*ЛР-7*

*Симаков Александр*

*А-16-20*

*Вариант : 29*

***Задание***

**Индивидуальные варианты заданий**

Номер варианта вычисляется как ((N - 1) mod 6) + 1 = 2, где N – номер студента в группе.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Вариант** | **RU MVCC** | **RC MVCC** | **RR MVCC** | **RC LOCK** | **RR LOCK** |
| 2 |  | + |  |  | + |

**1. Зависимость потерянного обновления**

|  |  |
| --- | --- |
| **T1** | **T2** |
| read(x) |  |
|  | write(x) |
| write(x) |  |

**RC  
MVCC**

|  |  |
| --- | --- |
| T1 | T2 |
| set transaction isolation level read committed; begin; select \* from players; |  |
|  | set transaction isolation level read committed;  begin;  update players set nickname = 'A' where player\_id = 31;  select \* from players;  Commit; |
| select \* from players;    update players set nickname = 'ola' where player\_id = 31; |  |

Вывод: потерянное обновление не допускается, т.к. при RC MVCC транзакция видит только те данные, которые были зафиксированы другой транзакцией.

**RR  
LOCK**

|  |  |
| --- | --- |
| T1 | T2 |
| set transaction isolation level repeatable read;  begin;  select \* from players LOCK IN SHARE MODE; |  |
|  | set transaction isolation level repeatable read;  begin;  update players set nickname = 'A' where player\_id = 31;  Running …. |
| update players set nickname = 'A' where player\_id = 31; |  |

Вывод: потерянное обновление не допускается, т.к. при RR LOCK вторая транзакция просто не может внести изменения в таблицу, она ожидает пока первая транзакция закончит работу.

**2. Зависимость грязного чтения**

|  |  |
| --- | --- |
| **T1** | **T2** |
|  | write(x) |
| read(x) |  |
|  | write(x) |

**RC  
MVCC**

|  |  |
| --- | --- |
| T1 | T2 |
|  | set transaction isolation level read committed; begin; select \* from players; update players set nickname = 'B' where player\_id = 31; |
| set transaction isolation level read committed;  begin;  select \* from players; |  |
|  | commit; |
| select \* from players; |  |

Вывод: RC MVCC не допускает грязное чтение, ведь транзакция фиксирует данный которые есть на её начало, поэтому пока вторая транзакция не закомитит данные, первая транзакция их не увидит.

**RR  
LOCK**

|  |  |
| --- | --- |
| T1 | T2 |
|  | set transaction isolation level repeatable read;  begin;  update players set nickname = 'B' where player\_id = 31; |
| select \* from players LOCK IN SHARE MODE;  Running… |  |
|  | update players set nickname = 'M' where player\_id = 31; |

Вывод: RR LOCK не допускает грязное чтение, ведь транзакция не даёт прочитать данные, которые были изменены другой транзакции, и ждёт завершения второй транзакции

**3. Зависимость неповторяемого чтения**

|  |  |
| --- | --- |
| **T1** | **T2** |
| read(x) |  |
|  | write(x) |
| read(x) |  |

**RC  
MVCC**

|  |  |
| --- | --- |
| T1 | T2 |
| set transaction isolation level read committed;  begin;  select \* from players; |  |
|  | set transaction isolation level read committed;  begin;  update players set nickname = 'C' where player\_id = 31;  Commit; |
| select \* from players; |  |

Вывод: RC MVCC не предотвращает неповторяемое чтение, ведь после того как вторая транзакция закомитит данные, первая транзакция прочитает их уже изменёнными.

**RR  
LOCK**

|  |  |
| --- | --- |
| T1 | T2 |
| set transaction isolation level repeatable read;  begin;  select \* from players LOCK IN SHARE MODE; |  |
|  | set transaction isolation level repeatable read;  begin;  update students set nickname = 'Z' where player\_id = 31;  Running… |
| select \* from players |  |

Вывод: RR LOCK предотвращает неповторяемое чтение, ведь вторая не может изменить данные до того пока первая транзакция не закончит работу.

**4. Зависимость фантомов**

|  |  |
| --- | --- |
| **T1** | **T2** |
| filter(t) |  |
|  | add(t) |
| filter(t) |  |

**RC  
MVCC**

|  |  |
| --- | --- |
| T1 | T2 |
| set transaction isolation level read committed;  begin;  select \* from players where player\_id >= 30; |  |
|  | set transaction isolation level read committed;  begin;  INSERT INTO players (nickname) VALUES ('lab7'); |
| select \* from players where player\_id >= 30; |  |
|  | Commit; |
| select \* from players where player\_id >= 30; |  |

Вывод: RC MVCC не устраняет фантомное чтение, ведь когда вторая транзакция добавляет данные, первая спокойно работает со старыми данными. Но после комита второй транзакции, первая получает новые данные.

**RR  
LOCK**

|  |  |
| --- | --- |
| **T1** | **T2** |
| set transaction isolation level repeatable read;  begin; select \* from players where nickname = ‘osha’ LOCK IN SHARE MODE; empty… |  |
|  | INSERT INTO players (nickname) VALUES ('osha');  Running... |
| select \* from players where nickname = ‘osha’ LOCK IN SHARE MODE; empty… |  |
| commit; |  |
|  | Done |

Вывод: RR LOCK фантомы не допускает, вторая транзакция не может вставить запись, пока первач не закончит работу.