Database Systems
CSE 414

Lecture 20: Introduction to Transactions
Announcements

• HW6 due on Wednesday

• WQ6 available for one more day

• WQ7 (last one!) due on Sunday
Demo
(see lec20-transactions-intro.sql)
Challenges

• Want to execute many apps concurrently
  – All these apps read and write data to the same DB

• Simple solution: only serve one app at a time
  – What’s the problem?

• Better: multiple operations need to be executed *atomically* over the DB
What can go wrong?

• Manager: balance budgets among projects
  – Remove $10k from project A
  – Add $7k to project B
  – Add $3k to project C

• CEO: check company’s total balance
  – SELECT SUM(money) FROM budget;

• This is called a dirty / inconsistent read aka WRITE-READ conflict
What can go wrong?

• App 1:
  SELECT inventory FROM products WHERE pid = 1

• App 2:
  UPDATE products SET inventory = 0 WHERE pid = 1

• App 1:
  SELECT inventory * price FROM products
  WHERE pid = 1

• This is known as an unrecoverable read aka READ-WRITE conflict
What can go wrong?

Account 1 = $100
Account 2 = $100
Total = $200

• App 1:
  – Set Account 1 = $200
  – Set Account 2 = $0

• App 2:
  – Set Account 2 = $200
  – Set Account 1 = $0

• At the end:
  – Total = $200

• App 1: Set Account 1 = $200

• App 2: Set Account 2 = $200

• App 1: Set Account 2 = $0

• App 2: Set Account 1 = $0

• At the end:
  – Total = $0

This is called the lost update aka WRITE-WRITE conflict
What can go wrong?

• Buying tickets to the next Bieber concert:
  – Fill up form with your mailing address
  – Put in debit card number
  – Click submit
  – Screen shows money deducted from your account
  – [Your browser crashes]

Changes to the database should be **ALL or NOTHING**
Transactions

• Collection of statements that are executed atomically (logically speaking)

BEGIN TRANSACTION
[SQL statements]
COMMIT or
ROLLBACK (=ABORT)

[single SQL statement]

If BEGIN… missing, then TXN consists of a single instruction
Transactions Demo
(see lec20-transactions-intro.sql)
Serial execution

• **Definition**: A SERIAL execution of transactions is one where each transaction is executed one after another.

• **Fact**: Nothing can go wrong if the DB executes transactions serially.

• **Definition**: A SERIALIZABLE execution of transactions is one that is equivalent to a serial execution.
ACID Transactions

- **Atomic**
  - State shows either all the effects of txn, or none of them
- **Consistent**
  - Txn moves from a state where integrity holds, to another where integrity holds
- **Isolated**
  - Effect of txns is the same as txns running one after another (i.e., looks like batch mode)
- **Durable**
  - Once a txn has committed, its effects remain in the database
Atomic

• **Definition**: A transaction is ATOMIC if all its updates must happen or not at all.

• **Example**: move $100 from A to B

```sql
BEGIN TRANSACTION;
UPDATE accounts SET bal = bal – 100 WHERE acct = A;
UPDATE accounts SET bal = bal + 100 WHERE acct = B;
COMMIT;
```

Crash!
**Isolated**

- **Definition** An execution ensures that txns are isolated, if the effect of each txn is as if it were the only txn running on the system.

- **Example**: Alice deposits $100, Bob withdraws $100 from account

```sql
BEGIN TRANSACTION;
x = select bal from accounts
   where acct = A;
x = x+100
update accounts
   set bal = x where acct = A;
COMMIT;
```

```sql
BEGIN TRANSACTION;
y = select bal from accounts
   where acct = A;
if y < 100 return “Error”
y = y - 100
update accounts
   set bal = y where acct = A;
COMMIT;
```
Consistent

• Recall: integrity constraints govern how values in tables are related to each other
  – Example: account.bal >= 0
  – Example: foreign key constraints

• Can be enforced by the DBMS or by the app

• How consistency is achieved by the app:
  – App programmer ensures that txns only takes a consistent DB state to another consistent state
  – DB makes sure that txns are executed atomically

• Can defer checking the validity of constraints until the end of a transaction
Durable

- A transaction is durable if its effects continue to exist after the transaction and even after the program has terminated

- How? By writing to disk
  - (often multiple disks since individual disks fail)
Rollback transactions

• If the app gets to a state where it cannot complete the transaction successfully, execute ROLLBACK

• The DB returns to the state prior to the transaction
ACID

• Atomic
• Consistent
• Isolated
• Durable

• Enjoy this in HW7!

• Note: by default each statement is its own txn
  – Exception: if auto-commit is off, then each statement
    starts a new txn
Transactions

Jim Gray

• Inventor of ACID transactions, 2PL, data cubes, ...
• Joined Microsoft in 1995
• Won the Turing Award in 1998
• His book “Transaction Processing” is probably still the best work on database implementation