Exercise 3 Report

Gruppe 16

Anastasiia Rubanovych   Sebastian Funck

May 17, 2020

3.1 Isolation Levels and SQL

a)

- **How can you determine the currently set isolation level?**
  The default isolation level of a database is configured inside the `postgresql.conf`
  with the attribute `default_transaction_isolation`. It can be queried via `SHOW`
  `default_transaction_isolation`.

- **What is the default isolation level of PostgreSQL?**
  The default isolation level of PostgreSQL is `read committed`.

- **How can the isolation level be changed during a session in PostgreSQL?**
  `SET SESSION CHARACTERISTICS AS TRANSACTION { SERIALIZABLE | REPEATABLE`
  `READ | READ COMMITTED | READ UNCOMMITTED }`

b)

```
create table OPK (
  ID    int4 ,
  NAME varchar (64)
)
```

c)

```
insert into OPK (ID, NAME)        VALUES (1, 'shaggy'), (2, 'fred'),        (3, '
```

d)

- **Discuss what locks you would expect are held at this point (and before a commit) with `Read Committed`?**
  We would expect 1 table lock and 1 row lock to be hold before and 0 table locks and 0 row locks to be hold after the commit. `Cursor Stability (Read Committed)` never holds more than 1 row lock.

- **Discuss what locks you would expect are held at this point (and before a commit) with `Repeatable Read`?**
  We would expect 1 table lock to be hold before and 0 locks after the commit. `Repeatable Read` locks the complete table until commit.

3.2 Lock Conflicts

a)

- **What happens? What is the output of Connection 1?**
  The output of Connection 1 is

  ```
  ( 4 ,  ' scooby ' )
  ( 5 ,  ' daphne ' )
  ```

- **Compare the state before the transactions with the state after the transactions.**
  After the transactions, one new row has been added to the OPK table.

- **What can be observed if Connection 1 commits and execute its SQL command again?**
  The output of Connection 1 then becomes

  ```
  ( 4 ,  ' scooby ' )
  ( 5 ,  ' daphne ' )
  ( 6 ,  ' scrappy ' )
  ```

- **Can we observe a Canonical Synchronization Problem? If yes, explain which one and why it appears.**
  We cannot observe any canonical synchronization problems.

b)

- **What happens? What is the output of Connection 1?**
  The output of Connection 1 is

  ```
  ( 4 ,  ' scooby ' )
  ( 5 ,  ' daphne ' )
  ```

- **Compare the state before the transactions with the state after the transactions.**
  After the transactions, one new row has been added to the OPK table.

- **What can be observed if Connection 1 commits and execute its SQL command again?**
  The output of Connection 1 then becomes

$$(4, \text{'scooby'})$$
$$(5, \text{'daphne'})$$
$$(6, \text{'scrappy'})$$

- **Can we observe a Canonical Synchronization Problem? If yes, explain which one and why it appears.**
  We cannot observe any canonical synchronization problems.

c)

- **In this scenario Connection 2 has to wait until Connection 1 commits. Explain why.**
  The `Serializable` isolation level ensures a serial execution of transactions, therefor Connection 1 has to be finished before Connection 2 can continue.

- **Discuss, what lock can be potentially encountered on the table OPK? Which Connection do the locks belong to?**
  Connection 1 is likely to hold an R lock on the table.

d)

```
create table MPK (
        id int4 ,
        name varchar(64) ,
        CONSTRAINT mpk_pk PRIMARY KEY (id)
);

insert into MPK (ID, NAME)
        VALUES (1, 'shaggy'), (2, 'fred'),
        (3, 'velma'), (4, 'scooby'), (5, 'daphne');
```

e)

We observed the exact same behaviour as for *c)*.