

Introduction to Data Management

CSE 344

Lecture 20: Introduction to Transactions

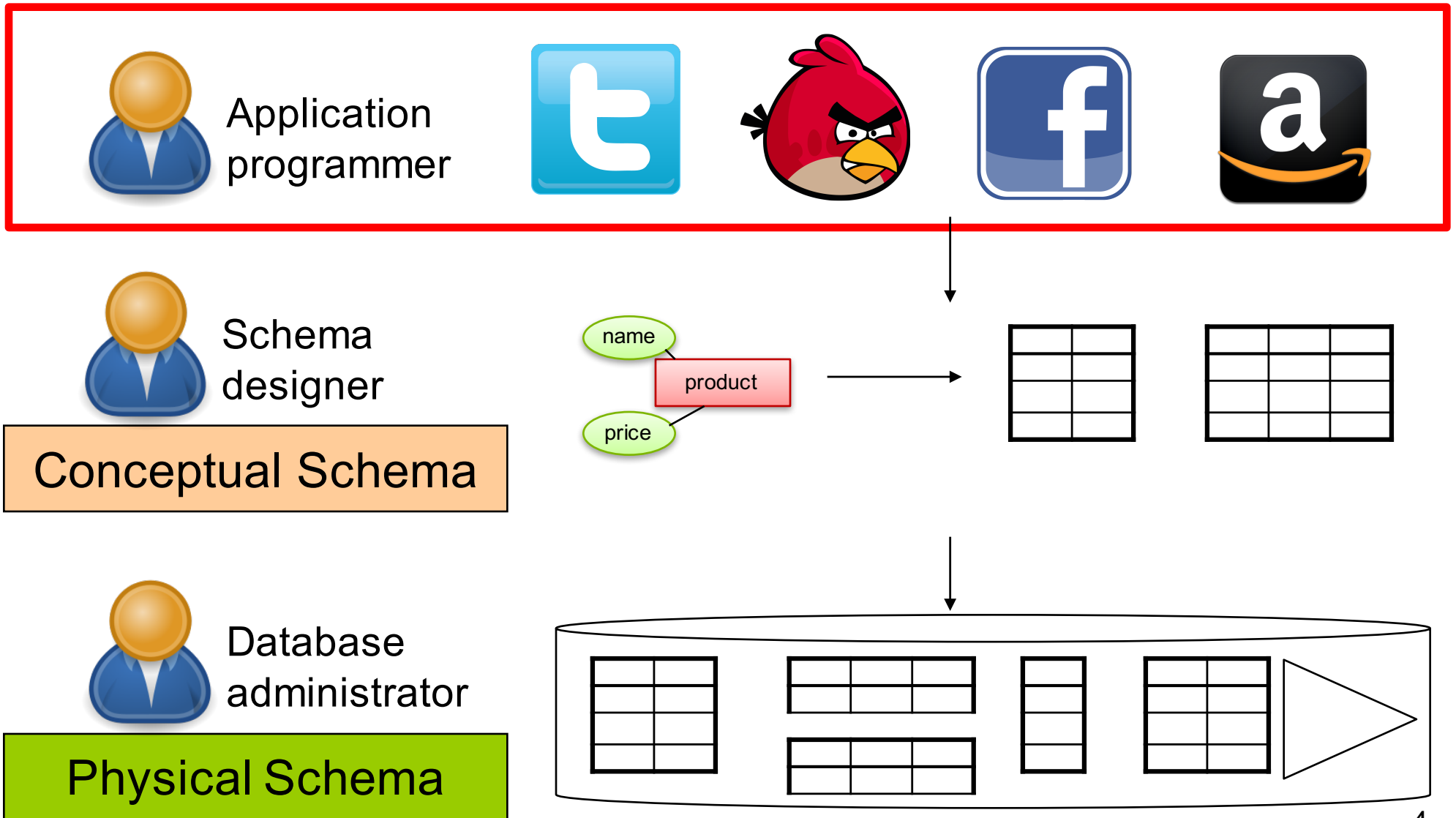
Announcements

- WQ6, HW6 due next Monday
- WQ7, HW7 will be out next Monday

Schema Refinements = Normal Forms

- 1st Normal Form = all tables are flat
- 2nd Normal Form = obsolete
- Boyce Codd Normal Form = no bad FDs
- 3rd and 4th Normal Form = see book
 - BCNF is lossless but can cause loss of ability to check some FDs (see book 3.4.4)
 - 3NF fixes that (is lossless and dependency-preserving), but some tables might not be in BCNF – i.e., they may have redundancy anomalies
 - 4NF deals with multi-valued dependencies (see book 3.6)

Data Management Pipeline



Transactions

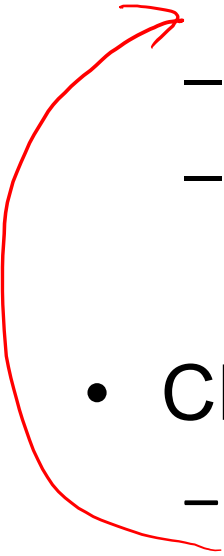
- We use database transactions everyday
 - Bank \$\$\$ transfers
 - Online shopping
 - Signing up for classes
- For this class, a transaction is a series of DB queries
 - Read / Write / Update / Delete / Insert
 - Unit of work issued by a user that is independent from others

What's the big deal?

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?
- **Want: multiple operations to be executed *atomically* over the same DBMS**

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 a **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 unrepeatable read
aka **READ-WRITE** conflict

What can go wrong?

Account 1 = \$100

Account 2 = \$100

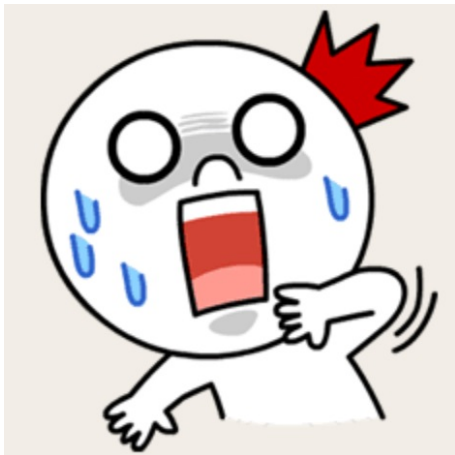
Total = \$200

- 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]



Lesson:

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
```

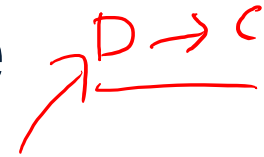
```
ROLLBACK (=ABORT)
```

ALL →
NOTHING

```
[single SQL statement]
```

If BEGIN... missing,
then TXN consists
of a single instruction

In-class Exercise



- Given 3 relations: $R(\underline{A}, \underline{B}, C)$, $S(C, \underline{D})$, $T(D, A)$
Show the key of the query's answer, and compute D^+ :

select R.A, R.B, R.C, S.D from R, S
where $R.C = S.C$ and $R.A = 20$;

Key =

$D^+ =$

select T.A, S.C, S.D from S, T where S.D = T.D;

Key =

$D^+ =$

In-class Exercise

- Given 3 relations: $R(\underline{A}, \underline{B}, C)$, $S(C, \underline{D})$, $T(D, A)$
Show the key of the query's answer, and compute D^+ :

```
select R.A, R.B, R.C, S.D from R, S
where R.C = S.C and R.A = 20;
```

Key = BD $D^+ =$ ACD

```
select T.A, S.C, S.D from S, T where
S.D = T.D;
```

Key = AD $D^+ =$ CD

Transactions Demo

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
 - (Up to everything that we have learned so far)
- **Definition:** A SERIALIZABLE execution of transactions is one that is equivalent to a serial execution