www.itsci.mju.ac.th/sayan

LEC 02: การจัดการข้อมูล (DML)

SAYAN UNANKARD

1/2558

DATA MANIPULATION LANGUAGE

คำสั่ง DML จะถูกใช้เมื่อ

- เพิ่มข้อมูลแถวใหม่ลงในตาราง
- แก้ไขข้อมูลในตาราง
- ลบข้อมูลที่ไม่ต้องการออกจากตาราง

ทรานแซคชัน (transaction) ประกอบด้วยชุดคำสั่งของ DML ซึ่งเป็นหน่วยของการทำงาน

ADDING A NEW ROW TO A TABLE

70	Public Relations	100	1700

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

9	70	Public Relations	100	1700

INSERT STATEMENT SYNTAX

การเพิ่มแถวใหม่ลงในตารางโดยใช้คำสั่ง INSERT

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

คำสั่งนี้จะเพิ่มข้อมูลเพียง 1 แถว

INSERTING NEW ROWS

การเพิ่มข้อมูลแถวใหม่ ประกอบด้วยคำสั่งที่ระบุคอลัมน์ที่จะเพิ่ม

หากไม่ระบุคอลัมน์ที่ต้องการเพิ่ม จะต้องเรียงลำดับข้อมูลตามคอลัมน์ในตารางนั้น ๆ

```
INSERT INTO departments(department_id,
      department_name, manager_id, location_id)
VALUES (70, 'Public Relations', 100, 1700);
```

```
1 rows inserted
```

ข้อมูลที่เป็นตัวอักษรจะต้องอยู่ภายในเครื่องหมาย single quotation mark เช่น ‘text’

INSERTING ROWS WITH NULL VALUES

Implicit method: การละเว้นบางคอลัมน์

```
INSERT INTOdepartments (department_id, department_name)
VALUES        (30, 'Purchasing');
```

```
1 rows inserted
```

Explicit method: การกำหนดค่าว่าง NULL keyword ในตาราง

```
INSERT INTOdepartments
VALUES        (100, 'Finance', NULL, NULL);
```

```
1 rows inserted
```

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID

INSERTING SPECIAL VALUES

การเพิ่มข้อมูลโดยใช้ฟังก์ชัน SYSDATE ซึ่งแสดงค่าวันและเวลาปัจจุบันของเครื่อง

```
INSERT INTO employees (employee_id,
                       first_name, last_name,
                       email, phone_number,
                       hire_date, job_id, salary,
                       commission_pct, manager_id,
                       department_id)
VALUES                 (113,
                       'Louis', 'Popp',
                       'LPOPP', '515.124.4567',
                       SYSDATE, 'AC_ACCOUNT', 6900,
                       NULL, 205, 110);
```

```
1 rows inserted
```

INSERTING SPECIFIC DATE AND TIME VALUES

- การเพิ่มข้อมูลพนักงานโดยระบุวันที่ตามที่กำหนด

```
INSERT INTO employees
VALUES      (114,
             'Den', 'Raphealy',
             'DRAPHEAL', '515.127.4561',
             TO_DATE('FEB 3, 1999', 'MON DD, YYYY'),
             'SA_REP', 11000, 0.2, 100, 60);
```

```
1 rows inserted
```

- Verify your addition.

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT
114	Den	Raphealy	DRAPHEAL	515.127.4561	03-FEB-99	SA_REP	11000	0.2

To_Date('13/11/2008', 'dd/mm/yyyy')

COPYING ROWS FROM ANOTHER TABLE

การเขียนคำสั่งเพิ่มข้อมูลโดยใช้คำสั่งในการคิวรีข้อมูล

```
INSERT INTO sales_reps(id, name, salary, commission_pct)
  SELECT employee_id, last_name, salary, commission_pct
  FROM   employees
  WHERE  job_id LIKE '%REP%';

4 rows inserted
```

ไม่มีการใช้คำสั่ง VALUES

คอลัมน์ของคิวรีที่เลือกข้อมูลจะต้องตรงกับคอลัมน์ในตารางที่ต้องการเพิ่ม

ข้อมูลจะถูกเพิ่มลงในตารางตามจำนวนแถวที่เลือกมา

CHANGING DATA IN A TABLE

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	MANAGER_ID	COMMISSION_PCT	DEPARTMENT_ID
100	Steven	King	24000	(null)	(null)	90
101	Neena	Kochhar	17000	100	(null)	90
102	Lex	De Haan	17000	100	(null)	90
103	Alexander	Hunold	9000	102	(null)	60
104	Bruce	Ernst	6000	103	(null)	60
107	Diana	Lorentz	4200	103	(null)	60
124	Kevin	Mourgos	5800	100	(null)	50

Update rows in the `EMPLOYEES` table:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	MANAGER_ID	COMMISSION_PCT	DEPARTMENT_ID
100	Steven	King	24000	(null)	(null)	90
101	Neena	Kochhar	17000	100	(null)	90
102	Lex	De Haan	17000	100	(null)	90
103	Alexander	Hunold	9000	102	(null)	80
104	Bruce	Ernst	6000	103	(null)	80
107	Diana	Lorentz	4200	103	(null)	80
124	Kevin	Mourgos	5800	100	(null)	50

UPDATE STATEMENT SYNTAX

คำสั่งในการแก้ไขข้อมูล

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

สามารถแก้ไขข้อมูลได้หลายๆ แถวพร้อมกันได้

UPDATING ROWS IN A TABLE

การแก้ไขข้อมูลโดยระบุเฉพาะแถวที่ต้องการเท่านั้น เช่นต้องการแก้ไขเฉพาะแถวที่รหัสพนักงานเป็น 113

```
UPDATE employees
SET    department_id = 50
WHERE  employee_id = 113;
```

```
1 rows updated
```

การแก้ไขข้อมูลทุก ๆ แถวในตาราง โดยไม่จำเป็นต้องระบุคำสั่ง where

```
UPDATE     copy_emp
SET        department_id = 110;
```

```
22 rows updated
```

หากต้องการกำหนดค่าว่าง สามารถทำได้ดังนี้ column_name = NULL

UPDATING TWO COLUMNS WITH A SUBQUERY

Update employee 113's job and salary to match those of employee 205.

```
UPDATE   employees
SET      job_id  = (SELECT  job_id
                    FROM    employees
                    WHERE   employee_id = 205),
         salary  = (SELECT  salary
                    FROM    employees
                    WHERE   employee_id = 205)
WHERE    employee_id    =  113;
```

```
1 rows updated
```

REMOVING A ROW FROM A TABLE

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700

DELETE STATEMENT

คำสั่งในการลบข้อมูล

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

DELETING ROWS FROM A TABLE

การกำหนดข้อมูลเฉพาะที่ต้องการลบ เช่น ลบเฉพาะแผนก Finance

```
DELETE FROM departments
WHERE  department_name = 'Finance';
```

```
1 rows deleted
```

การลบข้อมูลจากตารางทั้งหมด

```
DELETE FROM  copy_emp;
```

```
22 rows deleted
```

DELETING ROWS BASED ON ANOTHER TABLE

ใช้คิวรีย่อยในการลบข้อมูล เพื่อลบแถวข้อมูลของตารางหลัก จากผลลัพธ์ที่ได้จากคิวรีย่อยในตารางอื่น ๆ

```
DELETE FROM employees
WHERE  department_id =
          (SELECT department_id
           FROM   departments
           WHERE  department_name
                  LIKE '%Public%');
```

```
1 rows deleted
```

TRUNCATE STATEMENT

เป็นคำสั่งที่ใช้ลบข้อมูลใน Record อย่างถาวร

ไม่สามารถกู้คืนข้อมูลได้เหมือนกับคำสั่ง Delete

```
TRUNCATE TABLE table_name;
```

Example:

```
TRUNCATE TABLE copy_emp;
```

DATABASE TRANSACTIONS

ทรานแซคชันของฐานข้อมูล (ช่วงเวลาตั้งแต่เริ่มเขียนคำสั่งไปจัดเก็บในฐานข้อมูลอย่างถาวร) ประกอบด้วย

- ชุดคำสั่งการจัดการข้อมูล (DML statements) ซึ่งประกอบด้วย คำสั่งในการเปลี่ยนแปลงข้อมูล
- หนึ่งคำสั่งการกำหนดโครงสร้างข้อมูล (DDL statement)
- หนึ่งคำสั่งในการควบคุมข้อมูล (DCL statement)

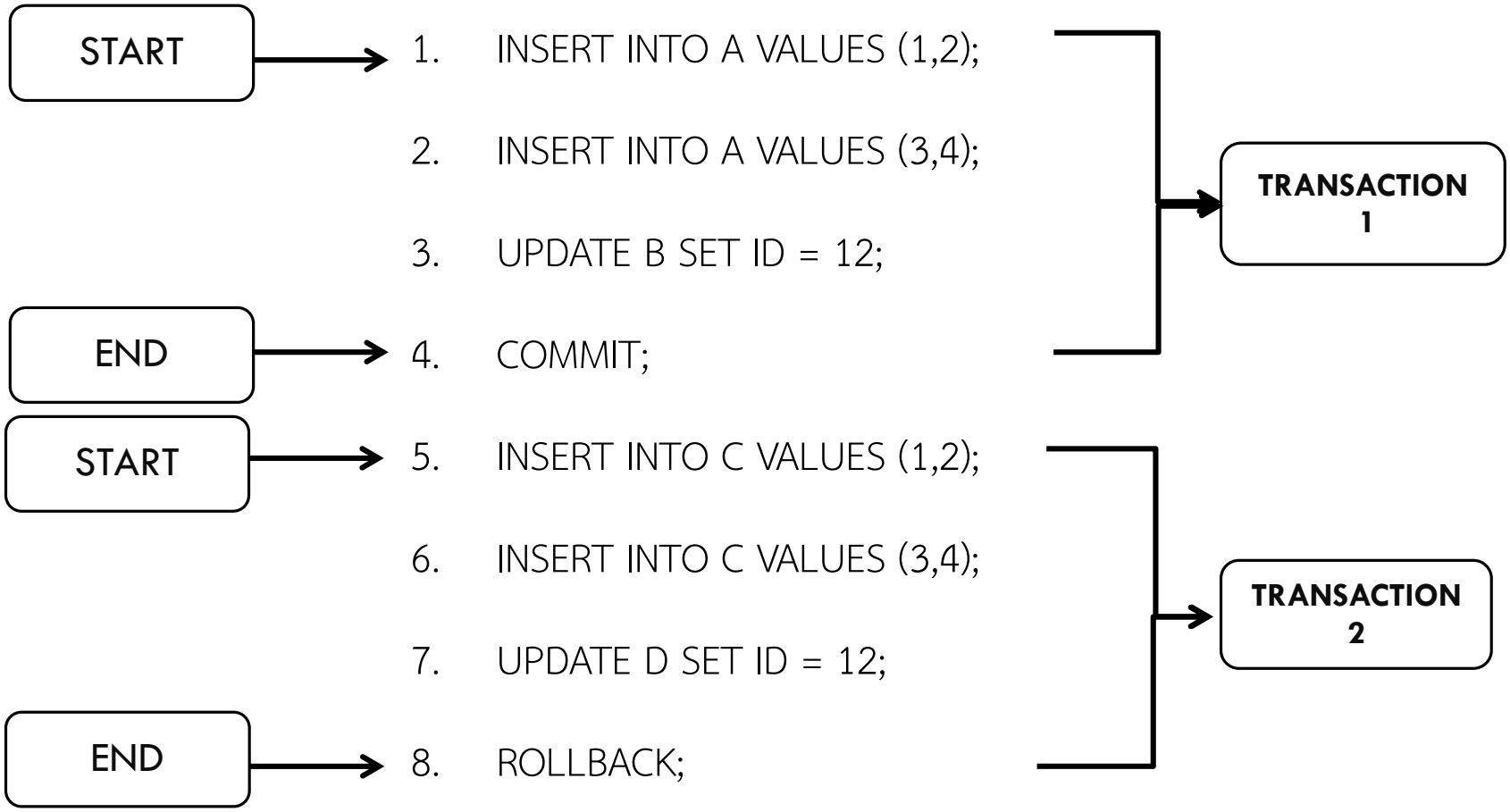
DATABASE TRANSACTIONS: START AND END

เริ่มเมื่อคำสั่งแรกถูกประมวลผล

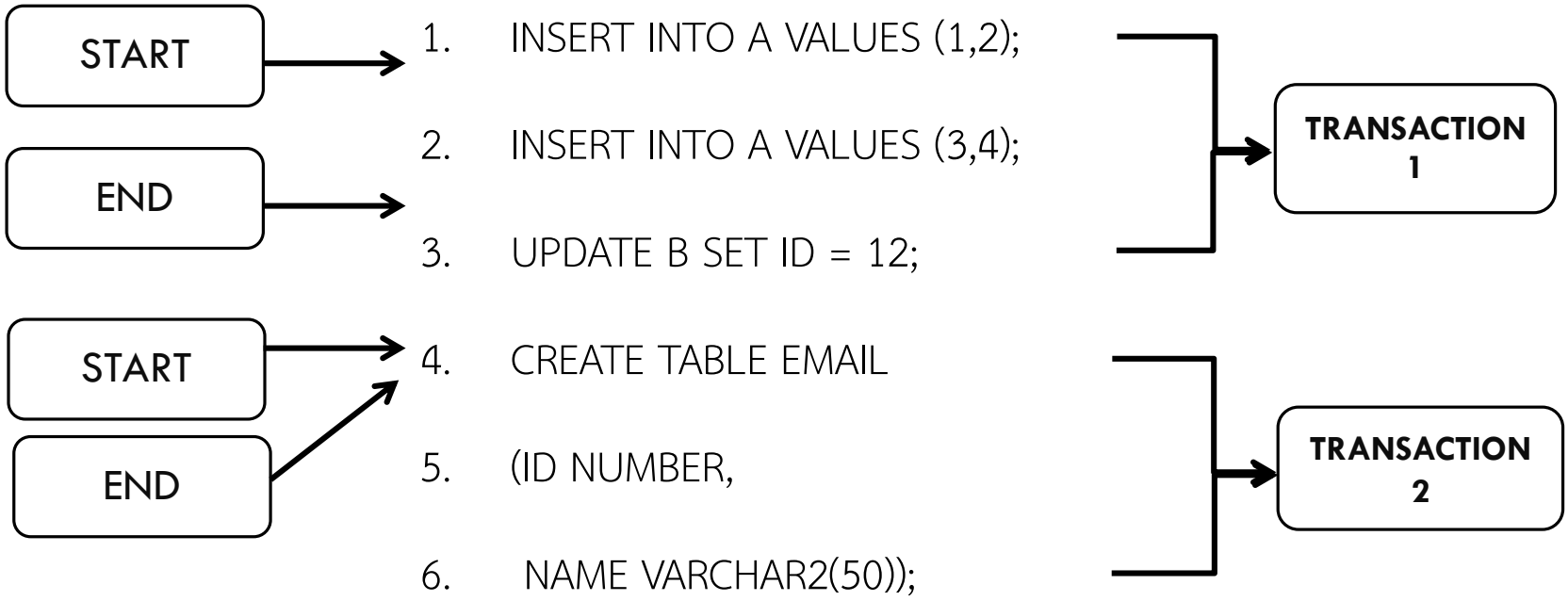
สิ้นสุดเมื่อเหตุการณ์ใดเหตุการณ์หนึ่งเกิดขึ้น ดังนี้

- เมื่อมีการ COMMIT หรือ ROLLBACK ข้อมูล
- เมื่อคำสั่งประเภท DDL หรือ DCL statement ถูกประมวลผล (automatic commit)
- เมื่อผู้ใช้ออกจากโปรแกรม SQL Developer หรือ SQL*Plus
- เมื่อระบบขัดข้อง

DATABASE TRANSACTIONS: START AND END
DML => เมื่อมีการ COMMIT หรือ ROLLBACK ข้อมูล



DATABASE TRANSACTIONS: START AND END
เมื่อคำสั่งประเภท DDL หรือ DCL STATEMENT

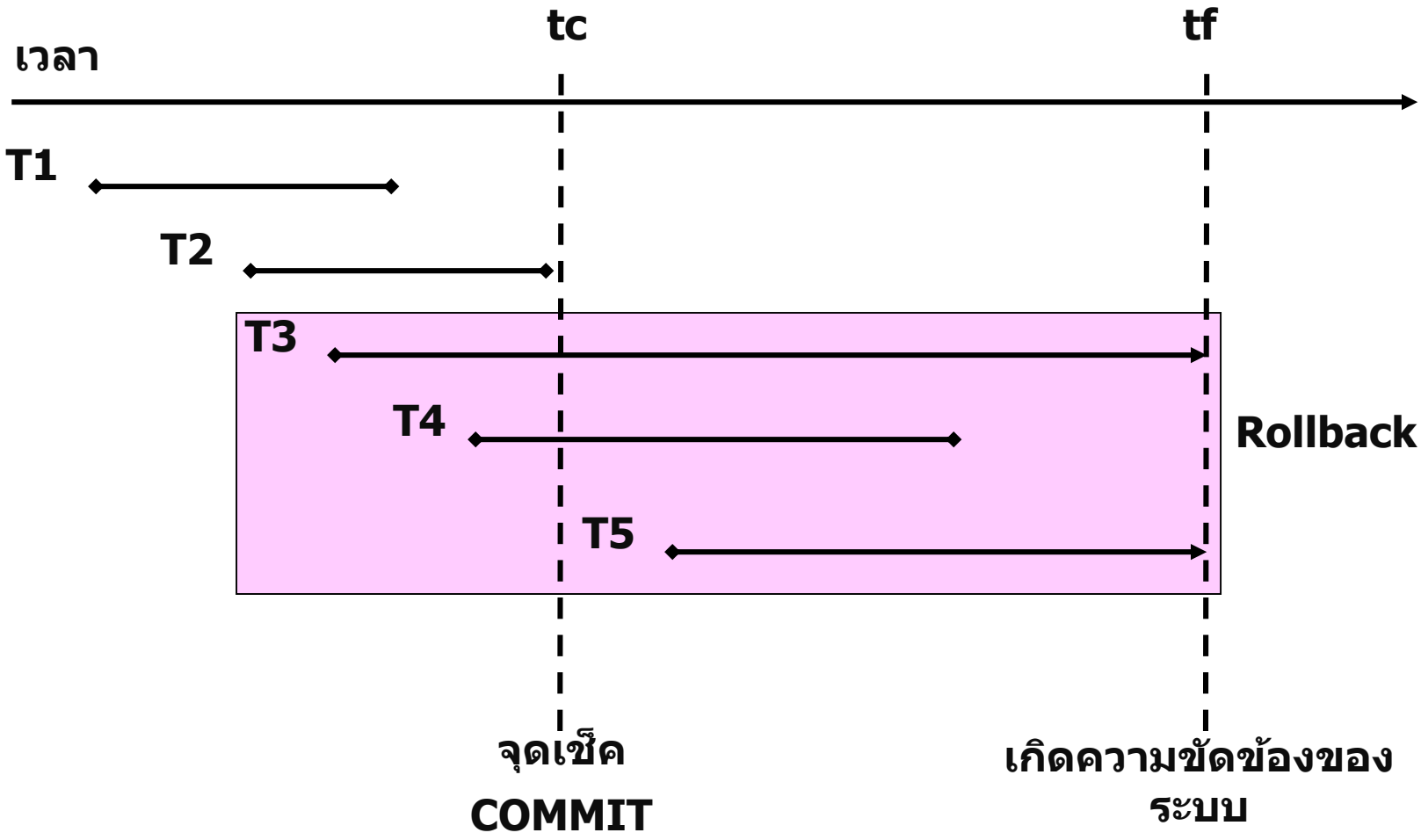


ADVANTAGES OF COMMIT AND ROLLBACK STATEMENTS

เมื่อมีการใช้คำสั่ง COMMIT และ ROLLBACK

- แน่ใจได้ว่าข้อมูลมีความถูกต้อง
- ดูการเปลี่ยนแปลงของข้อมูลก่อนที่จะทำการเปลี่ยนแปลงแบบถาวรได้
- จัดกลุ่มการทำงานของชุดคำสั่งให้ทำได้ต่อเนื่องกันได้

COMMIT AND ROLLBACK EXAMPLE



COMMIT AND ROLLBACK EXAMPLE

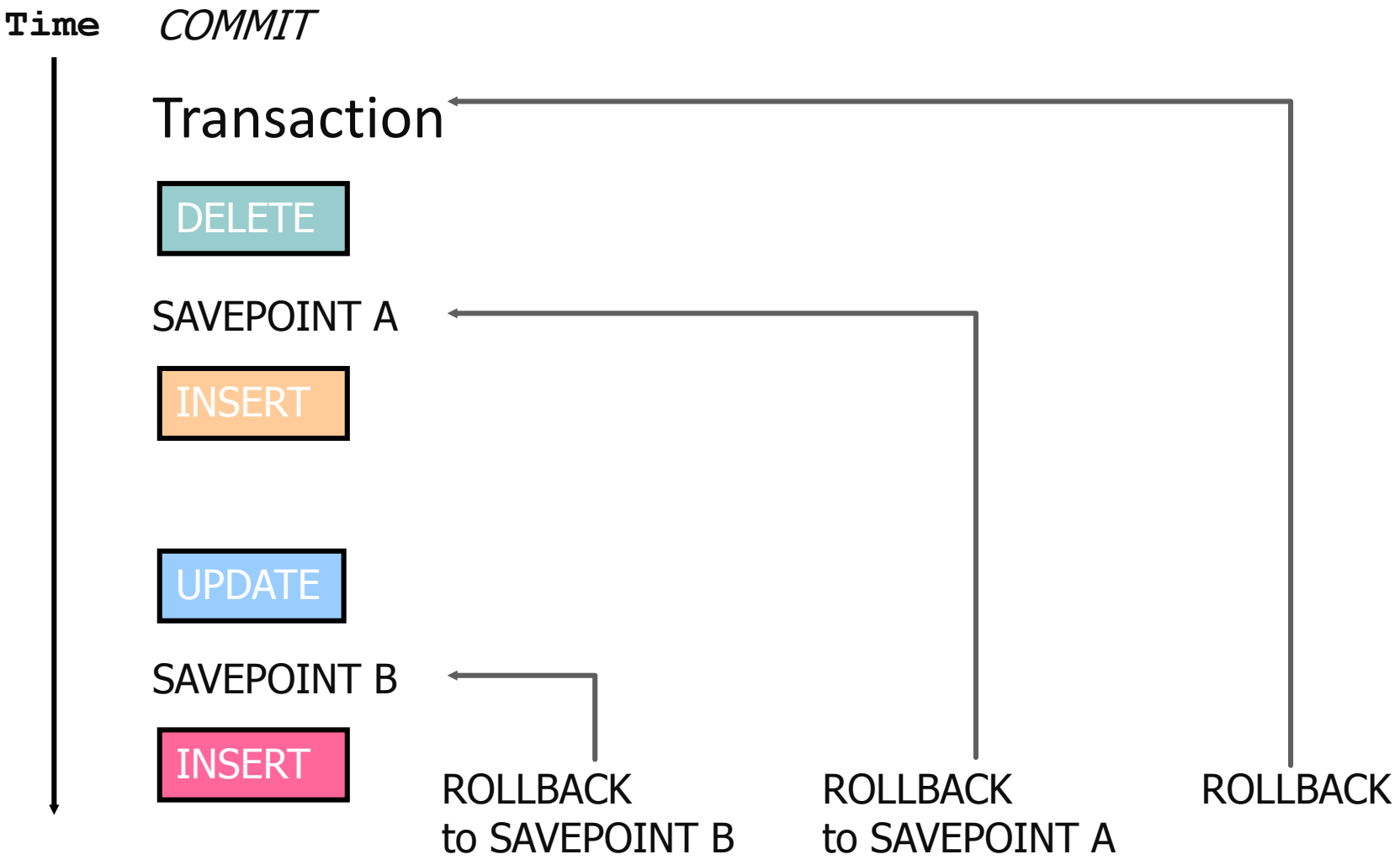
Transaction ที่ 1 และ 2 ได้รับการยืนยันการเปลี่ยนแปลงแล้ว ณ เวลา tc ดังนั้นเมื่อเวลา tf จึงไม่มีผลกระทบกับ Transaction 1 กับ 2

Transaction ที่ 3 ยังไม่เสร็จสิ้นการทำงานก่อนการขัดข้อง ดังนั้นเมื่อระบบเริ่มทำงานใหม่ Transaction ที่ 3 จะต้องกลับมาทำ ณ เวลา tc ใหม่

Transaction ที่ 4 ถึงแม้จะเสร็จสิ้นก่อนการขัดข้อง แต่ยังไม่ได้รับการยืนยันการเปลี่ยนแปลง ดังนั้นเมื่อระบบเริ่มทำงานใหม่ Transaction ที่ 4 จะต้องกลับมาทำ ณ เวลา tc ใหม่

Transaction ที่ 5 ยังไม่เสร็จสิ้นการทำงาน และ ยังไม่เคยยืนยันการเปลี่ยนแปลง ดังนั้น Transaction ที่ 5 จะต้องเริ่มต้นทำใหม่ทั้งหมด

EXPLICIT TRANSACTION CONTROL STATEMENTS



ROLLING BACK CHANGES TO A MARKER

การสร้างตำแหน่ง หรือ สัญลักษณ์ในทรานแซคชันนั้น ๆ โดยใช้คำสั่ง SAVEPOINT

เมื่อมีข้อผิดพลาด ใดๆ เกิดขึ้น สามารถใช้คำสั่งกู้คืนข้อมูลย้อนกลับได้ โดยใช้คำสั่ง Roll back กลับไปยังตำแหน่งที่ทำสัญลักษณ์ไว้ ดังคำสั่งต่อไปนี้ ROLLBACK TO SAVEPOINT

```
UPDATE...
SAVEPOINT update_done;
SAVEPOINT update_done succeeded.
INSERT...
ROLLBACK TO update_done;
ROLLBACK TO succeeded.
```

IMPLICIT TRANSACTION PROCESSING

คำสั่งในการยืนยันการบันทึกข้อมูลอัตโนมัติ (Automatic commit) จะทำก็ต่อเมื่อ

- มีการเรียกใช้คำสั่ง DDL
- มีการเรียกใช้คำสั่ง DCL
- เมื่อออกจากโปรแกรม SQL Developer หรือ SQL*Plus อย่างถูกต้องโดยไม่ได้ใช้คำสั่ง Commit หรือ Roll back

ระบบจะทำการกู้คืนข้อมูล (Automatic rollback) เมื่อมีการออกจากโปรแกรมกรณีไม่ปกติ หรือระบบล้มเหลว

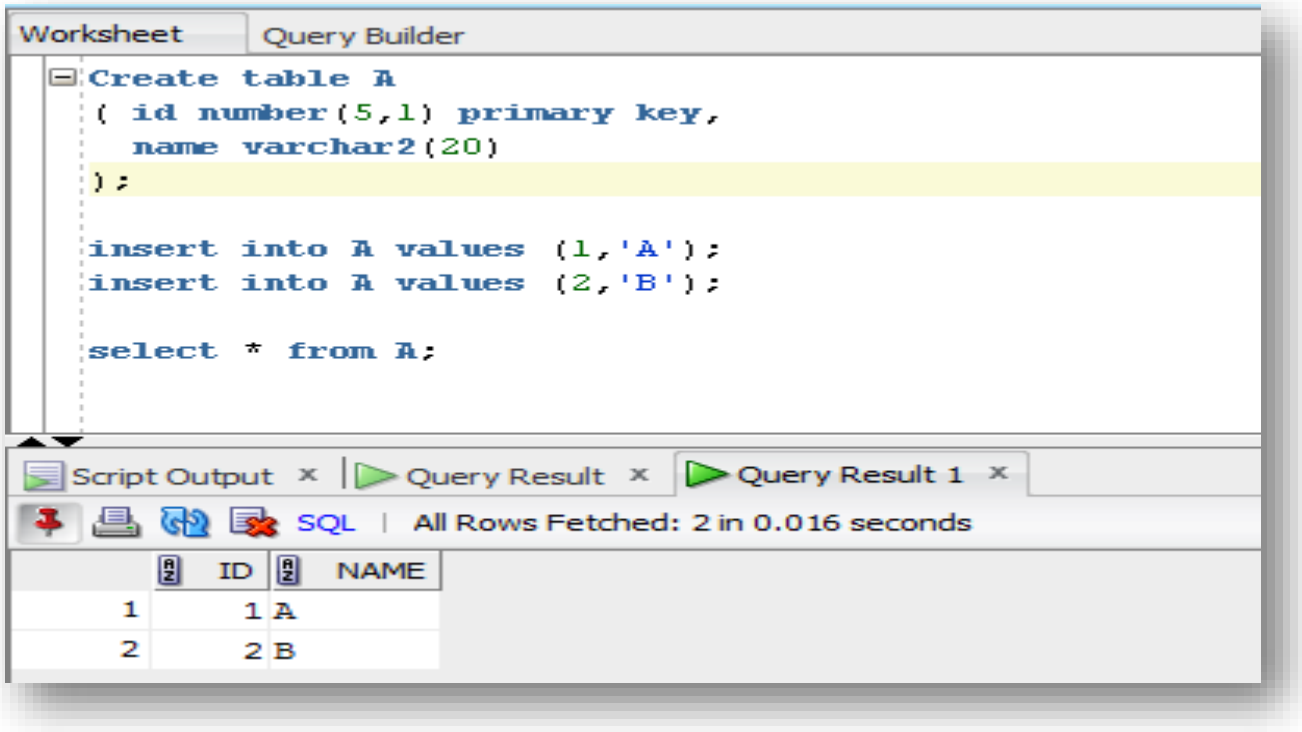
STATE OF THE DATA BEFORE COMMIT OR ROLLBACK

สถานะของข้อมูลก่อนที่จะมีการ Commit หรือ Rollback

- สถานะข้อมูลก่อนหน้าสามารถกู้คืนข้อมูลกลับมาได้
- ผู้ใช้ปัจจุบันสามารถดูผลลัพธ์ของคำสั่ง DML โดยใช้คำสั่ง SELECT statement
- ผู้ใช้คนอื่น ๆ จะไม่สามารถดูผลลัพธ์ของคำสั่ง DML ที่ทำโดยผู้ใช้ปัจจุบัน
- ผลลัพธ์ข้อมูลจะถูกล็อค โดยผู้ใช้คนอื่น ๆ จะไม่สามารถแก้ไข หรือเปลี่ยนแปลงข้อมูลที่ผู้ใช้คนปัจจุบันใช้ข้อมูลอยู่

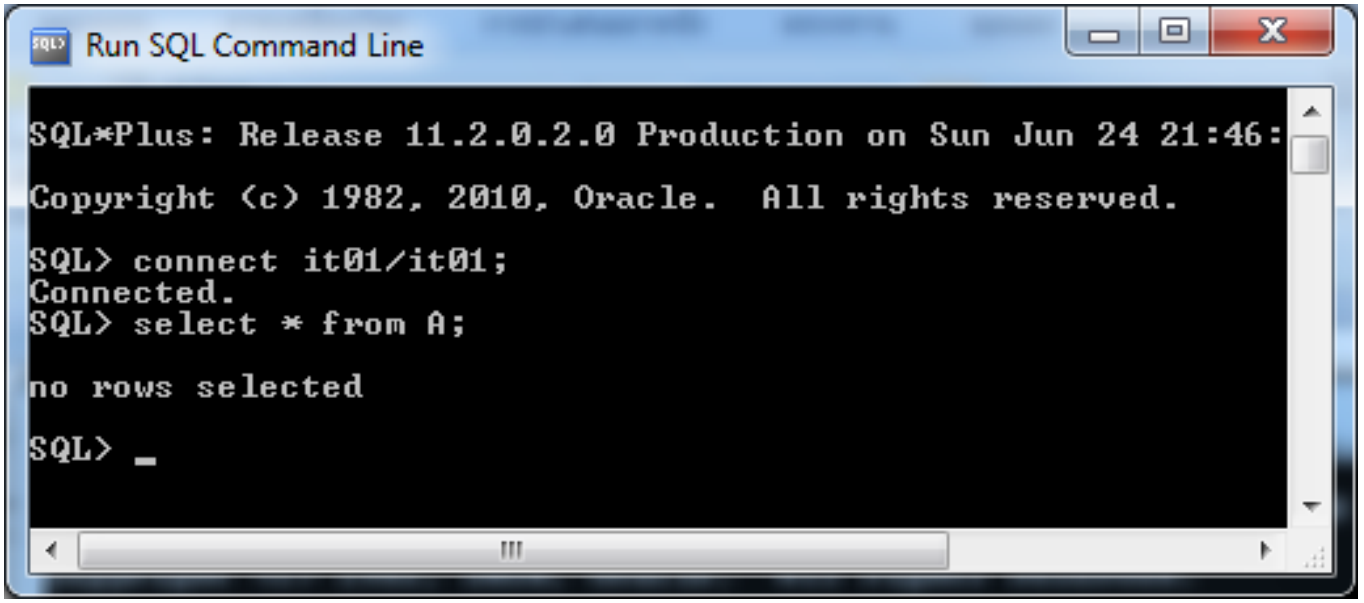
STATE OF THE DATA BEFORE COMMIT OR ROLLBACK

- ผู้ใช้ปัจจุบันสามารถดูผลลัพธ์ของคำสั่ง DML โดยใช้คำสั่ง SELECT statement



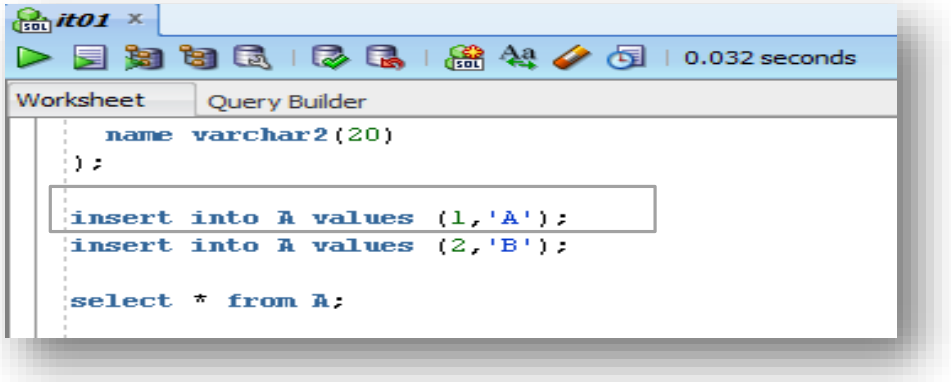
STATE OF THE DATA BEFORE COMMIT OR ROLLBACK

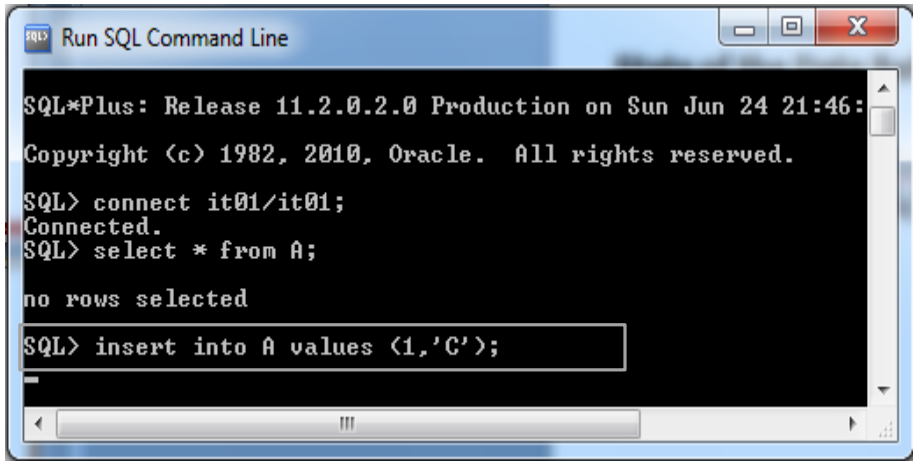
- ผู้ใช้คนอื่น ๆ จะไม่สามารถดูผลลัพธ์ของคำสั่ง DML ที่ทำโดยผู้ใช้ปัจจุบัน



STATE OF THE DATA BEFORE COMMIT OR ROLLBACK

- ผลลัพธ์ข้อมูลจะถูกล็อค โดยผู้ใช้คนอื่น ๆ จะไม่สามารถแก้ไข หรือเปลี่ยนแปลงข้อมูลที่ผู้ใช้คน





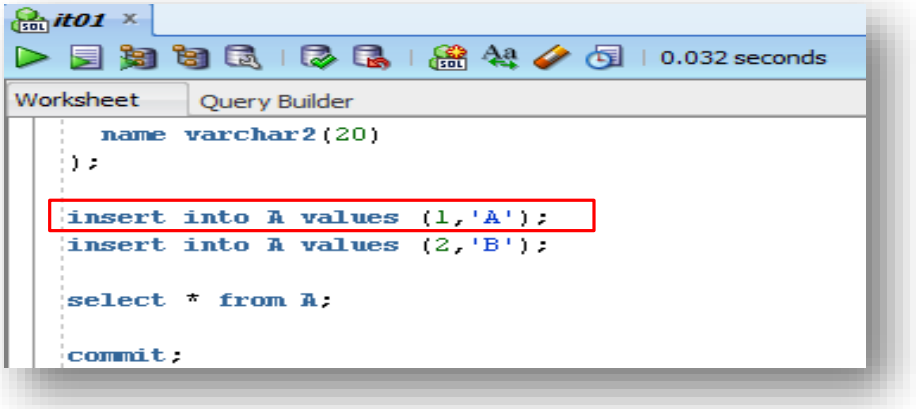
STATE OF THE DATA AFTER COMMIT

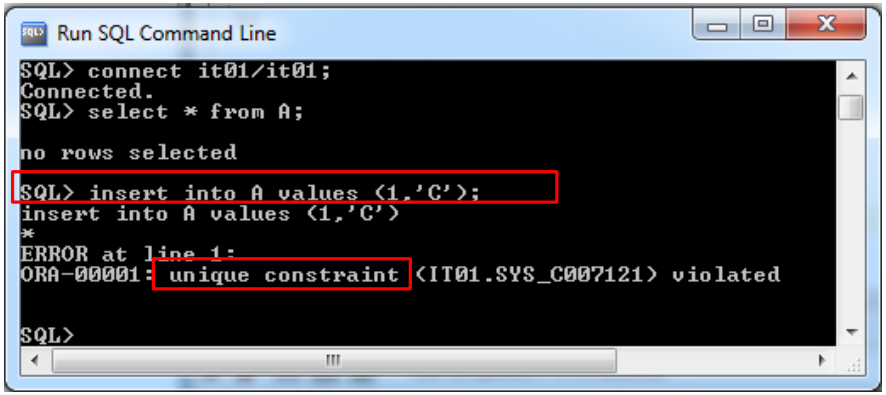
สถานะของข้อมูลหลังจาก Commit

- ข้อมูลที่มีการเปลี่ยนแปลงจะถูกบันทึกลงในฐานข้อมูล
- สถานะข้อมูลก่อนหน้าจะถูกเขียนทับทันที
- ผู้ใช้ทุกคนสามารถเห็นข้อมูลที่มีการแก้ไข
- ข้อมูลที่ถูกล็อคไว้จะถูกปลดล็อค และยินยอมให้ผู้ใช้คนอื่น ๆ เข้ามาจัดการข้อมูลได้
- ทุก ๆ จุด savepoints จะถูกลบทิ้งทั้งหมด

STATE OF THE DATA AFTER COMMIT

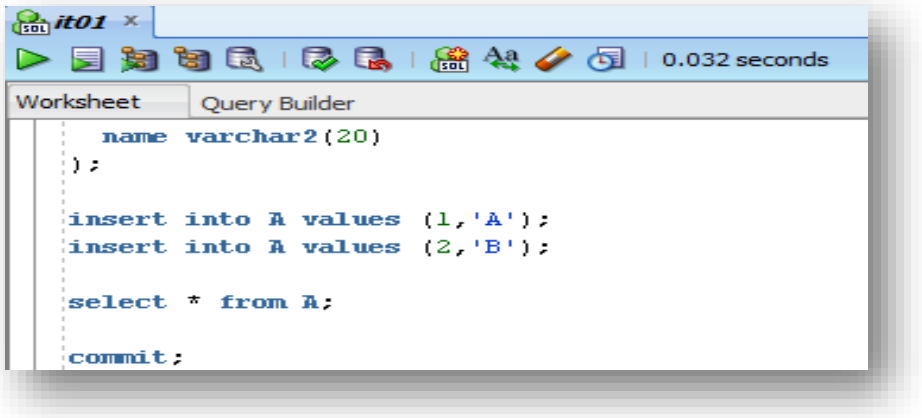
ข้อมูลที่ถูกล็อคไว้จะถูกปลดล็อค และยินยอมให้ผู้ใช้คนอื่น ๆ เข้ามาจัดการข้อมูลได้



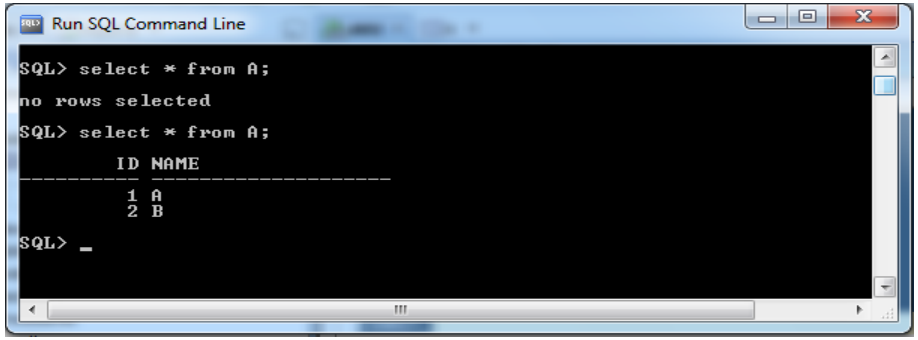


STATE OF THE DATA AFTER COMMIT

ผู้ใช้ทุกคนสามารถเห็นข้อมูลที่มีการแก้ไข



แสดงผลการจัดการข้อมูล



COMMITTING DATA

เมื่อมีการเปลี่ยนแปลงข้อมูลใด ๆ

```
DELETE FROM employees
WHERE  employee_id = 99999;
1 rows deleted

INSERT INTO departments
VALUES (290, 'Corporate Tax', NULL, 1700);
1 rows inserted
```

ยืนยันการเปลี่ยนแปลงข้อมูล

```
COMMIT;
COMMIT succeeded.
```

STATE OF THE DATA AFTER ROLLBACK

การยกเลิกการแก้ไขข้อมูล โดยใช้คำสั่ง ROLLBACK statement

- ข้อมูลที่มีการเปลี่ยนแปลงถูกกู้คืน
- สถานะของข้อมูลก่อนหน้าจะถูกกู้คืน
- ข้อมูลที่ถูกล็อคไว้ ถูกปลดล็อคให้ผู้ใช้คนอื่น ๆ เข้ามาจัดการได้

```
DELETE FROM copy_emp;
ROLLBACK ;
```

STATE OF THE DATA AFTER ROLLBACK: EXAMPLE

```
DELETE FROM test;
25,000 rows deleted.

ROLLBACK;
Rollback complete.

DELETE FROM test WHERE  id = 100;
1 row deleted.

SELECT * FROM   test WHERE  id = 100;
No rows selected.

COMMIT;
Commit complete.
```