

關聯式資料庫管理系統 結構化查詢語言(SQL) 使用MySQL

資料的維護與交易控制
Data Manipulation
Transaction Control

資料處理語言(DML)

- ▶ 資料處理語言(Data Manipulation Language)
 - INSERT:新增資料到資料表中
 - UPDATE:修改資料表中的資料
 - DELETE:刪除資料表中的資料
- ▶ DML 交易控制(DML Transaction Control)
 - COMMIT:確認交易
 - ROLLBACK:放棄交易
 - SAVEPOINT:設定交易儲存點

新增一筆記錄到資料表中

```
INSERT INTO dept(deptno, dname, loc)
VALUES (50, 'DEVELOPMENT', 'DETROIT');
```

DEPT

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

“...insert a new row
into DEPT table...”



DEPT

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

INSERT INTO 命令

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

- 可以用指定欄位或不指定欄位的方式來新增資料
 - 不指定欄位時，資料表中所有的欄位均要給值
 - 使用 DESC 指令可以得知欄位值的順序
- 欄位與值的數量需相同，位置對應且型態要相容

```
INSERT INTO dept(deptno, dname, loc)  
VALUES (50, 'MIS', 'NEW YORK');
```

指定欄位

```
INSERT INTO dept  
VALUES (50, 'MIS', 'NEW YORK');
```

不指定欄位

```
mysql> DESC DEPT;
```

Field	Type	Null	Key	Default	Extra
DEPTNO	smallint(6)		PRI	0	
DNAME	varchar(14)	YES		NULL	
LOC	varchar(13)	YES		NULL	

新增資料 - Null 值

▶ 新增空值到資料表的方法

- 未列舉的欄位, 新增資料時將值填入預設值(default value), 若該欄位沒有設定預設值則填入空值(NULL)

```
mysql> INSERT INTO dept(deptno, dname)
-> VALUES(60, 'MIS');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> SELECT *
-> FROM DEPT
-> WHERE DEPTNO=60;
+-----+-----+-----+
| DEPTNO | DNAME | LOC |
+-----+-----+-----+
|      60 | MIS   | NULL |
+-----+-----+-----+
1 row in set (0.00 sec)
```

- 使用 NULL 關鍵字

```
mysql> INSERT INTO dept(deptno, dname, loc)
-> VALUES(70, 'HR', NULL);
Query OK, 1 row affected (0.00 sec)
```

```
mysql> SELECT *
-> FROM DEPT
-> WHERE DEPTNO=60;
+-----+-----+-----+
| DEPTNO | DNAME | LOC |
+-----+-----+-----+
|      60 | MIS   | NULL |
+-----+-----+-----+
1 row in set (0.00 sec)
```

新增資料 - 日期資料

▶ 新增日期資料的方法

- 可以使用NOW()或CURRENT_DATE()...等函數
- 用字串輸入, 請參照資料型態章節

```
INSERT INTO EMP(empno,ename,job,mgr,hiredate,sal,comm,deptno)
VALUES(8001,'JAMES','ANALYST',7839,CURRENT_DATE(),2500,NULL,20);
```

```
INSERT INTO EMP(empno,ename,job,mgr,hiredate,sal,comm,deptno)
VALUES(8002,'JOHN','SALESMAN',7839,'1983-03-25',1400,500,30);
```

```
mysql> SELECT *
-> FROM emp
-> WHERE empno > 8000;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
8001	JAMES	ANALYST	7839	2004-07-05	2500	NULL	20
8002	JOHN	SALESMAN	7839	1983-03-25	1400	500	30

2 rows in set (0.01 sec)

修改資料表中的資料

EMP

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		10
7566	JONES	MANAGER		20
...				

```
UPDATE emp
SET   deptno = 20
WHERE empno = 7782;
```

“...update a row
in EMP table...”



EMP

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		20
7566	JONES	MANAGER		20
...				

UPDATE 命令

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

- 欄位與值的型態應相同
- 一次可以修改多個欄位, 用逗號隔開
- 若遺漏了WHERE子句則會更改資料表中所有的rows

```
UPDATE emp
SET sal = 1000
WHERE empno = 7900;
```

```
mysql> SELECT * FROM emp WHERE empno = 7900;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7900	JAMES	CLERK	7698	1981-10-03	1000	NULL	30

1 row in set (0.01 sec)

同時更新兩個欄位資料

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	NULL	1981-11-18	5000	NULL	10
7698	BLAKE	MANAGER	7839	1981-05-01	2850	NULL	30
7782	CLARK	MANAGER	7839	1981-06-09	2450	NULL	10
7566	JONES	MANAGER	7839	1981-04-02	2975	NULL	20
7654	MARTIN	SALESMAN	7698	1981-09-28	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-20	1600	300	30
7844	TURNER	SALESMAN	7698	1981-09-08	1500	0	30
7900	JAMES	CLERK	7698	1981-10-03	1000	NULL	30
7521	WARD	SALESMAN	7698	1981-02-22	1250	500	30
7902	FORD	ANALYST	7566	1981-10-03	3000	NULL	20
7369	SMITH	SALESMAN	7902	1980-10-17	800	NULL	30
7788	SCOTT	ANALYST	7566	1982-10-09	3000	NULL	20
7876	ADAMS	CLERK	7788	1983-01-12	1100	NULL	20
7934	MILLER	CLERK	7782	1982-01-23	1300	NULL	10

```

UPDATE emp
  SET job      = 'SALESMAN',
      deptno   = 30
 WHERE empno  = 7369;
  
```

使用子查詢來更新資料

▶ 使用子查詢

- 與子查詢合併使用時, 二者間資料表不可以是同一個

▶ 當取值使用

```
UPDATE emp
SET deptno = (SELECT deptno
              FROM dept
              WHERE dname='HR')
WHERE empno = 7900;
```

EMPNO	ENAME	JOB	SAL	DEPTNO
7566	JONES	MANAGER	2975	20
...				
7902	FORD	ANALYST	3000	20
7698	BLAKE	MANAGER	2850	30
7521	WARD	SALESMAN	1250	30
7900	JAMES	CLERK	1000	70
...				
8002	JOHN	SALESMAN	1400	30

● 當條件使用

```
UPDATE emp
SET sal = sal + 500
WHERE deptno = (SELECT deptno
                FROM dept
                WHERE dname='SALES');
```

以下為30部門員工異動後的薪水參考

EMPNO	SAL	SAL+500
7698	2850	3350
7654	1250	1750
7499	1600	2100
7844	1500	2000
7900	1000	1500
7521	1250	1750
8002	1400	1900

7 rows in set (0.00 sec)

刪除資料表中的資料


DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	DEVELOPMENT	DETROIT
60	MIS	
...		

```
DELETE FROM DEPT  
WHERE deptno = 50;
```

“...delete a row
from DEPT table...”

DEPT



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

DELETE FROM 命令

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

- 刪除資料表中的資料
- 若遺漏了WHERE子句時,則刪除資料表中所有的資料

```
DELETE FROM dept  
WHERE deptno = 70;
```



```
mysql> SELECT * FROM dept;  
+-----+-----+-----+  
| DEPTNO | DNAME      | LOC      |  
+-----+-----+-----+  
|      10 | ACCOUNTING | NEW YORK |  
|      20 | RESEARCH   | DALLAS   |  
|      30 | SALES      | CHICAGO  |  
|      40 | OPERATIONS | BOSTON   |  
|      50 | MIS        | NEW YORK |  
|      60 | MIS        | NULL     |  
+-----+-----+-----+  
6 rows in set (0.00 sec)
```

部門70的資料已被刪除

刪除資料表中的資料

- 指定刪除的對像

```
DELETE FROM dept  
WHERE dname = 'Finance';
```

- 省略WHERE子句刪除表格中全部的資料

```
DELETE FROM dept;
```

- 使用子查詢結果作為刪除資料的對像(條件)

```
DELETE FROM emp  
WHERE deptno = (SELECT deptno  
                FROM dept  
                WHERE dname LIKE '%Public%');
```

新增、修改與刪除資料的錯誤

▶ 常見的錯誤

- 違反資料檢查條件(Constraint)
 - 鍵值重複—違反了主鍵(primary key)或唯一鍵(Unique)
 - 違反外來鍵(Foreign Key)
 - 對非空值(NOT NULL)的欄位給定空值
- 資料長度過長
- 資料個數不夠
- 資料型態錯誤

```
UPDATE    emp
SET        deptno = 55
WHERE      deptno = 10;
```

ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`zb105`.`emp`, CONSTRAINT `EMP_DEPTNO_FK` FOREIGN KEY (`DEPTNO`) REFERENCES `dept` (`DEPTNO`))

Department number 55 does not exist

新增資料-違反 NOT NULL

- NOT NULL的錯誤

```
mysql> DESC EMP;
```

Field	Type	Null	Key	Default	Extra
EMPNO	smallint(6)		PRI	0	
ENAME	varchar(10)	YES		NULL	
JOB	varchar(9)	YES		NULL	
MGR	smallint(6)	YES		NULL	
HIREDATE	date	YES		NULL	
SAL	int(11)	YES		NULL	
COMM	int(11)	YES		NULL	
DEPTNO	smallint(6)			0	

8 rows in set (0.00 sec)

```
mysql> INSERT INTO emp(empno, ename, job, deptno )  
      ->   VALUES(9001, 'SANDY', 'CLERK', NULL);  
ERROR 1048 (23000): Column 'DEPTNO' cannot be null
```

修改資料-違反NOT NULL

▶ NOT NULL的錯誤

```
mysql> DESC EMP;
```

Field	Type	Null	Key	Default	Extra
EMPNO	smallint(6)		PRI	0	
ENAME	varchar(10)	YES		NULL	
JOB	varchar(9)	YES		NULL	
MGR	smallint(6)	YES		NULL	
HIREDATE	date	YES		NULL	
SAL	int(11)	YES		NULL	
COMM	int(11)	YES		NULL	
DEPTNO	smallint(6)			0	

8 rows in set (0.00 sec)

```
UPDATE EMP
```

```
  SET DEPTNO = NULL
```

```
WHERE EMPNO = 8001;
```


新增資料-違反外來鍵

▶ 新增資料時的錯誤

- 違反外來鍵(Foreign Key)的錯誤

PARENT	CHILD
TABLE : PKT	TABLE : FKT
+-----+	+-----+-----+
PK	FK C2
+-----+	+-----+-----+
1	1 ROW1
2	2 ROW2
3	2 ROW3
+-----+	1 ROW4
	3 ROW5
	3 ROW6
	3 ROW7
	+-----+-----+
	4 ROW8
	+-----+-----+

```
INSERT INTO FKT  
Values (4, 'ROW8')
```

在PARENT TABLE中並沒有 4 這筆資料, 所以無法新增

修改資料-違反外來鍵

▶ 修改資料時的錯誤

- 違反整合資料檢查條件(Integrity Constraint Error)

PARENT	CHILD
TABLE : PKT	TABLE : FKT
+-----+	+-----+-----+
PK	FK C2
+-----+	+-----+-----+
1	3 ROW1
2	2 ROW2
3	2 ROW3
+-----+	1 ROW4
	4 ROW5
	3 ROW6
	3 ROW7
	+-----+-----+

在PARENT TABLE中並沒有 4 這筆資料, 所以無法更新

刪除資料-違反外來鍵

▶ 刪除資料的錯誤

◦ Integrity Constraint Error

PARENT TABLE: PKT	CHILD TABLE: FKT
+-----+	+-----+-----+
PK	FK C2
+-----+	+-----+-----+
1	1 ROW1
2	2 ROW2
3	2 ROW3
+-----+	1 ROW4
	3 ROW5
	3 ROW6
	3 ROW7
	+-----+-----+

共3筆資料

在CHILD TABLE中有3筆資料參考著PARENT TABLE中 PK=3 這筆資料，故刪除PARENT TABLE中 PK=3 時會產生錯誤訊息，如下：

```
mysql> DELETE FROM pkt
-> WHERE pk = 3;
ERROR 1217 (23000): Cannot delete or update a parent row: a foreign key
constraint fails
```

從其他資料表複製資料

- ▶ 使用子查詢將其他資料表中的資料新增到指定的資料表

```
INSERT INTO table[(column,...)]  
  SELECT column,...  
  FROM table  
  WHERE ...
```

Sub-Query

- 範例:

```
INSERT INTO bonus(ename,job,sal,comm)  
  SELECT ename, job, sal, comm  
  FROM emp  
  WHERE deptno=10;
```

```
mysql> SELECT * FROM bonus;  
+-----+-----+-----+-----+  
| ENAME  | JOB      | SAL   | COMM  |  
+-----+-----+-----+-----+  
| KING   | PRESIDENT | 5000  | NULL  |  
| CLARK  | MANAGER   | 2450  | NULL  |  
| MILLER | CLERK     | 1300  | NULL  |  
+-----+-----+-----+-----+  
3 rows in set (0.00 sec)
```

資料庫交易 (Database Transactions)

- ▶ 交易 (Database Transactions)
 - 改變資料庫中現有資料
- ▶ 依不同類型的指令所產生的交易：
 - DML 交易
 - DDL 交易
 - DCL 交易
- ▶ MySQL 預設的交易處理方式：
 - DDL & DCL 交易 - Auto commit (自動確認)
 - DML 交易 - Auto commit (自動確認)



User Commit (使用者確認或放棄)

交易控制(Transactions Control)

▶ 交易控制的主要目的

- 維持資料庫中資料的一致性(Consistency)和正確性(Correctness)

▶ 何謂交易(Database Transactions)

- 是多個 DML statements 所構成的邏輯工作單位
- 交易(Transaction) 提供了一個資料處理的邏輯單元(Logic Unit)
 - 在該邏輯單元中
 - 如果全部執行成功，則會確定交易期間所修改的所有資料正式成為資料庫的內容(Commit)
 - 如果有發生錯誤，則必須取消或回復該交易期間內所有的資料修改(Rollback)
- 交易的結果:成功或失敗

MySQL 的交易控制(Transactions Control)

- ▶ MySQL 只有 InnoDB Type 才支援交易處理
- ▶ MySQL 預設的DML交易控制
 - 自動確認(AUTO COMMIT=1)
- ▶ MySQL 的DML交易控制方式
 - 起始一個交易：
 - SET AUTOCOMMIT = 0 - 改變預設交易模式
 - 使用 BEGIN | START TRANSACTION - 開啟單一交易
 - 結束一個交易：
 - COMMIT/ROLLBACK
 - Auto-Commit Command
 - DDL, DCL
 - SET AUTOCOMMIT

資料在交易中的狀態

▶ 交易進行中

- DML會產生row locking(列的鎖定)
- 可以檢視資料的改變
- 可做邏輯交易單位(savepoint 儲存點)的測試
- 其他sessions並無法看到資料的改變(讀取舊值)
 - Row Locking 並不會影響到其他的資料查詢

確認交易 (COMMIT) 後的資料狀態

▶ 確認交易：COMMIT

- 將資料儲存到資料庫中
- 解除交易過程中的 Locking
- 刪除設定的儲存點(savepoint)
- 其他連線(sessions)的使用者將可以看到變更後的資料
- 無法再復原(Rollback)資料
- 立即結束此交易

放棄交易(ROLLBACK)後的資料狀態

▶ 放棄交易:ROLLBACK

- 取消交易中資料的變更
- 解除交易過程中的 Locking
- 刪除設定的儲存點(savepoint)
- 其他連線(sessions)的使用者立即可對資料做操作(DML)
- 立即結束此交易

交易控制(Controlling Transactions)

▶ 設定交易區塊(TRANSACTION BLOCK)

```
START TRANSACTION | BEGIN;  
    dml_statement...;  
{COMMIT | ROLLBACK};
```

◦ 適合單次交易的處理

```
START TRANSACTION;  
    INSERT INTO DEPT VALUES (70, '財務部', '桃園');  
    SELECT * FROM DEPT;  
ROLLBACK;  
    SELECT * FROM DEPT;
```

交易控制(Controlling Transactions)

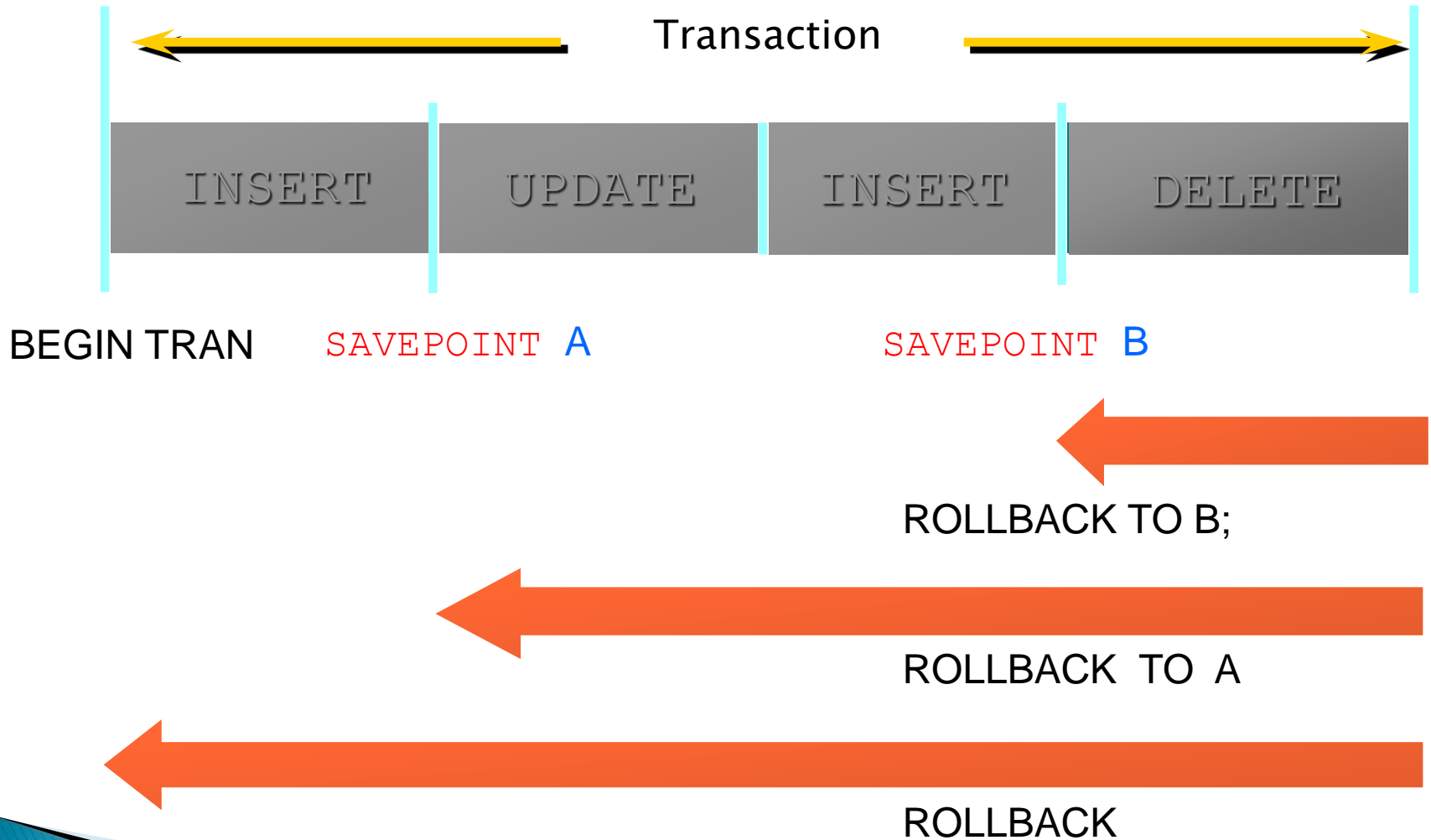
▶ 取消自動確認(DISABLE AUTOCOMMIT)

```
SET AUTOCOMMIT = {0|1}; /*預設為:1, 自動確認*/  
SELECT @@AUTOCOMMIT;    /*查詢目前設定值 */
```

◦ 可執行一連串的交易

```
SET AUTOCOMMIT = 0;          /* 安全交易控制開始 */  
INSERT INTO TX VALUES (NULL, NOW()); /* 交易開始 */  
SELECT * FROM TX;  
ROLLBACK;                    /* 交易倒回 */  
SELECT * FROM TX;  
/* ...仍可繼續其他交易的進行 */  
  
SET AUTOCOMMIT = 1; /* 安全交易控制結束 */
```

交易控制(Controlling Transactions)

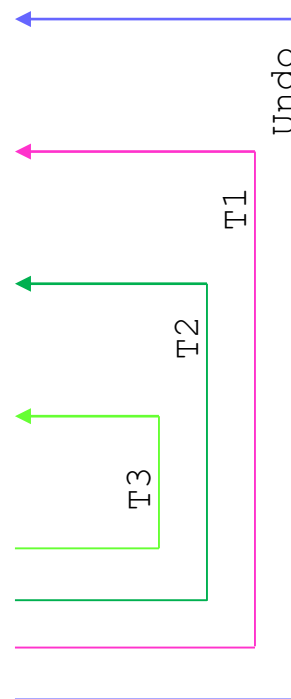


交易儲存點(SAVEPOINT)

▶ 交易儲存點

- 在交易中,可以有很多個statement,用邏輯的交易階段控制有助於交易的進行,儲存點即是實作的方法。
- 儲存點的名稱有分大小寫

```
Start transaction;  
  dml_statement;  
  ...  
  SAVEPOINT TX1;  
  dml_statement;  
  ...  
  SAVEPOINT TX2;  
  dml_statement;  
  ...  
  SAVEPOINT TX3;  
  dml_statement;  
  ...  
  ROLLBACK TO TX3;  
  ROLLBACK TO TX2;  
  ROLLBACK TO TX1;  
  ROLLBACK;
```



/* 交易控制開始 */

/* 邏輯交易階段T1 */

/* 邏輯交易階段T2 */

/* 邏輯交易階段T3 */

/* 交易控制結束 */

作業練習

1. 將下列的資料新增至MY_EMPLOYEE 資料表中, 不要列舉欄位.

1	Patel	Ralph	rpatel	795
---	-------	-------	--------	-----

2. 使用列舉欄位方式, 將下列的資料新增至 MY_EMPLOYEE資料表中.

2	Dancs	Betty	bdancs	860
---	-------	-------	--------	-----

3. 將下列的資料新增至 MY_EMPLOYEE資料表中.

3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750

4. 將員工編號為3的名字 (last name) 更改為 Drexler

5. 將薪資低於900元的所有員工薪資調整為1000元

6. 確認你所做的資料更新已更改到資料庫中

7. 刪除 Betty Dancs 的資料

8. 啟動一個資料庫交易

將所有員工的薪資調升10%

設定一個交易儲存點

刪除所有MY_EMPLOYEE資料表中的資料

確認資料已被你刪光了

放棄刪除資料的動作

確認交易