CSCI235 Database Systems

Transaction Processing in PostgreSQL DBMS

Dr Janusz R. Getta

School of Computing and Information Technology -
University of Wollongong

Transaction Processing in PostgreSQL DBMS

Outline

Transaction scope

A transaction starts from `BEGIN` or `START TRANSACTION` statement and it ends with `COMMIT` or `ROLLBACK` statement

An SQL statements outside of an explicit transaction automatically become a single-statement transaction

PostgreSQL supports transactional DDL

It means that all DDL statemehts except `CREATE/DROP/ALTER DATABASE`, `TABLESPACE`, `CLUSTER` are reversible

If an error occurs inside a transaction then it is automatically rolledback

However, still a statement to complete the current transaction (`COMMIT`, `ROLLBACK`, `ABORT`) is required

Transaction Processing in PostgreSQL DBMS

Outline

Isolation levels in PostgreSQL

At `READ COMMITED` isolation level a transaction may exhibit:

- non-repeatable read phenomenon,
- phantom phenomenon

At `REPEATABLE READ` isolation level (SI) a transaction may exhibit:

- other phonemena (write skew)

At `SERIALIZABLE` isolation level (SSI)a transaction may exhibit:

- none of the phenomena

Isolation levels versus phenomena

Level	Dirty Read	Nonrepeatable Read	Phantom	Other (Write skew)
READ COMMITTED	Not possible	Possible	Possible	Possible
REPEATABLE READ	Not possible	Not possible	Not possible	Possible
SERIALIZABLE	Not possible	Not possible	Not possible	Not possible

Transaction Processing in PostgreSQL DBMS

Outline

READ COMMITTED isolation level

We consider a database created through processing of the following `CREATE TABLE` statements

Relational table DEPARTMENT

```
CREATE TABLE DEPARTMENT(
 name               VARCHAR(50)        NOT NULL,
 code               CHAR(5)            NOT NULL,
 total_staff_number DECIMAL(2)         NOT NULL,
 chair              VARCHAR(50)            NULL,
 budget             DECIMAL(9,1)       NOT NULL,
  CONSTRAINT dept_pkey PRIMARY KEY(name),
  CONSTRAINT dept_ckey1 UNIQUE(code),
  CONSTRAINT dept_ckey2 UNIQUE(chair),
  CONSTRAINT dept_check1 CHECK (total_staff_number BETWEEN 1 AND 50) );
```

Relational table COURSE

```
CREATE TABLE COURSE(
 cnum               CHAR(7)           NOT NULL,
 title              VARCHAR(200)      NOT NULL,
 credits            DECIMAL(2)        NOT NULL,
 offered_by         VARCHAR(50)           NULL,
  CONSTRAINT course_pkey PRIMARY KEY(cnum),
  CONSTRAINT course_check1 CHECK (credits IN (6, 12)),
  CONSTRAINT course_fkey1 FOREIGN KEY(offered_by)
                          REFERENCES DEPARTMENT(name) ON DELETE CASCADE );
```

READ COMMITTED isolation level

Dirty read is not possible

Select title

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SELECT title FROM COURSE WHERE cnum='CSCI111';
 title
-------
 C++
```

Update title

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
UPDATE COURSE SET title = 'Java' WHERE cnum='CSCI111';
```

Select title

```
SELECT title FROM COURSE WHERE cnum='CSCI111';
 title
-------
C++
```

Commit modification

```
COMMIT;
```

READ COMMITTED isolation level

Lost update is not possible

Select budget

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SELECT budget FROM DEPARTMENT
WHERE name = 'Computer Science';
  budget
-----------
 223426.9
```

Update budget

```
UPDATE DEPARTMENT SET budget = budget + 100
WHERE name = 'Computer Science';
```

Update budget

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
UPDATE DEPARTMENT SET budget = budget + 200
WHERE name = 'Computer Science';
WAIT !!!
```

Continued on the next slide ...

READ COMMITTED isolation level

Continuation from the previous slide ...

Commit modification

```
COMMIT;
```

A lock is released, modification is done and committed

```
                                COMMIT;
```

Select budget

```
SELECT budget from DEPARTMENT
WHERE name = 'Computer Science';
  budget
__________
 223726.9
```

READ COMMITTED isolation level

Non-repeatable read phenomenon is possible

Select budget

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SELECT budget FROM DEPARTMENT
WHERE name = 'Computer Science';
  budget
-----------
 223426.9
```

Update budget

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
UPDATE DEPARTMENT SET budget = budget + 200
WHERE name = 'Computer Science';
```

Commit modification

```
COMMIT;
```

Select budget again

```
SELECT budget FROM DEPARTMENT
WHERE name = 'Computer Science';
  budget
-----------
 223626.9
```

READ COMMITTED isolation level

Phantom phenomenon is possible

Count the rows

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SELECT count(*) FROM COURSE;
 count
-------
    15
```

Delete one row

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
DELETE FROM COURSE
WHERE offered_by = 'Mathematics';
```

Commit deletion

```
COMMIT;
```

Count the rows

```
SELECT count(*) FROM COURSE;
 count
-------
    14
```

READ COMMITTED isolation level

Corruption of a database is possible

We would like to set a budget of Computer Science department equal to 2*budget of Mathematics department

Displays the budgets

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SELECT name,budget FROM DEPARTMENT
WHERE name IN ('Computer Science','Mathematics');
       name        |  budget
-------------------+----------
 Computer Science | 223426.9
 Mathematics      | 133456.9
```

Set budget of Computer Science department = budget of Mathematics department

```
UPDATE DEPARTMENT SET budget =
       (SELECT budget FROM DEPARTMENT
        WHERE name = 'Mathematics')
WHERE name = 'Computer Science';
```

Continued on the next slide ...

READ COMMITTED isolation level

Continued from the previous slide ...

Update a budget of Mathematics department

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
UPDATE DEPARTMENT SET BUDGET = 1000
WHERE name = 'Mathematics';
```

Commit update

```
COMMIT;
```

Set budget of Computer Science department = 2*budget of Mathematics department

```
UPDATE DEPARTMENT SET budget =  budget +
       (SELECT budget FROM DEPARTMENT
        WHERE name = 'Mathematics')
WHERE name = 'Computer Science';
```

Displays the budgets again

```
SELECT name,budget FROM DEPARTMENT
WHERE name IN ('Computer Science','Mathematics');
       name        |  budget
-------------------+----------
 Mathematics       |   1000.0
 Computer Science | 134456.9
```

READ COMMITTED isolation level

Write skew is possible

We would like to remove one of the courses, either ENGG100 or ENGG103

Display ENGG courses

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SELECT cnum, title FROM COURSE WHERE cnum LIKE 'ENGG%';
  cnum   |               title
---------+-----------------------------------
 ENGG103 | Materials in Design
 ENGG100 | Engineering Computing and Analysis
 ENGG111 | C++
```

Delete a course ENGG103

```
DELETE FROM COURSE WHERE cnum = 'ENGG103';
```

Continued on the next slide ...

READ COMMITTED isolation level

Continued from the previous slide ...

Display ENGG courses

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SELECT cnum, title FROM COURSE WHERE cnum LIKE 'ENGG%';
  cnum   |                title
---------+-----------------------------------
 ENGG103 | Materials in Design
 ENGG100 | Engineering Computing and Analysis
 ENGG111 | C++
```

Delete a course ENGG100

```
DELETE FROM COURSE WHERE cnum = 'ENGG100';
```

Commit deletion

```
COMMIT;
```

Commit deletion

```
COMMIT;
```

DisplaysENGG courses

```
SELECT cnum, title FROM COURSE WHERE cnum LIKE 'ENGG%';
  cnum   |                title
---------+-----------------------------------
 ENGG111 | C++
```

Transaction Processing in PostgreSQL DBMS

Outline

REPEATABLE READ isolation level (SI)

Non-repeatable read phenomenon is not possible

Select budget

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
SELECT budget FROM DEPARTMENT
WHERE name = 'Computer Science';
  budget
-----------
 223426.9
```

Update budget

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
UPDATE DEPARTMENT SET budget = budget + 200
WHERE name = 'Computer Science';
```

Commit modification

```
COMMIT;
```

Select budget again

```
SELECT budget FROM DEPARTMENT
WHERE name = 'Computer Science';
  budget
-----------
 223426.9
```

REPEATABLE READ isolation level (SI)

Phantom phenomenon is not possible

Count the rows

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
SELECT count(*) FROM COURSE;
 count
-------
    15
```

Delete one row

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
DELETE FROM COURSE
WHERE offered_by = 'Mathematics';
```

Commit deletion

```
COMMIT;
```

Count the rows

```
SELECT count(*) FROM COURSE;
 count
-------
    15
```

REPEATABLE READ isolation level (SI)

Corruption of a database is not possible

We would like to set a budget of Computer Science department equal to 2*budget of Mathematics department

Displays the budgets

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
SELECT name,budget FROM DEPARTMENT
WHERE name IN ('Computer Science','Mathematics');
       name        |  budget
-------------------+----------
 Computer Science | 223426.9
 Mathematics      | 133456.9
```

Set budget of Computer Science department = budget of Mathematics department

```
UPDATE DEPARTMENT SET budget =
       (SELECT budget FROM DEPARTMENT
        WHERE name = 'Mathematics')
WHERE name = 'Computer Science';
```

Continued on the next slide ...

READ COMMITTED isolation level

Continued from the previous slide ...

Update a budget of Mathematics department

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
UPDATE DEPARTMENT SET BUDGET = 1000
WHERE name = 'Mathematics';
```

Commit update

```
COMMIT;
```

Set budget of Computer Science department = 2*budget of Mathematics department

```
UPDATE DEPARTMENT SET budget =  budget +
       (SELECT budget FROM DEPARTMENT
        WHERE name = 'Mathematics')
WHERE name = 'Computer Science';
```

Displays the budgets again

```
SELECT name,budget FROM DEPARTMENT
WHERE name IN ('Computer Science','Mathematics');
       name        |  budget
-------------------+----------
 Mathematics       |   1000.0
 Computer Science | 266913.8
```

REPEATABLE READ isolation level

Lost update is not possible (first committer wins)

Select budget

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
SELECT budget FROM DEPARTMENT
WHERE name = 'Computer Science';
  budget
-----------
 223426.9
```

Update budget

```
UPDATE DEPARTMENT SET budget = budget + 100
WHERE name = 'Computer Science';
```

Update budget

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
UPDATE DEPARTMENT SET budget = budget + 200
WHERE name = 'Computer Science';
WAIT !!!
```

Continued on the next slide ...

REPEATABLE READ isolation level

Continuation from the previous slide ...

Commit modification

```
COMMIT;
```

A transaction is automatically aborted and rolled back

```
ERROR:  could not serialize access due to concurrent update
```

Select budget

```
SELECT budget from DEPARTMENT
WHERE name = 'Computer Science';
  budget
----------
 223526.9
```

REPEATABLE READ isolation level

Write skew is possible

We would like to remove one of the courses, either ENGG100 or ENGG103

Display ENGG courses

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL REPATABLE READ;
SELECT cnum, title FROM COURSE WHERE cnum LIKE 'ENGG%';
  cnum   |               title
---------+-----------------------------------
 ENGG103 | Materials in Design
 ENGG100 | Engineering Computing and Analysis
 ENGG111 | C++
```

Delete a course ENGG103

```
DELETE FROM COURSE WHERE cnum = 'ENGG103';
```

Continued on the next slide ...

REPEATABLE READ isolation level

Continued from the previous slide ...

Display ENGG courses

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
SELECT cnum, title FROM COURSE WHERE cnum LIKE 'ENGG%';
  cnum   |               title
---------+-----------------------------------
 ENGG103 | Materials in Design
 ENGG100 | Engineering Computing and Analysis
 ENGG111 | C++
```

Delete a course ENGG100

```
DELETE FROM COURSE WHERE cnum = 'ENGG100';
```

Commit deletion

```
COMMIT;
```

Commit deletion

```
COMMIT;
```

DisplaysENGG courses

```
SELECT cnum, title FROM COURSE WHERE cnum LIKE 'ENGG%';
  cnum   |               title
---------+-----------------------------------
 ENGG111 | C++
```

Transaction Processing in PostgreSQL DBMS

Outline

SERIALIZABLE isolation level

Write skew is not possible

We would like to remove one of the courses, either ENGG100 or ENGG103

Display ENGG courses

```
BEGIN;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SELECT cnum, title FROM COURSE WHERE cnum LIKE 'ENGG%';
  cnum   |               title
---------+-----------------------------------
 ENGG103 | Materials in Design
 ENGG100 | Engineering Computing and Analysis
 ENGG111 | C++
```

Delete a course ENGG103

```
DELETE FROM COURSE WHERE cnum = 'ENGG103';
```

Continued on the next slide ...

SERIALIZABLE isolation level

Continued from the previous slide ...

```
                                                                    Display ENGG courses
                          BEGIN;
                          SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
                          SELECT cnum, title FROM COURSE WHERE cnum LIKE 'ENGG%';
                            cnum   |                title
                          ---------+-------------------------------------
                           ENGG103 | Materials in Design
                           ENGG100 | Engineering Computing and Analysis
                           ENGG111 | C++
```

```
                                                                  Delete a course ENGG100
                          DELETE FROM COURSE WHERE cnum = 'ENGG100';
```

```
                          COMMIT;                                                 Commit deletion
```

```
COMMIT;                                                                           Commit deletion
```

```
                                                          A transaction is aborted and rolled back
ERROR:  could not serialize access due to read/write dependencies among transactions
DETAIL:  Reason code: Canceled on identification as a pivot, during commit attempt.
```

References

PostgreSQL 3 Documentation, Chapter 3. Advanced Features, 3.4 Transactions

PostgreSQL 3 Documentation, SQL Commands, SET TRANSACTION

E. Rogov, PostgreSQL 14 Internals, Postgres Professional, 2023

D. R. K. Ports, K. Grittner, Serializable Snapshot Isolation in PostgreSQL, Proceedings of the VLDB Endowment, Vol. 5, No. 12, 2012