CS422 Principles of Database Systems

Introduction to Transactions

Chengyu Sun
California State University, Los Angeles

Adapted from Jeffrey Ullman's lecture notes at http://www-db.stanford.edu/~ullman/dscb.html

SQL Statements

```
create table products (
        id              integer primary key,
        category        char(3),
        description     varchar2(2048),
        price           number(10,2)
);

insert into products values (1,'CPU','Intel P4',199.99);
insert into products values (2,'MB','ASUS Motherboard', 128.99);

select * from product;
```

Transaction

- A group of of statements

Transaction starts →

```
select id, price from products;
update products set price = 99.99 where id = 1;
commit;
```

→ Transaction ends

Start and End of A Transaction in Oracle

- Start
  - First DML statement after connection or the end of last transaction
- End
  - First DDL or DCL statement (except `SAVEPOINT`) after a transaction starts
  - Failed DML statements are automatically rolled back
  - Disconnect

ACID

- Database transactions are expected to have *ACID* properties
  - Atomic
  - Consistent
  - Isolated
  - Durable

Atomicity

- A transaction completes or fails as a whole, e.g. either all operations in the transaction are performed or none of them are.
- Example: transfer $100 from account A to account B

```
Read A (SELECT)
If A > 100
   A -= 100 (UPDATE)   ← system crash
   B += 100 (UPDATE)
COMMIT
```

Consistency

- Transaction should preserve database constraints.

Durability

- The changes made by committed transactions are guaranteed to be permanent, despite possible system failures.
- Example: deposit $100 to an account A

```
UPDATE Accounts SET balance = balance+100 WHERE account = 'A';
COMMIT;
```

← system crash

Isolation

- Databases are often accessed by many user at the same time.
- Generally speaking, multiple transactions running concurrently should not interfere with each other.
- More specifically, it should *appear* to the user that the database system execute *one transaction at a time*.

Isolation Example …

Sells

bar	beer	price
Joe's	Bud	2.50
Joe's	Miller	2.75
Sue's	Bud	2.50
Sue's	Miller	3.00

- Sue is querying `Sells` for the highest and lowest price Joe charges.
- Joe decides to stop selling Bud and Miller, but to sell only Heineken at $3.50

… Isolation Example

Sue's transaction:

```
-- MAX
SELECT MAX(price) FROM Sells WHERE bar='Joe''s';
-- MIN
SELECT MIN(price) FROM Sells WHERE bar='Joe''s';
COMMIT;
```

Joe's transaction:

```
-- DEL
DELETE FROM Sells WHERE bar='Joe''s';
-- INS
INSERT INTO Sells VALUES( 'Joe''s', 'Heineken', 3.50 );
COMMIT;
```

Potential Problems of Concurrent Transactions

- Caused by *interleaving operations*
- Caused by *aborted operations*

SQL Isolation Levels

- Serializable
- Repeatable read
- Read committed
- Read uncommitted

Read Uncommitted

- May read data written by an transaction that has not committed (and may never)
- For example, Sue may see the price 3.50 even if Joe's transaction later aborts

Read Committed

- Read only committed data, but not necessarily the same data every time.
- For example, the interleaving of (MAX)(DEL)(INS)(MIN) is possible
  - MAX 2.75
  - MIN 3.50

Read Repeatable

- Read only committed data, and, everything seen the first time will be seen the second time.
- For example, the interleaving of (MAX)(DEL)(INS)(MIN) is still possible, however:
  - MAX 2.75
  - MIN 2.50

Serializable

- It appears to the user that the transactions are executed one at a time.
- For example, Sue will see either
  - MAX 2.75 and MIN 2.50, or
  - MAX 3.50 and MIN 3.50

Isolation Levels in Oracle

- Only `READ COMMITTED` and `SERIALIZABLE` are supported
- `READ COMMITTED` is default
- Change to serializable:

```
set transaction isolation level serializable;
```

Beyond Introduction

- Implementation of concurrency control and failure recovery is quite complex
- Read Chapter 17, 18, 19 or take CS522 if you are interested.