Khoa Công Nghệ Thông Tin
Trường Đại Học Cần Thơ

Những hỗ trợ tiên tiến khác của SQL

PostgreSQL

Đỗ Thanh Nghị
dtnghi@cit.ctu.edu.vn

Cần Thơ
24-04-2005

Nội dung

- Kế thừa
- Định nghĩa kiểu
- Định nghĩa hàm
- Định nghĩa ràng buộc dữ liệu
- Định nghĩa trigger
- Tạo View
- Truy cập cạnh tranh

- **Kế thừa**
- Định nghĩa kiểu
- Định nghĩa hàm
- Định nghĩa ràng buộc dữ liệu
- Định nghĩa trigger
- Tạo View
- Truy cập cạnh tranh

Định nghĩa kế thừa

- CREATE TABLE
  - Hỗ trợ cho định nghĩa kế thừa (INHERITES)
  - Phép truy vấn cũng làm việc trên dữ liệu thừa kế
  - SELECT (ONLY)

Ví dụ 1

```
mydb=# CREATE TABLE capitals (
mydb(#          name text,
mydb(#          population integer,
mydb(#          altitude integer,
mydb(#          state char(2));
CREATE TABLE

mydb=# CREATE TABLE cities (
mydb(#          name text,
mydb(#          population integer,
mydb(#          altitude integer);
CREATE TABLE
```

Ví dụ 2

```
mydb=# SELECT name, population, altitude
mydb-# FROM cities;
    name    | population | altitude
------------+------------+----------
 city one   |       1023 |      150
 city two   |      19723 |     1200
 city three |       5043 |      170
 city four  |       1013 |      100
(4 rows)

mydb=# SELECT name, population, altitude
mydb-# FROM capitals;
 name | population | altitude
------+------------+----------
 toto |       1072 |      160
 titi |       1972 |      175
 tutu |        972 |      115
(3 rows)
```

Ví dụ 3

```
mydb=# SELECT name, population, altitude
mydb-# FROM cities
mydb-# UNION
mydb-# SELECT name, population, altitude
mydb-# FROM capitals;
    name     | population | altitude
-------------+------------+----------
 city four   |       1013 |      100
 city one    |       1023 |      150
 city three  |       5043 |      170
 city two    |      19723 |     1200
 titi        |       1972 |      175
 toto        |       1072 |      160
 tutu        |        972 |      115
(7 rows)
```

Ví dụ 4 với kế thừa

```
mydb=# CREATE TABLE cities (
mydb(#          name text,
mydb(#          population integer,
mydb(#          altitude integer);
CREATE TABLE

mydb=# CREATE TABLE capitals (
mydb(#          state char(2))
mydb-#          INHERITS (cities);
CREATE TABLE
```

Ví dụ 5 với kế thừa

```
mydb=# SELECT *
mydb-# FROM cities;
    name    | population | altitude
------------+------------+----------
 city one   |       1023 |      150
 city two   |      19723 |     1200
 city three |       5043 |      170
 city four  |       1013 |      100
 toto       |       1072 |      160
 titi       |       1972 |      175
 tutu       |        972 |      115
(7 rows)
```

Ví dụ 6 với kế thừa

```
mydb=# SELECT *
mydb-# FROM ONLY cities;
    name    | population | altitude
------------+------------+----------
 city one   |       1023 |      150
 city two   |      19723 |     1200
 city three |       5043 |      170
 city four  |       1013 |      100
(4 rows)

mydb=# SELECT *
mydb-# FROM capitals;
 name | population | altitude | state
------+------------+----------+-------
 toto |       1072 |      160 | TO
 titi |       1972 |      175 | TI
 tutu |        972 |      115 | TU
(3 rows)
```

- Kế thừa
- **Định nghĩa kiểu**
- Định nghĩa hàm
- Định nghĩa ràng buộc dữ liệu
- Định nghĩa trigger
- Tạo View
- Truy cập cạnh tranh

Định nghĩa kiểu

- Cú pháp:

```
CREATE TYPE typename (
INPUT = input_function,
OUTPUT = output_function,
INTERNALLENGTH = { internallength | VARIABLE }
[, DEFAULT = default ]
[, ELEMENT = element ]
[, DELIMITER = delimiter ]
[, PASSEDBYVALUE ]
[, ALIGNMENT = alignment ]
[, STORAGE = storage ])

CREATE TYPE typename AS
(column_name data_type [, ... ])
```

Ví dụ 7

```
mydb=# CREATE TYPE inventory_item AS (
mydb(#     name            text,
mydb(#     supplier_id     integer,
mydb(#     price           numeric );
CREATE TYPE

mydb=# CREATE TABLE on_hand (
mydb(#     item      inventory_item,
mydb(#     count     integer );
CREATE TABLE

mydb=# INSERT INTO on_hand
mydb-# VALUES (ROW('fuzzy dice', 42, 1.99), 1000);
INSERT 17512 1

mydb=# UPDATE on_hand
mydb-# SET item.name = 'funny';
UPDATE 1
```

Ví dụ 7 (cont.)

```
mydb=# select * from on_hand;
          item          | count
------------------------+-------
 ("fuzzy dice",42,1.99) |  1000
(1 row)


mydb=# select item from on_hand;
          item
------------------------
 ("fuzzy dice",42,1.99)
(1 row)


mydb=# select item.price from on_hand;
ERROR:  missing FROM-clause entry for table "item"
LINE 1: select item.price from on_hand;
                ^
mydb=#
mydb=# select (item).price from on_hand;
 price
-------
  1.99
(1 row)
```

- Kế thừa
- Định nghĩa kiểu
- **Định nghĩa hàm**
- Định nghĩa ràng buộc dữ liệu
- Định nghĩa trigger
- Tạo View
- Truy cập cạnh tranh

Định nghĩa hàm

- Cú pháp:

CREATE FUNCTION name ([ftype [, ...]])
RETURNS rtype
AS definition
LANGUAGE 'langname ' [WITH (attribute [, ...])]

CREATE FUNCTION name ([ftype [, ...]])
RETURNS rtype
AS obj_file , link_symbol
LANGUAGE 'C' [WITH (attribute [, ...])]

Ví dụ 8

```
mydb=# SELECT status
mydb-# FROM s;
 status
--------
     20
     10
     30
(3 rows)

mydb=# CREATE FUNCTION minus_one(integer)
mydb-# RETURNS integer AS 'SELECT $1 - 1 AS RESULT;'
mydb-# LANGUAGE 'sql';
CREATE FUNCTION

mydb=# SELECT minus_one(status)
mydb-# FROM s;
 minus_one
-----------
        19
         9
        29
(3 rows)
```

Ví dụ 9

```
mydb=# CREATE FUNCTION add_one(i integer)
mydb-# RETURNS integer AS
mydb-# '
mydb'# BEGIN
mydb'#  RETURN i + 1;
mydb'# END;
mydb'# ' LANGUAGE 'plpgsql';
CREATE FUNCTION

mydb=# SELECT add_one(status)
mydb-# FROM s;
 add_one
---------
      21
      11
      31
(3 rows)
```

Ví dụ 10

```
bash-2.05b$ more inc.c

#include <pgsql/server/postgres.h>
#include <fmgr.h>
PG_MODULE_MAGIC;

int     increment(int arg);

/* By Value */

int     increment(int arg) {
        return arg + 1;
}


bash-2.05b$ gcc -c -fpic inc.c
bash-2.05b$ gcc -shared inc.o -o libinc.so
bash-2.05b$ ls lib*
libinc.so
```

Ví dụ 11

```
mydb=# CREATE FUNCTION increment(integer)
mydb-# RETURNS integer AS
mydb-# '/var/lib/pgsql/cde/libinc.so'
mydb-# LANGUAGE 'c';
CREATE FUNCTION

mydb=# SELECT increment(status)
mydb-# FROM s;
 increment
-----------
        21
        11
        31
(3 rows)
```

Ví dụ 11(cont.)

```
mydb=# CREATE FUNCTION price_extension(inventory_item, integer)
mydb-# RETURNS numeric
mydb-# AS 'SELECT $1.price * $2'
mydb-# LANGUAGE SQL;
CREATE FUNCTION

mydb=# SELECT price_extension(item, 10)
mydb-# FROM on_hand;
 price_extension
-----------------
           19.90
(1 row)
```

- Kế thừa
- Định nghĩa kiểu
- Định nghĩa hàm
- **Định nghĩa ràng buộc dữ liệu**
- Định nghĩa trigger
- Tạo View
- Truy cập cạnh tranh

Định nghĩa ràng buộc dữ liệu

- CREATE TABLE hỗ trợ cho định nghĩa những ràng buộc:
  - **NOT NULL**
  - **UNIQUE**
  - **CHECK**
  - **PRIMARY KEY (UNIQUE và NOT NULL)**
  - **REFERENCES (Khóa ngoài)**

Ví dụ 12

```
test=# CREATE TABLE cities (
test(#     nc smallint PRIMARY KEY,
test(#     name varchar(30),
test(#     pop bigint NOT NULL,
test(#     nb_museums smallint DEFAULT 0);

test=# CREATE TABLE hotels (
test(#     nh integer PRIMARY KEY,
test(#     nc smallint REFERENCES cities,
test(#     name varchar(30),
test(#     category varchar(12)
test(#     CHECK (category IN ('modeste','confortable','luxe')));
```

Ví dụ 13

```
test=# CREATE TABLE tourists (
test(#     nt integer PRIMARY KEY,
test(#     name varchar(20),
test(#     last_name varchar(20),
test(#     age smallint CHECK (age > 0),
test(#     type varchar(14)
test(#     CHECK (type IN ('sportif','intellectuel','ecologiste')));

test=# CREATE TABLE museums (
test(#     nm integer,
test(#     nc smallint REFERENCES cities,
test(#     name varchar(30) NOT NULL,
test(#     PRIMARY KEY (nm,nc));
```

Ví dụ 14

```
test=# CREATE TABLE bookings (
test(#     nt integer REFERENCES tourists,
test(#     nh integer REFERENCES hotels,
test(#     begin_date date NOT NULL,
test(#     nb_days smallint NOT NULL,
test(#     PRIMARY KEY (nt,nh,begin_date));

test=# CREATE TABLE ut (
test(#     id integer PRIMARY KEY,
test(#     acc_no integer UNIQUE);
```

Ví dụ 15

```
test=# SELECT *
test-# FROM hotels;
 nh | nc |   name     |  category
----+----+------------+-------------
  1 |  1 | Plaza      | confortable
  2 |  2 | Royal      | luxe
  3 |  3 | Onmi       | confortable
  4 |  4 | New World  | luxe
  5 |  1 | Sami       | modeste
(5 rows)

test=# SELECT *
test-# FROM tourists;
 nt |  name  |   last_name     | age |     type
----+--------+-----------------+-----+--------------
  1 | Do     | Thanh Nghi      |  31 | sportif
  2 | Le     | Thi Huyen       |  30 | intellectuel
  3 | Do     | Hiep Thuan      |  25 | ecologiste
  4 | Do     | Thi Bich Hanh   |  28 | sportif
  5 | Do     | Thi Thanh Dung  |  33 | intellectuel
  6 | Do     | Thi Kim Phuong  |  35 | sportif
  7 | Nguyen | Thanh Chuong    |  35 | sportif
(7 rows)
```

Ví dụ 16

```
test=# SELECT *
test-# FROM museums;
 nm | nc |   name
----+----+----------
  1 |  1 | CULTURE
  2 |  1 | ROI
  3 |  1 | CHARLIE
  1 |  2 | LOUVRE
  1 |  4 | HISTOIRE
  2 |  4 | SIECLE
(6 rows)
```

```
test=# SELECT *
test-# FROM bookings;
 nt | nh | begin_date | nb_days
----+----+------------+--------
  1 |  4 | 1999-04-30 |       2
  2 |  1 | 1999-05-03 |       5
  2 |  2 | 1999-04-03 |       2
  2 |  4 | 1999-04-10 |       4
  3 |  2 | 1999-04-30 |       2
  6 |  2 | 1999-05-10 |       3
(6 rows)
```

```
test=# SELECT *
test-# FROM cities;
 nc |   name    |  pop   | nb_museums
----+-----------+--------+------------
  3 | Laval     |   9800 |          0
  1 | Montreal  | 150000 |          3
  2 | Hull      |    900 |          1
  4 | McGill    |   5000 |          2
(4 rows)
```

Kiểm tra ràng buộc: primary key, not null, references


```
test=# INSERT INTO cities(nc, name, pop) VALUES (3, 'Waterloo',
    150000);
ERROR:  duplicate key violates unique constraint "cities_pkey"

test=# INSERT INTO museums VALUES (2, 2, NULL);
ERROR:  null value in column "name" violates not-null constraint

test=# INSERT INTO museums VALUES (1, 10, 'CULTURE');
ERROR:  insert or update on table "museums" violates foreign key
    constraint "museums_nc_fkey"
DETAIL:  Key (nc)=(10) is not present in table "cities".
```

Kiểm tra ràng buộc: check, unique


```
test=# INSERT INTO tourists VALUES (8, 'Dinh', 'Gia Bao', 34, 'foo');
ERROR:  new row for relation "tourists" violates check constraint
    "tourists_type_check"

test=# INSERT INTO tourists VALUES (8, 'Dinh', 'Gia Bao', 0,
    'sportif');
ERROR:  new row for relation "tourists" violates check constraint
    "tourists_age_check"

test=# INSERT INTO ut VALUES (1,1072);

test=# INSERT INTO ut VALUES (2,1072);
ERROR:  duplicate key violates unique constraint "ut_acc_no_key"
```

- Kế thừa
- Định nghĩa kiểu
- Định nghĩa hàm
- Định nghĩa ràng buộc dữ liệu
- **Định nghĩa trigger**
- Tạo View
- Truy cập cạnh tranh

Định nghĩa Trigger

- Cú pháp:

CREATE TRIGGER name { BEFORE | AFTER } { event [OR ...] }

ON table

FOR EACH { ROW | STATEMENT }

EXECUTE PROCEDURE func (arguments)

Ví dụ 17

- Tạo trigger để đảm bảo khi cập nhật dữ liệu trong bảng museums, cột nb_museums trong bảng cities là tổng số museum có trong thành phố

Ví dụ 18

```
test=# CREATE FUNCTION insert_museums() RETURNS TRIGGER AS
test-# '
test'# BEGIN
test'#   UPDATE cities SET nb_museums = nb_museums + 1 WHERE nc = NEW.nc;
test'#   RETURN NEW;
test'# END;
test'# ' LANGUAGE 'plpgsql';
CREATE FUNCTION

test=# CREATE TRIGGER trigger_insert_museums BEFORE INSERT ON museums
test-# FOR EACH ROW EXECUTE PROCEDURE insert_museums();
CREATE TRIGGER

test=# CREATE FUNCTION del_museums() RETURNS TRIGGER AS
test-# '
test'# BEGIN
test'#   UPDATE cities SET nb_museums = nb_museums - 1 WHERE nc = OLD.nc;
test'#   RETURN NEW;
test'# END;
test'# ' LANGUAGE 'plpgsql';
CREATE FUNCTION

test=# CREATE TRIGGER trigger_del_museums BEFORE DELETE ON museums
test-# FOR EACH ROW EXECUTE PROCEDURE del_museums();
CREATE TRIGGER
```

Ví dụ 19

```
test=# CREATE FUNCTION update_museums() RETURNS TRIGGER AS
test-# '
test'# BEGIN
test'#   IF OLD.nc <> NEW.nc THEN
test'#     UPDATE cities SET nb_museums = nb_museums - 1 WHERE nc = OLD.nc;
test'#     UPDATE cities SET nb_museums = nb_museums + 1 WHERE nc = NEW.nc;
test'#   END IF;
test'#   RETURN NEW;
test'# END;
test'# ' LANGUAGE 'plpgsql';
CREATE FUNCTION

test=# CREATE TRIGGER trigger_update_museums BEFORE UPDATE ON museums
test-# FOR EACH ROW EXECUTE PROCEDURE update_museums();
CREATE TRIGGER
```

Ví dụ 20

- Tạo trigger để đảm bảo khi cập nhật dữ liệu trong bảng bookings, một người du lịch thuộc típ intellectuel chỉ ở trong khách sạn của một thành phố có ít nhất 1 bảo tàng

Ví dụ 21

```
test=# CREATE FUNCTION insert_update_bookings1() RETURNS TRIGGER AS
test-# '
test'# DECLARE
test'#   nbm   INTEGER;
test'# BEGIN
test'#   SELECT cities.nb_museums INTO nbm
test'#   FROM cities, hotels, tourists
test'#   WHERE (NEW.nt = tourists.nt)
test'#   AND (tourists.type = ''intellectuel'')
test'#   AND (NEW.nh = hotels.nh)
test'#   AND (hotels.nc = cities.nc);
test'#
test'#   IF (nbm < 1) THEN
test'#     RAISE EXCEPTION ''A tourist of the intellectuel type ....'';
test'#   END IF;
test'#   RETURN NEW;
test'# END;
test'# ' LANGUAGE 'plpgsql';
CREATE FUNCTION

test=# CREATE TRIGGER trigger_insert_update_bookings1
test-# BEFORE INSERT OR UPDATE ON bookings
test-# FOR EACH ROW EXECUTE PROCEDURE insert_update_bookings1();
CREATE TRIGGER
```

Ví dụ 22

- Tạo trigger để đảm bảo khi cập nhật dữ liệu trong bảng bookings, một người du lịch thuộc típ ecologiste chỉ ở trong khách sạn của một thành phố có dân số ít hơn 1000

Ví dụ 23

```
test=# CREATE FUNCTION insert_update_bookings2() RETURNS TRIGGER AS
test-# '
test'# DECLARE
test'#   nbp   INTEGER;
test'# BEGIN
test'#   SELECT cities.pop INTO nbp
test'#   FROM cities, hotels, tourists
test'#   WHERE (NEW.nt = tourists.nt)
test'#   AND (tourists.type =''ecologiste'')
test'#   AND (NEW.nh = hotels.nh)
test'#   AND (hotels.nc = cities.nc);
test'#
test'#   IF (nbp >= 1000) THEN
test'#     RAISE EXCEPTION ''A tourist of the ecologiste type ...'';
test'#   END IF;
test'#   RETURN NEW;
test'# END;
test'# ' LANGUAGE 'plpgsql';
CREATE FUNCTION

test=# CREATE TRIGGER trigger_insert_update_bookings2
test-# BEFORE INSERT OR UPDATE ON bookings
test-# FOR EACH ROW EXECUTE PROCEDURE insert_update_bookings2();
CREATE TRIGGER
```

Kiểm tra những triggers: trigger_insert_update_bookings1 & trigger_insert_update_bookings2

test=# INSERT INTO bookings VALUES (3, 3, '1999-04-30', 2);

ERROR: A tourist of the ecologiste type only books the hotel in a city having less 1000 populations

test=# INSERT INTO bookings VALUES (2, 3, '1999-04-30', 2);

ERROR: A tourist of the intellectuel type only books the hotel in a city having at least one museum

- Kế thừa
- Định nghĩa kiểu
- Định nghĩa hàm
- Định nghĩa ràng buộc dữ liệu
- Định nghĩa trigger
- **Tạo View**
- Truy cập cạnh tranh

Định nghĩa View

- Cú pháp:

CREATE VIEW view_name AS SELECT query

- Từ ver. 9.2 trở về trước postgres không cho phép insert, update, delete trên view. Cần kết hợp với CREATE RULE để thực hiện insert, update, delete

- Nhưng từ ver. 9.3 postgres cho phép insert, update, delete trên view đơn giản (không có mệnh đề WITH, DISTINCT, GROUP BY, HAVING, LIMIT, OFFSET, UNION, INTERSECT, EXCEPT, etc.)

Ví dụ


```
mydb=# CREATE TABLE realtable (
mydb(#         col integer);
CREATE TABLE

mydb=# CREATE VIEW view_realtable AS
mydb-# SELECT *
mydb-# FROM realtable;
CREATE VIEW

mydb=# INSERT INTO realtable VALUES (1);
INSERT 17527 1

mydb=# INSERT INTO view_realtable VALUES (2);
ERROR:  cannot insert into a view
HINT:  You need an unconditional ON INSERT DO INSTEAD rule.
ERROR:  cannot insert into a view
```

Ví dụ

```
mydb=# SELECT *
mydb-# FROM realtable;
 col
-----
   1
(1 row)

mydb=# SELECT *
mydb-# FROM view_realtable;
 col
-----
   1
(1 row)
```

Ví dụ

```
mydb=# CREATE RULE view_realtable_insert AS
mydb-# ON INSERT TO view_realtable
mydb-# DO INSTEAD
mydb-#         INSERT INTO realtable VALUES (new.col);
CREATE RULE

mydb=# INSERT INTO view_realtable VALUES (3);
INSERT 17529 1

mydb=# SELECT *
mydb-# FROM realtable;
 col
-----
   1
   3
(2 rows)
```

Ví dụ

```
mydb=# CREATE RULE view_realtable_update AS
mydb-# ON UPDATE TO view_realtable
mydb-# DO INSTEAD
mydb-#         UPDATE realtable
mydb-#         SET col = new.col
mydb-#         WHERE col = old.col;
CREATE RULE

mydb=# CREATE RULE view_realtable_delete AS
mydb-# ON DELETE TO view_realtable
mydb-# DO INSTEAD
mydb-#         DELETE FROM realtable
mydb-#         WHERE col = old.col;
CREATE RULE
```

- Kế thừa
- Định nghĩa kiểu
- Định nghĩa hàm
- Định nghĩa ràng buộc dữ liệu
- Định nghĩa trigger
- Tạo View
- **Truy cập cạnh tranh**

Truy cập cạnh tranh

- Nhiều người dùng chia sẻ cơ sở dữ liệu
  - Cập nhật, truy vấn đồng thời
  - Làm thế nào để đảm bảo được tính nhất quán của dữ liệu từ phía các người dùng ?
  - Khái niệm giao dịch (**TRANSACTION**)
  - **ACID (Atomic, Consistent, Isolated, Durable)**
  - **MVCC (MultiVersion Concurrency Control)**
  - **BEGIN, ABORD, END, COMMIT, ROLLBACK, SAVEPOINT, CHECKPOINT, SET TRANSACTION, START TRANSACTION, SET CONSTRAINTS, LOCK**

Truy cập cạnh tranh

- Tính nguyên tử
  - Giao dịch kết thúc làm « **all-or-nothing** » trên cơ sở dữ liệu
  - Giao dịch bắt đầu bằng **BEGIN**
  - Lệnh trong giao dịch chỉ được thực hiện khi gặp « **COMMIT** »
  - Lệnh trong giao dịch sẽ bị hủy bỏ khi gặp « **ROLLBACK** »
  - Mọi câu SQL đều nên thực thi trong một giao dịch

Truy cập cạnh tranh

- Tính nguyên tử
  - Một giao dịch nhóm nhiều phép toán thành phép toán nguyên tử
  - Kết quả của giao dịch là « **all-or-nothing** »

One transaction

```
BEGIN;
UPDATE accounts SET balance = balance - 100.00
  WHERE name = 'Alice';
UPDATE branches SET balance = balance - 100.00
  WHERE name = (SELECT branch_name FROM accounts
  WHERE name = 'Alice');
UPDATE accounts SET balance = balance + 100.00
  WHERE name = 'Bob';
UPDATE branches SET balance = balance + 100.00
  WHERE name = (SELECT branch_name FROM accounts
  WHERE name = 'Bob');
COMMIT;
```

Truy cập cạnh tranh

- Tính cô lập
  - Có 4 mức độ cô lập: **SERIALIZABLE, REPEATABLE READ, READ COMMITTED, READ UNCOMMITTED**
  - 3 hiện tượng thường gặp: **dirty read, nonrepeatable read, phantom read**

Isolation Level	Dirty Read	Nonrepeatable Read	Phantom Read
Read uncommitted	Possible	Possible	Possible
Read committed	Not possible	Possible	Possible
Repeatable read	Not possible	Not possible	Possible
Serializable	Not possible	Not possible	Not possible

Truy cập cạnh tranh

- Tính cô lập
  - Mặc định PostgreSQL sử dụng **READ COMMITTED**

User: bruce	Time	User:sheila
BEGIN TRANSACTION	T1	BEGIN TRANSACTION
UPDATE customers SET balance = balance - 3 WHERE customer_id = 2;	T2	
	T3	SELECT SUM(balance) FROM customers;
	T4	COMMIT TRANSACTION;
ROLLBACK TRANSACTION;	T5	

Truy cập cạnh tranh

- Tính cô lập: **READ COMMITTED => nonrepeatable read**

User: bruce	Time	User: sheila
`BEGIN TRANSACTION;`	T1	`BEGIN TRANSACTION;`

	T2	`SELECT balance`
		`FROM customers`
		`WHERE`
		`customer_id = 2;`

	T3	

`UPDATE customers`
`SET balance = 20`
`WHERE`
`customer_id = 2;`		

`COMMIT TRANSACTION;`	T4	

	T5	`SELECT balance`
		`FROM customers`
		`WHERE`
		`customer_id = 2;`

	T6	`COMMIT TRANSACTION;`

Truy cập cạnh tranh

- Tính cô lập
  - Sử dụng **SERIALIZABLE**
  - Cho phép chỉ thấy dữ liệu đã được committed trước thời điểm bắt đầu của giao dịch
  - Cho phép thấy sự thay đổi bởi các lệnh trước đó trong cùng giao dịch mặc dù chưa được committed

Truy cập cạnh tranh

- Công cụ **LOCK**
  - Sử dụng **LOCK** tường minh hay **SELECT … FOR UPDATE**

```
BEGIN WORK;

SELECT *
FROM lock_test
WHERE name = 'James'
FOR UPDATE;
-- the SELECTed row is locked

UPDATE lock test
SET name = 'Jim'
WHERE name = 'James';

COMMIT WORK;
```

Savepoints

- Đánh dấu bên trong một giao dịch
  - Giúp cho việc hủy bỏ những phép toán từ vị trí Savepoint đến lệnh Rollback trong một giao dịch
  - Một giao dịch có thể có nhiều Savepoints

One transaction

```
BEGIN;
UPDATE accounts SET balance = balance - 100.00 WHERE name = 'Alice';
SAVEPOINT my_savepoint;
UPDATE accounts SET balance = balance + 100.00 WHERE name = 'Bob';
```

- Oops ... use the account of Charlie instead!

```
ROLLBACK TO my_savepoint;
UPDATE accounts SET balance = balance + 100.00 WHERE name = 'Charlie';
COMMIT;
```

Cám ơn !