Final Review #2

Monday, May 4, 2015

Final Week

- Today: Revisit transactions
- Wednesday: Final exam

- Reminder: Course evaluations

Grading Announcements

- Class projects will be graded by 05/10
- HW 4 will be graded by 05/15
- Exams will be graded by 05/17
- Final grades will be submitted morning of 05/18
- Final grades will use plus/minus option (A, A-, B+, etc.)
- Grade cut offs will be determined after final exams have been graded

Transactions

- A *transaction* = a sequence of one or more SQL statements treated as a unit of work

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
[SQL statements]
COMMIT; or
ROLLBACK; (=ABORT)
```

or

```
[SQL statement]
COMMIT; or
ROLLBACK; (=ABORT)
```

Recall: ACID Properties

- A
- C
- I
- D

Recall: ACID Properties

- Atomicity
  - Effects of each tx are all-or-nothing; never half undone even if the system crashes in the middle of execution
- Consistency
  - Integrity constraints are guaranteed to hold at the end of a tx if they are satisfied at the start of a tx
- Isolation
  - Txs may be interleaved, but execution must be equivalent to some sequential (serial) order
- Durability
  - Once a tx has committed, its effects remain in the database even if the system crashes immediately after the commit

Without Transactions

Suppose transactions didn't exist and these two updates are run concurrently.
What are the possible final values of graduated students?
Assume initial graduated value = 0.

```
UPDATE Students
SET graduated = graduated + 1000
WHERE college = 'Natural Sciences' AND cohort_year = 2015;
```

concurrent with

```
UPDATE Students
SET graduated = graduated + 1500
WHERE college = 'Natural Sciences' AND cohort_year = 2015;
```

With Transactions

Suppose we have transactions and the same two updates are run concurrently.
What are the possible final values of graduated students?
Assume initial graduated value = 0.

T1

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
UPDATE Students
SET graduated = graduated + 1000
WHERE college = 'Natural Sciences' AND cohort_year = 2015;
COMMIT;
```

concurrent with

T2

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
UPDATE Students
SET graduated = graduated + 1500
WHERE college = 'Natural Sciences' AND cohort_year = 2015;
COMMIT;
```

Practice Problem #1

What are the possible final values of graduated students from T2?
Assume initial graduated value = 0.

T1

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
UPDATE Students
SET graduated = graduated + 500
WHERE college = 'Natural Sciences' AND cohort_year = 2015;
COMMIT;
```

concurrent with

T2

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SELECT  SUM(graduated)
FROM Students
WHERE college = 'Natural Sciences' AND cohort_year = 2015;
```

Practice Problem #2

What are the final values of students who are offered admission?

Assume: gpa > 3.8 = 1000; gpa > 3.45 with highschool_size > 2500 = 5000

T1

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
UPDATE Apply SET decision = 'Y'
WHERE eid IN (SELECT eid FROM Applicants WHERE gpa > 3.8);
COMMIT;
```

concurrent with

T2

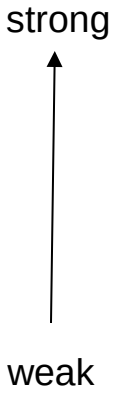
```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
UPDATE Applicants
SET gpa = (1.1) * gpa
WHERE highschool_size > 2500;
COMMIT;
```

Practice Problem #3

What can go wrong with this transaction?

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
<get input from user>
[SQL statements based on input]
<confirm results with user>
If ans='OK' Then
   COMMIT;
Else
   ROLLBACK;
```

Isolation Levels



- Serializability
- Repeatable Read
- Read Committed
- Read Uncommitted

Read Uncommitted

Txs with this isolation level may perform dirty reads

T1

```
UPDATE Students
SET graduated = graduated + 1000
WHERE college = 'Natural Sciences' AND cohort_year = 2015;
COMMIT;
```

concurrent with

T2

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
SELECT SUM(graduated)
FROM Students
WHERE cohort_year = 2015;
```

Read Committed

Txs with this isolation level may read values modified by other concurrently running txs as long as those value have been committed

T1

```
UPDATE Students
SET graduated = graduated + 1000
WHERE college = 'Natural Sciences' AND cohort_year = 2015;
COMMIT;
```

concurrent with

T2

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SELECT SUM(graduated)
FROM Students  WHERE cohort_year = 2015;
SELECT college, SUM(graduated)
FROM Students WHERE cohort_year = 2015
GROUP BY college;
```

Repeatable Read

Txs with this isolation level may read values modified by other txs as long as those values have been committed and those values are unchanged

T1

```
UPDATE Students
SET graduated = graduated + 1000
WHERE college = 'Natural Sciences' AND cohort_year = 2015;
COMMIT;
```

concurrent with

T2

```
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
SELECT SUM(graduated)
FROM Students  WHERE cohort_year = 2015;
SELECT college, SUM(graduated)
FROM Students  WHERE cohort_year = 2015 GROUP BY college;
```

Repeatable Read

Txs with this isolation level may read values modified by other txs as long as those values have been committed and those values are unchanged

T1

```
INSERT INTO Students [new record for cohort year = 2015]
COMMIT;
```

concurrent with

T2

```
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
SELECT SUM(graduated)  FROM Students
WHERE cohort_year = 2015;
SELECT college, SUM(graduated)  FROM Students
WHERE cohort_year = 2015 GROUP BY college;
```

Practice Problem #4

What can go wrong?

T1

```
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
UPDATE Apply SET decision = 'Y' WHERE eid = 1000;
UPDATE Apply SET decision = 'Y' WHERE eid = 2000;
COMMIT;
```

concurrent with

T2

```
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
UPDATE Apply SET major = ‘Physics' WHERE eid = 2000;
UPDATE Apply SET major = 'Biology' WHERE eid = 1000;
COMMIT;
```

Isolation Levels: In-Class Exercise

	dirty	non-repeatable	phantom
Read Uncommitted	Y	Y	Y
Read Committed			
Repeatable Read			
Serializable	N	N	N

Isolation Levels: With Answers

	dirty	non-repeatable	phantom
Read Uncommitted	Y	Y	Y
Read Committed	N	Y	Y
Repeatable Read	N	N	Y
Serializable	N	N	N

COMMIT

(The End)