，select... for update
     * Share Row Exclusive Table Lock (SRX)：share-subexclusive table lock (SSX
       * SRX比共享表锁更具限制性，同一时间只有一个事务可以获得SRX锁，
       * 持有SRX锁的事务允许其他事务查询表（除了select ... for update），不能更新表。
```

* 表锁模式

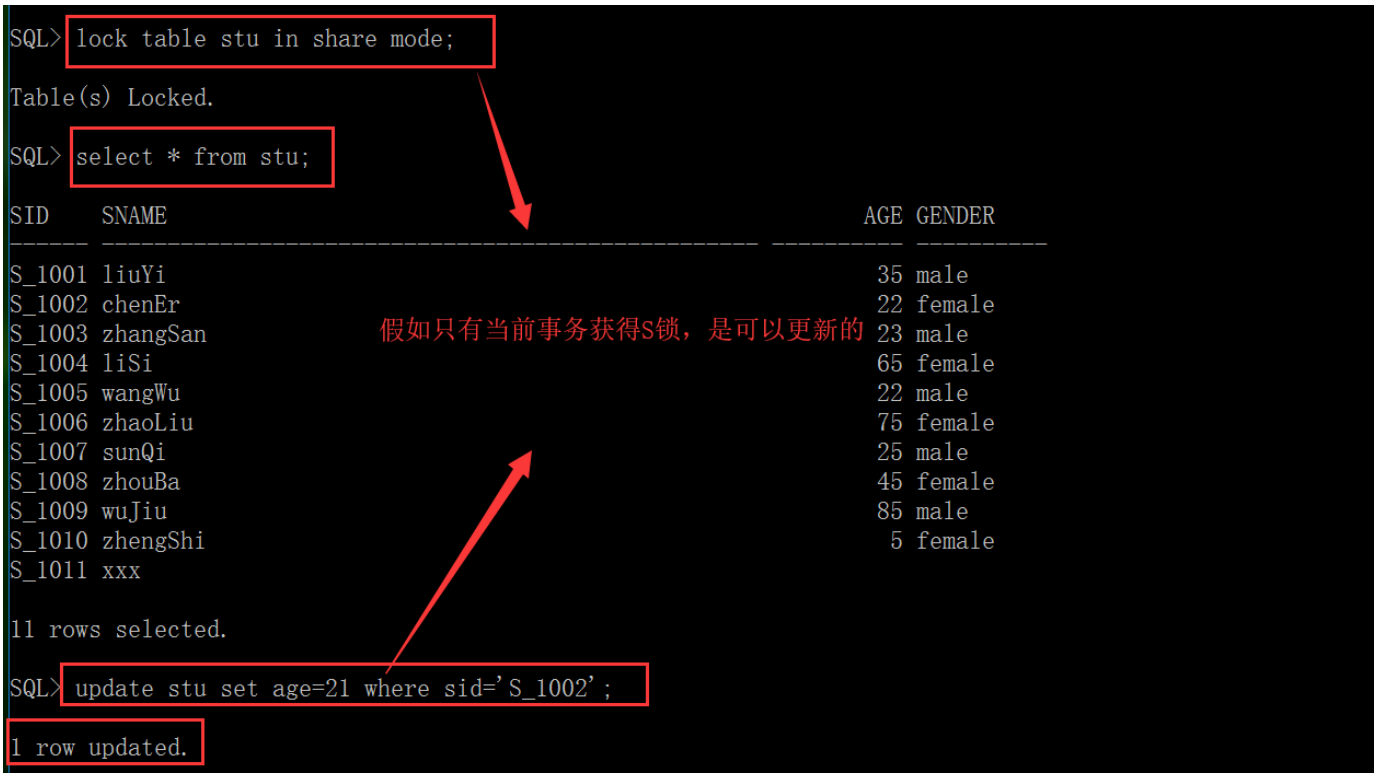
锁模式	锁描述	解释	相关SQL操作
0	NONE	不存在锁	
1	NULL	空锁，不与其他任何锁冲突	select
2	Row Share (RS)	也叫subshare table lock (SS) 行级共享锁	lock table in row share mode、lock table in share update mode
3	Row Exclusive Table Lock (RX)	也叫subexclusive table lock (SX) 行级独占锁	insert、delete、update、select ... for update、lock table in row exlusive mode
4	Share Table Lock (S)	表级共享锁	create index、lock table in share mode
5	Share Row Exclusive Table Lock (SRX)	也叫share-subexclusive table lock (SSX)	lock table in share row exclusive mode
6	Exclusive Table Lock (X)	表级独占锁	lock table in exclusive mode、alter/drop table、drop index、truncate table

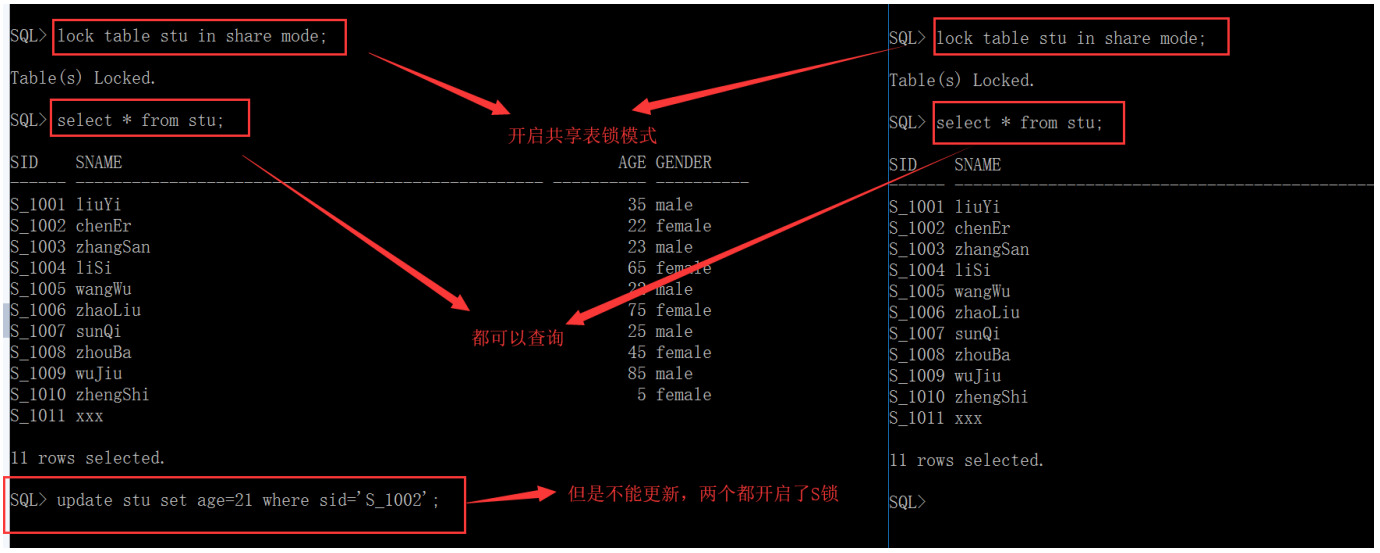
* 显示锁

```
* Oracle数据库自动执行锁，以确保数据并发性、数据完整性和语句级读取一致性。
  但是，您可以手动覆盖Oracle数据库的默认锁定机制。
*在以下情况下，重写默认锁定非常有用：
 * 应用程序需要可重复读取。
 * 应用程序要求事务具有对资源的独占访问权，这样事务就不必等待其他事务完成。
* 修改方式
  * SET TRANSACTION ISOLATION LEVEL（事务已经演示）
  * LOCK TABLE
     * lock table stu in share mode;(共享表锁S)
     * lock table stu in row share mode;（行级共享锁RS，SS）
     * lock table stu in share update mode;（行级共享锁RS，SS）
     * lock table stu in exclusive mode;（排他表锁X）
     * lock table stu in row exclusive mode;（行级排他锁，RX，SX）
     * lock table stu in share row exclusive mode;（共享行级排他锁，SRX，SSX）
  * SELECT ... FOR UPDATE

* 演示
 * 测试共享表锁（S）
  * 事务持有的共享表锁允许其他事务查询表（不使用select…for update），
    但仅当单个事务持有共享表锁时才允许更新。
    由于多个事务可能同时持有共享表锁，因此持有此锁不足以确保事务可以修改表。
  * 部分关键的SQL
    lock table stu in share mode;
    select * from stu;
    update stu set age=21 where sid='S_1002';
```

```
    commit;

* 测试排他表锁X
 * 这个锁是最严格的，禁止其他事务执行任何类型的dml语句或将任何类型的锁放在表上。
 * 部分关键的SQL
    lock table stu in exclusive mode;
    select * from stu;
    update stu set age=21 where sid='S_1002';
    commit;

* 测试共享行级锁
 * 表示持有表锁的事务已锁定表中的行并打算更新它们。
   行共享锁是表锁中限制最少的模式，为表提供最高程度的并发性。
    * lock table stu in row share mode;（行级共享锁RS，SS）
    * lock table stu in share update mode;（行级共享锁RS，SS）
* 测试行级排他锁（RX）
  * 通常表示持有该锁的事务已经更新了表行或者已经发出select ... for update.
    RX锁允许其他事务同时查询、插入、更新、删除或锁定同一表中的行。
   * lock table stu in row exclusive mode;
   * 测试结果和共享行级锁差不多
     * 他们小的区别是在：该锁的事务是否已经更新了表行或者已经发出select ... for upd
        * 是的话：RX，不是的话：RS
 * 测试时共享行级排他锁（SRX）
    * SRX比共享表锁更具限制性，同一时间只有一个事务可以获得SRX锁，
    * 持有SRX锁的事务允许其他事务查询表（除了select ... for update），不能更新表。
    * lock table stu in share row exclusive mode;
 * select ... for update 的测试

```

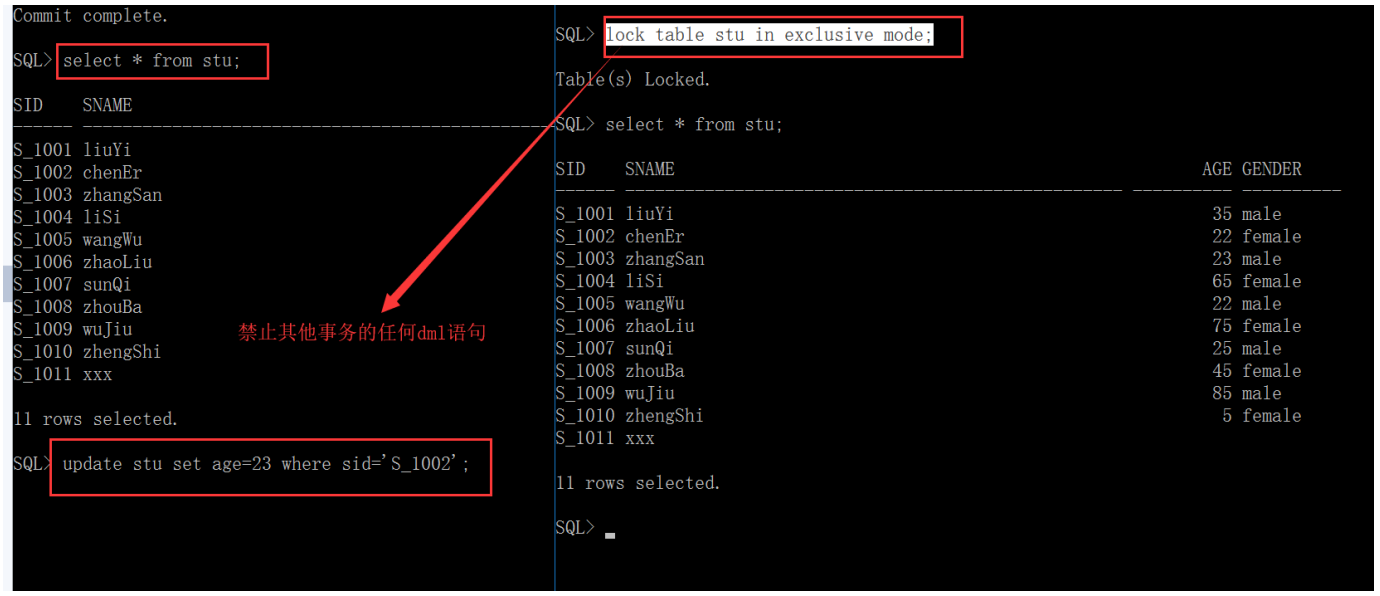
* 测试共享锁（S）效果图


SQL> lock table stu in share mode;
Table(s) Locked.
SQL> select * from stu;
SID SNAME AGE GENDER
S_1001 liuYi 35 male
S_1002 chenEr 22 female
S_1003 zhangSan 23 male
S_1004 liSi 65 female
S_1005 wangWu 22 male
S_1006 zhaoLiu 75 female
S_1007 sunQi 25 male
S_1008 zhouBa 45 female
S_1009 wuJiu 85 male
S_1010 zhengShi 5 female
S_1011 xxx
11 rows selected.
SQL> update stu set age=21 where sid='S_1002';
1 row updated.
假如只有当前事务获得S锁，是可以更新的

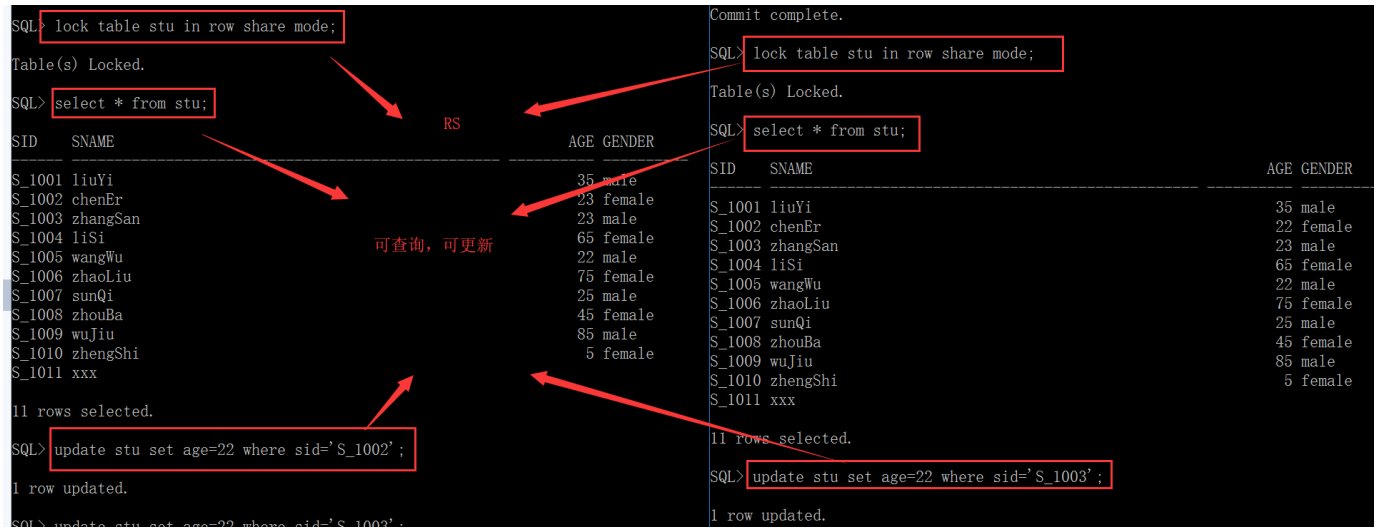

SQL> lock table stu in share mode;
Table(s) Locked.
SQL> select * from stu;
SID SNAME AGE GENDER
S_1001 liuYi 35 male
S_1002 chenEr 22 female
S_1003 zhangSan 23 male
S_1004 liSi 65 female
S_1005 wangWu 22 male
S_1006 zhaoLiu 75 female
S_1007 sunQi 25 male
S_1008 zhouBa 45 female
S_1009 wuJiu 85 male
S_1010 zhengShi 5 female
S_1011 xxx
11 rows selected.
SQL> update stu set age=21 where sid='S_1002';
开启共享表锁模式
都可以查询
但是不能更新，两个都开启了S锁
SQL> lock table stu in share mode;
Table(s) Locked.
SQL> select * from stu;
SID SNAME
S_1001 liuYi
S_1002 chenEr
S_1003 zhangSan
S_1004 liSi
S_1005 wangWu
S_1006 zhaoLiu
S_1007 sunQi
S_1008 zhouBa
S_1009 wuJiu
S_1010 zhengShi
S_1011 xxx
11 rows selected.
SQL>

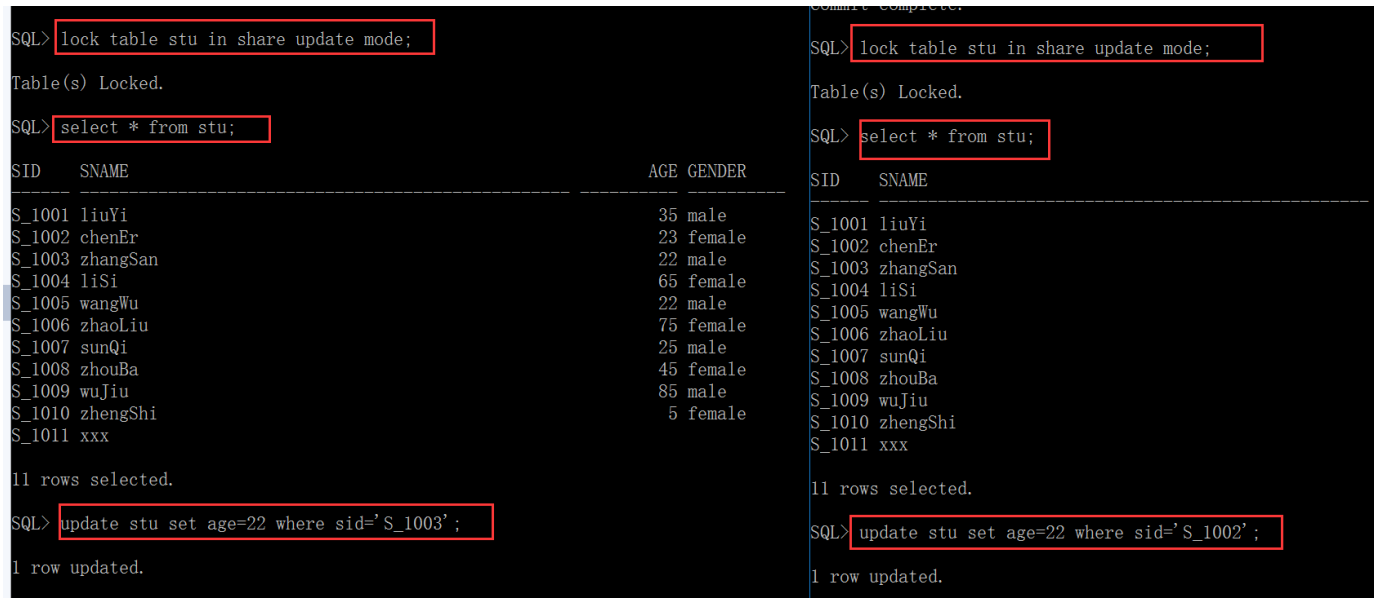
* 测试排他表锁


SQL> lock table stu in exclusive mode;
Table(s) Locked.
SQL>
获得排他锁
Commit complete.
SQL> lock table stu in exclusive mode;
获取不这个表排它锁了

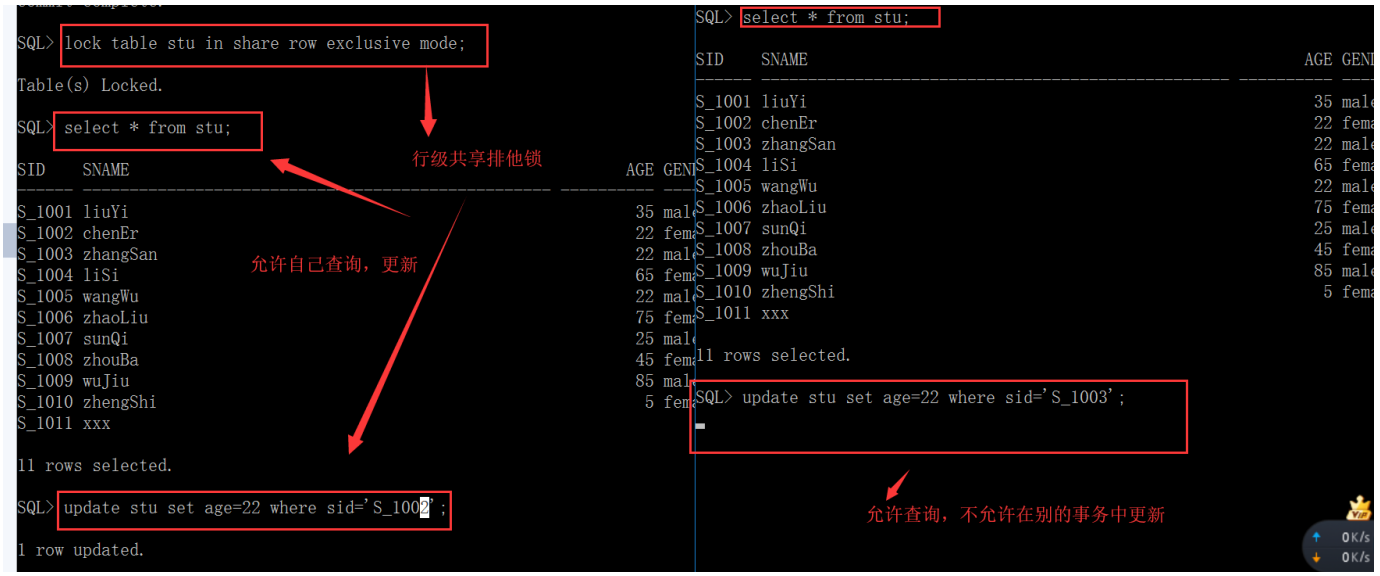


* 测试行级共享锁

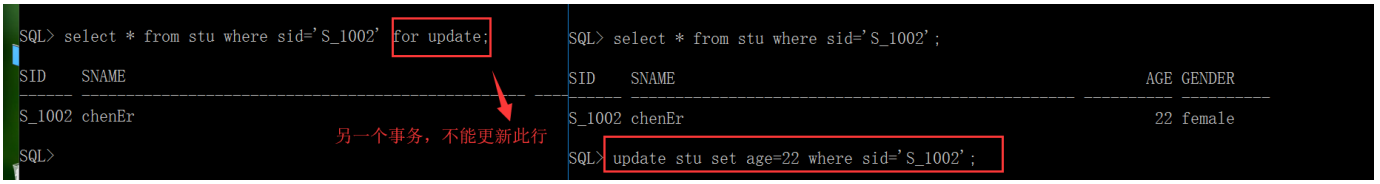




* 测试行级共享排他锁（SRX）



* 测试select ... for update



* 能够掌握Oracle常用的运算符

```
* 算术运算符
 +、-、*、/等略。
 例如： update stu set age=age+1 where sid='S_1002';
* 比较（关系）运算符
 =、！= （<>）、< 、 > 、 <= 、 >= 、 between...and... 、 in（not in） 、 like
* 逻辑运算符
 and 、 or 、 not
* 连接运算符
  ||
 select sid || sname from stu where sname='chenEr';
 结果：S_1002chenEr

* 集合运算符
 * union all（并集有重复）
```

```
 * union（并集无重复）
 * intersect（交集，共有部分）
 * minus（减集，第一个查询具有，第二个查询不具有的数据）
业务需求：
 * 公司需要修订员工信息，系统维护人员对员工表进行备份后，
 * 管理员操作了员工信息表，进行了数据处理，因为某种原因，操作失误，
 * 现领导需要查询所有员工的信息，可重复，没有变动的员工，
 * 如果有变动，找出哪些被修改过的员工，包括删除、修改、新增的。
步骤
 * 创建员工表
--drop table employee;
--drop table employee_bak
CREATE TABLE employee
(
     emp_no CHAR(8) PRIMARY KEY NOT NULL,   --工号，主键，非空
     emp_name VARCHAR2(30) NOT NULL,--姓名,非空
     emp_id VARCHAR2(18), --身份证号，代表18位整数
     emp_age NUMBER(3,0)  --年龄
);
 * 插入员工表原有员工数据
insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg001','张大','441
insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg002','张二','441
insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg003','张三','441
insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg004','张四','441

 * 维护人员备份数据，利用现有表创建新表（及数据），select 后边有多少字段，新表将有多少
CREATE TABLE employee_bak AS SELECT * FROM employee;
 * 管理员处理数据
update employee set emp_id='331521199909092115' where emp_no='lg003';
insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg005','张五','441
insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg006','张六','441
delete from  employee where emp_no ='lg004';
commit;
select * from employee;
select * from employee_bak;
 * 使用示例
--union使用示例，需要查看所有完整的员工的信息
select emp_no,emp_name,emp_id,emp_age from employee union select emp_no,emp_nam

--使用order by时 select后不能用* 号，如：select * from employee union select * fr
```

```
55
56 --union all,  --这里仍然以员工表为示例，查询所有员工的信息，可重复
57 select emp_no,emp_name,emp_id,emp_age from employee union all select emp_no,emp
58
59 --使用order by时 select后不能用* 号，如：select * from employee union select * fr
60
61 --intersect使用示例，找出信息没有变动的员工
62 select emp_no,emp_name,emp_id,emp_age from employee intersect  select emp_no,em
63
64 --mimus使用示例，找出备份后被删除或修改的员工信息
65 select emp_no,emp_name,emp_id,emp_age from employee_bak minus  select emp_no,em
66
67 --mimus使用示例，找出新增的员工信息
68 select * from employee where emp_no in(
69 select emp_no from employee minus  select emp_no from employee_bak
70 );
71
```

* 能够掌握Oracle常用的函数

```
1 * 多行函数
2  * sum:求和
3  * avg:求平均数
4  * count：计数
5  * max：求最大值
6  * min：求最小值
7  * 在分组与过滤group by与having使用
8
9 * 转换函数
10  * TO_CHAR
11  * TO_DATE
12  * TO_NUMBER
13 * 测试
14 -- 一般用来格式化日期数据或数值
15 select sysdate from dual;
16 select to_char(sysdate,'YYYY-MM-DD hh:mm:ss') from dual;
17 SELECT to_char（1210.7, '$9,999.00'） FROM dual;
18 -- 将字符串转为日期类型。
19 -- 示例：insert into 表名  values(列值, ……,to_date( '2014-10-10 12:12:22','YYYY-
```

```
  create table testDate(
      birthday date);
  insert into testDate(birthday) values(to_date('1999-10-10','YYYY-MM-DD'));
  select * from testDate;
-- 将字符串转为数字（通常能隐式转换）
SELECT to_number（'16'）+3 FROM dual;
结果等同于下边
SELECT '16'+3 FROM dual;

* 分析函数
 * 分析函数可以每个组返回多行。
 * RANK （排名）
 * DENSE_RANK （排名）
 * ROW_NUMBER（排名）
* 三个函数的使用示例：
CREATE TABLE employee
(
     emp_no CHAR(5) PRIMARY KEY NOT NULL,   --工号，主键，非空
     emp_name VARCHAR2(30) NOT NULL,--姓名,非空
     emp_id VARCHAR2(18), --身份证号，代表18位整数
     emp_age NUMBER(3,0)  --年龄
);
insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg001','张大','441
insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg002','张二','441
insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg003','张三','441
insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg004','张四','441
insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg005','张三','441
insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg006','张四','441
insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg007','张7','4415
commit;
select t.* from employee t;
排名不分区
select t.*,rank() over(order by emp_age desc) from employee t;
  * 排名结果：1 2 3  3  5 6 7
  * RANK使用相同排序排名一样，后继数据空出排名
select t.*,dense_rank() over(order by emp_age desc) from employee t;
  * 排名结果：1 2 3 3 4 5 6
  * DENSE_RANK使用，使用相同排序排名一样，后继数据不空出排名
select t.*,row_number() over(order by emp_age desc) from employee t;
  * 排名结果：1 2 3 4 5 6 7
```

```
  * ROW_NUMBER使用，不管排名是否一样，都按顺序排名

* 排名分区
 * PARTITION BY类似group by,根据ORDER BY排序字段的值重新由一开始排序。
select t.*,rank() over(partition by emp_name order by emp_age desc) from emplo
  * 排名结果：1 1 1  1 2 1 2
  * 在相同名字里排名
select t.*,dense_rank() over(partition by emp_name order by emp_age desc) from
  * 排名结果：1 1 1  1 2 1 2
  * 在相同名字里排名

select t.*,row_number() over(partition by emp_name order by emp_age desc) from
  * 排名结果：1 1 1  1 2 1 2
  * 在相同名字里排名


* 其他的函数
 * NVL(EXP1,EXP2): EXP1的值不为null，返回自己，否则返回EXP2;
 * NVL2(EXP1,EXP2,EXP3): EXP1的值不为null，将返回EXP2,否则返回EXP3;
 * DECODE(VALUE,IF1,THEN1,IF2,THEN2,...,ELSE):
   如果value等于if1,则返回then1,如果value等于if2,则返回then2,...否则返回else的值.

* 测试
--创建员工表,插入测试数据
--drop table employee;
CREATE TABLE employee
(
     emp_no CHAR(8) PRIMARY KEY NOT NULL,   --工号，主键，非空
     emp_name VARCHAR2(30) NOT NULL,--姓名,非空
     emp_id VARCHAR2(18), --身份证号，代表18位整数
     emp_address varchar2(30)
);
insert into employee(emp_no,emp_name,emp_id,emp_address) values('lg001','张大',
insert into employee(emp_no,emp_name,emp_id,emp_address) values('lg002','张二',
insert into employee(emp_no,emp_name,emp_id,emp_address) values('lg003','张三',
insert into employee(emp_no,emp_name,emp_id) values('lg004','张四','44152119990
commit;

select emp_no as 工号,nvl(emp_address,'没地址') as 住址 from employee;
select emp_no as 工号,nvl2(emp_address,'有地址','没地址') as 住址 from employee;
```

```
100 select emp_no 工号,decode(emp_address,4401,'广州','4403','深圳','其它') 住址  fr
```