Database Management System

Lecture 1

Introduction to Relational Database

* Some materials adapted from R. Ramakrishnan, J. Gehrke

Today’s Agenda

- Course Layout
- Introduction to Relational Database
- Overview of SQL and MySQL

Course Layout

About this Course

- Instructor
  - Seongjin Lee
  - Email: insight@gnu.ac.kr
  - Office: 407-314
  - Office Hour: Every Thursday 11:00-12:00 or Make appointment
- Class
  - Time: Thursday 16:00-19:00
  - Place: 407-202
- Course webpage
  - http://203.255.57.228/MediaWiki/index.php?title=Database
- Textbook
  - Database Management System, 3rd Ed., Ramakrishnan & Gerhke, McGraw Hill, 2003

About this Course

- Goal – To cover major topics in database management system
- Evaluation
  - Attendance – 10%
  - Quiz – 10% Short quizzes at the beginning of most classes
  - Assignments – 10%
  - Project – 20%
  - Midterm – 30%
  - Final - 30%
  - Closed book and notes
  - Request for regrade within one week upon return; describe reasons in writing
    - what and why the score is incorrect or unfair
    - The written argument must be self-contained

Topics

- Relational Database
- SQL
- Logical Database Design
  - Conceptual Modeling (Entity-Relationship Diagram)
  - Normalization
- Database Internals
  - Storage and Indexing
  - Query Optimization
- Physical Database Design
- Transaction and Recovery

About this Course

- Reading Assignments – Due before following class period
- Up to 6 Homework assignments (may be less)
  - Please turn in assignments by the due date (check the website)
- Attendance
  - Participation is important part of this course
  - 3 absences without prior arrangement will lower your grade by one letter (each subsequent 1 absences will lower a grade by one letter)

About this Course

- Academic Honesty
  - Assignments, quizzes, and exams done individually
  - No lying, cheating, copying
  - If found, no grade for that particular assessment
  - Suspicious work will be questioned thoroughly

About this Course

- No classes on 추석 that is on October 5th
- Midterm and Final
  - Close book and notes
  - Midterm on Oct. 19th (in class)
  - Final on Dec. 14th (in class)
- Quizzes
  - To give feedback on your understanding of material as well as help with material
  - at the beginning of most of the classes
  - Based on the previous class material

Introduction to Relational Database

What Is a DBMS?

- A very large, integrated collection of data.
- Models real-world enterprise.
  - Entities (e.g., students, courses)
  - Relationships (e.g., Madonna is taking CS564)
- A Database Management System (DBMS) is a software package designed to store and manage databases.

Files vs. DBMS

- Application must stage large datasets between main memory and secondary storage (e.g., buffering, page-oriented access, 32-bit addressing, etc.)
- Special code for different queries
- Must protect data from inconsistency due to multiple concurrent users
- Crash recovery
- Security and access control

Why Use a DBMS?

- Data independence and efficient access.
- Reduced application development time.
- Data integrity and security.
- Uniform data administration.
- Concurrent access, recovery from crashes.

Why Study Databases?? (1/2)

- It is critical to government, business, science, etc.
- Many tech companies are build on data management (Google, Amazon, Facebook, Netflix, etc.)
  - or they offer database products (IBM, Oracle, Microsoft, etc.)
- Shift from computation to information
  - at the “low end”: scramble to web space (a mess!)
  - at the “high end”: scientific applications
- Datasets increasing in diversity and volume.
  - Digital libraries, interactive video, Human Genome project, EOS project
  - ... need for DBMS exploding

Why Study Databases?? (2/2)

- It spans major areas of computer engineering
  - Operating systems (file, memory, process management)
  - Theory (languages, algorithms, complexity)
  - Artificial Intelligence (knowledge-based systems, logic, search)
  - Software Engineering (application development)
  - Data structures (trees, hashtables)
- Data may be very large
  - Amazon > 42TB
  - Youtube > 45TB
  - AT&% > 323TB
  - National Energy Research Scientific Computing Center > 2.8 Petabytes

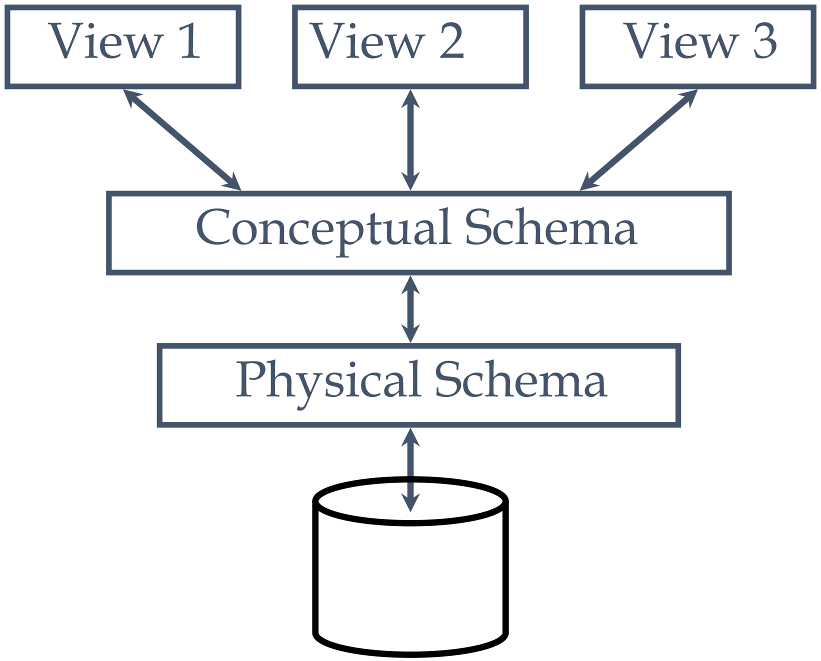
1 TB = 1,000,000,000,000 Bytes 1 PB = 1,000,000,000,000,000 Bytes

Data Models

- A data model is a collection of concepts for describing data.
- A schema is a description of a particular collection of data, using the a given data model.
- The relational model of data is the most widely used model today.
  - Main concept: relation, basically a table with rows and columns.
  - Every relation has a schema, which describes the columns, or fields.

Levels of Abstraction

- Many views, single conceptual (logical) schema and physical schema.
  - Views describe how users see the data.
  - Conceptual schema defines logical structure
  - Physical schema describes the files and indexes used.



☛ *Schemas are defined using DDL; data is modified/queried using DML.*

Example: University Database

- Conceptual schema:

  Students(sid: string, name: string, login: string, age: integer, gpa:real)

  Courses(cid: string, cname:string, credits:integer)

  Enrolled(sid:string, cid:string, grade:string)

- Physical schema:
  - Relations stored as unordered files.
  - Index on first column of Students.
- External Schema (View):

  Course_info(cid:string,enrollment:integer)

Data Independence*

- Applications insulated from how data is structured and stored.
- Logical data independence: Protection from changes in logical structure of data.
- Physical data independence: Protection from changes in physical structure of data.

☛ *One of the most important benefits of using a DBMS!*

Concurrency Control

- Concurrent execution of user programs is essential for good DBMS performance.
  - Because disk accesses are frequent, and relatively slow, it is important to keep the CPU humming by working on several user programs concurrently.
- Interleaving actions of different user programs can lead to inconsistency: e.g., check is cleared while account balance is being computed.
- DBMS ensures such problems don't arise: users can pretend they are using a single-user system.

Transaction: An Execution of a DB Program

- Key concept is transaction, which is an atomic sequence of database actions (reads/writes).
- Each transaction, executed completely, must leave the DB in a consistent state if DB is consistent when the transaction begins.
  - Users can specify some simple integrity constraints on the data, and the DBMS will enforce these constraints.
  - Beyond this, the DBMS does not really understand the semantics of the data. (e.g., it does not understand how the interest on a bank account is computed).
  - Thus, ensuring that a transaction (run alone) preserves consistency is ultimately the user's responsibility!

Scheduling Concurrent Transactions

- DBMS ensures that execution of {T1, ... , Tn} is equivalent to some serial execution T1’ ... Tn’.
  - Before reading/writing an object, a transaction requests a lock on the object, and waits till the DBMS gives it the lock. All locks are released at the end of the transaction. (Strict 2PL locking protocol.)
  - Idea: If an action of Ti (say, writing X) affects Tj (which perhaps reads X), one of them, say Ti, will obtain the lock on X first and Tj is forced to wait until Ti completes; this effectively orders the transactions.
  - What if Tj already has a lock on Y and Ti later requests a lock on Y? (Deadlock!) Ti or Tj is aborted and restarted!

Ensuring Atomicity

- DBMS ensures *atomicity* (all-or-nothing property) even if system crashes in the middle of a Xact.
- Idea: Keep a log (history) of all actions carried out by the DBMS while executing a set of Xacts:
  - Before a change is made to the database, the corresponding log entry is forced to a safe location. (WAL protocol; OS support for this is often inadequate.)
  - After a crash, the effects of partially executed transactions are undone using the log. (Thanks to WAL, if log entry wasn't saved before the crash, corresponding change was not applied to database!)

The Log

- The following actions are recorded in the log:
  - Ti writes an object: The old value and the new value.
    - Log record must go to disk before the changed page!
  - Ti commits/aborts: A log record indicating this action.
- Log records chained together by Xact id, so it's easy to undo a specific Xact (e.g., to resolve a deadlock).
- Log is often duplexed and archived on "stable" storage.
- All log related activities (and in fact, all CC related activities such as lock/unlock, dealing with deadlocks etc.) are handled transparently by the DBMS.

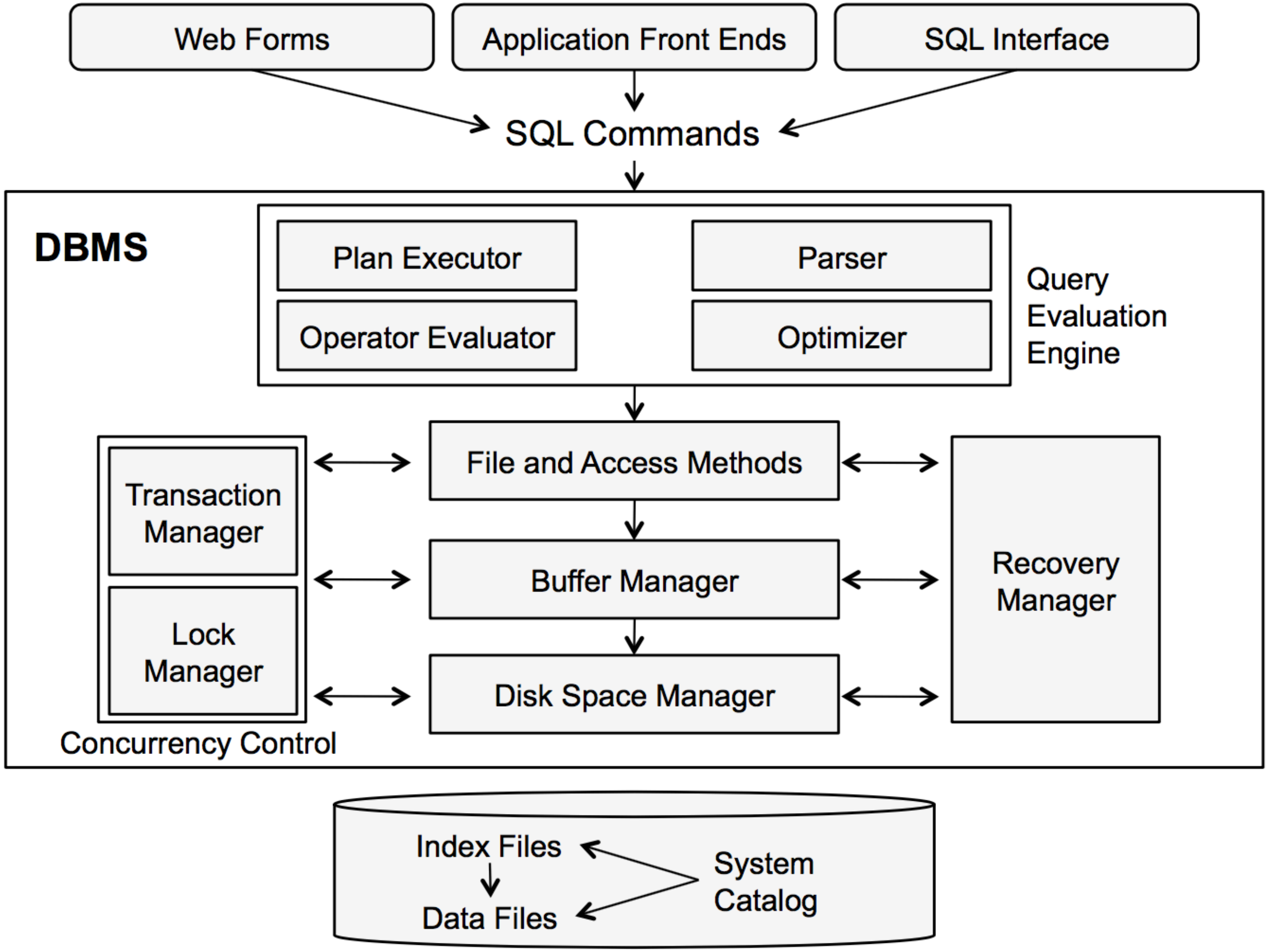
Databases make these folks happy ...

- End users and DBMS vendors
- DB application programmers
  - E.g., smart webmasters
- Database administrator (DBA)
  - Designs logical /physical schemas
  - Handles security and authorization
  - Data availability, crash recovery
  - Database tuning as needs evolve

Must understand how a DBMS works!

Structure of a DBMS (Fig. 1.3, p. 20)

- A typical DBMS has a layered architecture.
- This is one of several possible architectures; each system has its own variations.



Summary of Chapter 1

- DBMS used to maintain, query large datasets.
- Benefits include recovery from system crashes, concurrent access, quick application development, data integrity and security.
- Levels of abstraction give data independence.
- A DBMS typically has a layered architecture.
- DBAs hold responsible jobs and are well-paid! ☺
- DBMS R&D is one of the broadest, most exciting areas in CS.

The Relational Model

Chapter 3 to 3.3

The terminology

Relational Database

- Assume the following table (A.K.A Relation)has been defined to keep track of students

Account

Number	Name	Balance	Type
7003001	Jane Smith	1,000,000	Savings
7003003	Alfred Hitchcock	4,400,200	Savings
7003005	Takumi Fujiwara	2,230,000	Checking
7003007	Brian Mills	1,200,000	Savings
7003009	Jason Bourn	3,025,000	Checking

Terminology

- The name of the table (relation) : Students

Account

Number	Name	Balance	Type
7003001	Jane Smith	1,000,000	Savings
7003003	Alfred Hitchcock	4,400,200	Savings
7003005	Takumi Fujiwara	2,230,000	Checking
7003007	Brian Mills	1,200,000	Savings
7003009	Jason Bourn	3,025,000	Checking

Terminology

- The name of the attributes (columns): SID, NAME, Department, GPA)

Account

Number	Name	Balance	Type
7003001	Jane Smith	1,000,000	Savings
7003003	Alfred Hitchcock	4,400,200	Savings
7003005	Takumi Fujiwara	2,230,000	Checking
7003007	Brian Mills	1,200,000	Savings
7003009	Jason Bourn	3,025,000	Checking

Terminology

- Schema of the table
  - Definition and structure of the relation (includes types and constraints)

Account

Number	Name	Balance	Type
7003001	Jane Smith	1,000,000	Savings
7003003	Alfred Hitchcock	4,400,200	Savings
7003005	Takumi Fujiwara	2,230,000	Checking
7003007	Brian Mills	1,200,000	Savings
7003009	Jason Bourn	3,025,000	Checking

Constraints

- For every attribute of every relation, the schema specifies allowable values

Account(Number: integer, name: string,
Balance: currency, type:string)

- The allowable values for an attribute is called the “domain” of the attribute

Terminology

- Each entry in the relation is called “row”, “tuple”, or “record”
- Instance of the schema is the current set of rows

Account

rows

Number	Name	Balance	Type
7003001	Jane Smith	1,000,000	Savings
7003003	Alfred Hitchcock	4,400,200	Savings
7003005	Takumi Fujiwara	2,230,000	Checking
7003007	Brian Mills	1,200,000	Savings
7003009	Jason Bourn	3,025,000	Checking

instance

Terminology

- “Degree” of relation is the number of attributes
- “Cardinality” of relation is the number of rows in the current instance

Account

Number	Name	Balance	Type
7003001	Jane Smith	1,000,000	Savings
7003003	Alfred Hitchcock	4,400,200	Savings
7003005	Takumi Fujiwara	2,230,000	Checking
7003007	Brian Mills	1,200,000	Savings
7003009	Jason Bourn	3,025,000	Checking

Cardinality

Degree of this relation is 4
(there are four attributes,
SID, Name, Department, GPA)

Cardinality of this instance is 6
(There are 6 rows)

Terminology

- Each Table has key
- The value of the key must be unique
- What is the key for the each relations

Account

Number	Name	Balance	Type
7003001	Jane Smith	1,000,000	Savings
7003003	Alfred Hitchcock	4,400,200	Savings
7003005	Takumi Fujiwara	2,230,000	Checking
7003007	Brian Mills	1,200,000	Savings
7003009	Jason Bourn	3,025,000	Checking

Deposit

Accnt	TxID	Date	Amount
7003007	1	16-Aug-2017	200
7003001	2	19-Aug-2017	400
7003003	3	26-Aug-2017	100
7003009	4	26-Aug-2017	840

Check

Number	ChkNo	Date	Amount
7003003	123	29-Aug-2017	840
7003003	124	30-Aug-2017	320

Terminology

- Key consists of one or more attributes
- Generally underline the key attributes

Account

<u>Number</u>	Name	Balance	Type
7003001	Jane Smith	1,000,000	Savings
7003003	Alfred Hitchcock	4,400,200	Savings
7003005	Takumi Fujiwara	2,230,000	Checking
7003007	Brian Mills	1,200,000	Savings
7003009	Jason Bourn	3,025,000	Checking

Deposit

Accnt	<u>TxID</u>	Date	Amount
7003007	1	16-Aug-2017	200
7003001	2	19-Aug-2017	400
7003003	3	26-Aug-2017	100
7003009	4	26-Aug-2017	840

Check

<u>Number</u>	<u>ChkNo</u>	Date	Amount
7003003	123	29-Aug-2017	840
7003003	124	30-Aug-2017	320

Constraints

- See the Deposit Relations
- How can we prevent it from happening?
  - Use foreign Key
- Deposit.Accnt is a **foreign key** that references Account.Number
- Called enforcing referential integrity

Account

Number	Name	Balance	Type
7003001	Jane Smith	1,000,000	Savings
7003003	Alfred Hitchcock	4,400,200	Savings
7003005	Takumi Fujiwara	2,230,000	Checking
7003007	Brian Mills	1,200,000	Savings
7003009	Jason Bourn	3,025,000	Checking

Deposit

Accnt	TxID	Date	Amount
7003007	1	16-Aug-2017	200
7003001	2	19-Aug-2017	400
7003003	3	26-Aug-2017	100
7003009	4	26-Aug-2017	840
~~7003011~~	~~5~~	~~27-Aug-2017~~	~~2100~~

Terminology

- Foreign keys may or may not be part of the key for the relation
- Deposit.Accnt is not part of the key for Deposit
- Check.Number is part of the key for check

Account

Number	Name	Balance	Type
7003001	Jane Smith	1,000,000	Savings
7003003	Alfred Hitchcock	4,400,200	Savings
7003005	Takumi Fujiwara	2,230,000	Checking
7003007	Brian Mills	1,200,000	Savings
7003009	Jason Bourn	3,025,000	Checking

Deposit

Accnt	TxID	Date	Amount
7003007	1	16-Aug-2017	200
7003001	2	19-Aug-2017	400
7003003	3	26-Aug-2017	100
7003009	4	26-Aug-2017	840

Check

Number	ChkNo	Date	Amount
7003003	123	29-Aug-2017	840
7003003	124	30-Aug-2017	320

More on Schema

- Select the tables, with a name for each table
  - a database schema may have multiple tables
  - Each table has its own schema

- Select attributes for each table and give the domain for each attribute

- Specify the key(s) for each table
  - there can be more than 1 key for a table

- Specify all appropriate foreign keys

Exercise

- Create a relation with one key
- Create a relation with some foreign keys based on the first relation

Structured Query Language (SQL)

Chapter 3.4

Structured Query Language (SQL)

- It is the language used to talk to DBMS, and serves many purposes

Account(Number: integer, name: character,
Balance: currency, type:character)

- To define above schema

```
CREATE TABLE Account (
        Number integer NOT NULL,
        Name character,
        Balance currency,
        Type character,
        PRIMARY KEY (Number)
)
```

Structured Query Language (SQL)

- To query the database

*SELECT **
FROM Account
WHERE Type = “checking”;

- To insert rows into a table:

INSERT INTO Account
VALUES (106, “Lucy Lou”, 124000, “savings”);

Query Example 1

- Show the account that made deposit more than 500

SELECT Accnt, Amount
FROM Deposit
WHERE Amount > 500;

Deposit

Accnt	TxID	Date	Amount
7003007	1	16-Aug-2017	200
7003001	2	19-Aug-2017	400
7003003	3	26-Aug-2017	100
7003009	4	26-Aug-2017	840

Accnt	TxID	Date	Amount
7003007	1	16-Aug-2017	200
7003001	2	19-Aug-2017	400
7003003	3	26-Aug-2017	100
7003009	4	26-Aug-2017	840

- SQL queries return new tables representing the answer to the query

Query Example 1 cont'd

- SQL queries return new tables representing the answer to the query

Deposit

Accnt	Amount
7003009	840

Query Example 2

- Show the transaction number 3

*SELECT **
FROM Deposit
WHERE TxID = 3;

Deposit

Accnt	TxID	Date	Amount
7003007	1	16-Aug-2017	200
7003001	2	19-Aug-2017	400
7003003	3	26-Aug-2017	100
7003009	4	26-Aug-2017	840

- Each row is checked to see if WHERE clause evaluates to true
- “*” Return every column

Exercises

- Write the query to show all information from accounts with checking type
- How about all accounts with savings type
- If you want to know the owner of the accounts with checking type, how would you write the query

Query Example 3

*SELECT **
FROM Account
WHERE Type = “checking”
AND
Type = “savings”;

Account

Number	Name	Balance	Type
7003001	Jane Smith	1,000,000	Savings
7003003	Alfred Hitchcock	4,400,200	Savings
7003005	Takumi Fujiwara	2,230,000	Checking
7003007	Brian Mills	1,200,000	Savings
7003009	Jason Bourn	3,025,000	Checking

- What would be the outcome of the query?

Empty Query Results

*SELECT **
FROM Account
WHERE Type = “checking”
AND
Type = “savings”;

Account

Number	Name	Balance	Type
7003001	Jane Smith	1,000,000	Savings
7003003	Alfred Hitchcock	4,400,200	Savings
7003005	Takumi Fujiwara	2,230,000	Checking
7003007	Brian Mills	1,200,000	Savings
7003009	Jason Bourn	3,025,000	Checking

- It is not error, and can be informative in a sense
- Because of the domain of the type attribute will never give any result to the query

Evaluating the SQL Query

1. FROM clause tells the input tables
2. Where clause is evaluated for all combinations of rows from the input tables
3. SELECT clause decides which attributes remains in the query result

SELECT Accnt, Amount
FROM Deposit
WHERE Amount > 500;

Queries over two tables: a ”Join”

```
SELECT A.Name, A.Balance
FROM Account A, Deposit D
WHERE A.Number = D.Account AND
        A.Balance > 2,200,000;
```

- Which rows from which tables are evaluated in the WHERE clause?
- “A” is a correlation name for Account
- “D” is a correlation name for Deposit
    - Correlation name acts as a local variable
    - Holds one row from the corresponding table
    - Table name can also be used as a correlation name, but it is longer

Query Example 4

Account

<u>Number</u>	Name	Balance	Type
7003001	Jane Smith	1,000,000	Savings
7003003	Alfred Hitchcock	4,400,200	Savings
7003005	Takumi Fujiwara	2,230,000	Checking
7003007	Brian Mills	1,200,000	Savings
7003009	Jason Bourn	3,025,000	Checking

Deposit

Accnt	<u>TxID</u>	Date	Amount
7003007	1	16-Aug-2017	200
7003001	2	19-Aug-2017	400
7003003	3	26-Aug-2017	100
7003009	4	26-Aug-2017	840

SELECT A.Name, A.Balance
FROM Account A, Deposit D
WHERE A.Number = D.Account AND A.Balance > 2,200,000*;*

- **Check every combination of one row from each table**

Query Example 4

Account

Number	Name	Balance	Type
7003001	Jane Smith	1,000,000	Savings
7003003	Alfred Hitchcock	4,400,200	Savings
7003005	Takumi Fujiwara	2,230,000	Checking
7003007	Brian Mills	1,200,000	Savings
7003009	Jason Bourn	3,025,000	Checking

Deposit

Accnt	TxID	Date	Amount
7003007	1	16-Aug-2017	200
7003001	2	19-Aug-2017	400
7003003	3	26-Aug-2017	100
7003009	4	26-Aug-2017	840

Evaluate each row
If not true, Throw!

SELECT A.Name, A.Balance
FROM Account A, Deposit D
WHERE A.Number = D.Account AND A.Balance >* 2,200,000*;

Intermediate Query Result Table

Number	Name	Balance	Type	Accnt	TxID	Date	Amount

Query Example 4

Account

Number	Name	Balance	Type
7003001	Jane Smith	1,000,000	Savings
7003003	Alfred Hitchcock	4,400,200	Savings
7003005	Takumi Fujiwara	2,230,000	Checking
7003007	Brian Mills	1,200,000	Savings
7003009	Jason Bourn	3,025,000	Checking

Deposit

Accnt	TxID	Date	Amount
7003007	1	16-Aug-2017	200
7003001	2	19-Aug-2017	400
7003003	3	26-Aug-2017	100
7003009	4	26-Aug-2017	840

Evaluate each row
If not true, Throw!

SELECT A.Name, A.Balance
FROM Account A, Deposit D
WHERE A.Number = D.Account AND A.Balance > 2,200,000;

Intermediate Query Result Table

Number	Name	Balance	Type	Accnt	TxID	Date	Amount

Query Example 4

Account

Number	Name	Balance	Type
7003001	Jane Smith	1,000,000	Savings
7003003	Alfred Hitchcock	4,400,200	Savings
7003005	Takumi Fujiwara	2,230,000	Checking
7003007	Brian Mills	1,200,000	Savings
7003009	Jason Bourn	3,025,000	Checking

Deposit

Accnt	TxID	Date	Amount
7003007	1	16-Aug-2017	200
7003001	2	19-Aug-2017	400
7003003	3	26-Aug-2017	100
7003009	4	26-Aug-2017	840

Evaluate each row
If true, Take!

SELECT A.Name, A.Balance
FROM Account A, Deposit D
WHERE A.Number = D.Account AND A.Balance >* 2,200,000*;

Intermediate Query Result Table

Number	Name	Balance	Type	Accnt	TxID	Date	Amount
7003003	Alfred Hitchcock	4,400,200	Savings	7003003	3	26-Aug-2017	100

Query Example 4

Account

<u>Number</u>	Name	Balance	Type
7003001	Jane Smith	1,000,000	Savings
7003003	Alfred Hitchcock	4,400,200	Savings
7003005	Takumi Fujiwara	2,230,000	Checking
7003007	Brian Mills	1,200,000	Savings
7003009	Jason Bourn	3,025,000	Checking

Deposit

Accnt	<u>TxID</u>	Date	Amount
7003007	1	16-Aug-2017	200
7003001	2	19-Aug-2017	400
7003003	3	26-Aug-2017	100
7003009	4	26-Aug-2017	840

Evaluate each row
If true, Take!

SELECT A.Name, A.Balance
FROM Account A, Deposit D
WHERE A.Number = D.Account AND A.Balance > 2,200,000;

Intermediate Query Result Table

Number	Name	Balance	Type	Accnt	TxID	Date	Amount
7003003	Alfred Hitchcock	4,400,200	Savings	7003003	3	26-Aug-2017	100
7003009	Jason Bourn	3,025,000	Checking	7003009	4	26-Aug-2017	840

Query Example 4

Account

<u>Number</u>	Name	Balance	Type
7003001	Jane Smith	1,000,000	Savings
7003003	Alfred Hitchcock	4,400,200	Savings
7003005	Takumi Fujiwara	2,230,000	Checking
7003007	Brian Mills	1,200,000	Savings
7003009	Jason Bourn	3,025,000	Checking

Deposit

Accnt	<u>TxID</u>	Date	Amount
7003007	1	16-Aug-2017	200
7003001	2	19-Aug-2017	400
7003003	3	26-Aug-2017	100
7003009	4	26-Aug-2017	840

Evaluate each row
If true, Take!

SELECT A.Name, A.Balance
FROM Account A, Deposit D
WHERE A.Number = D.Account AND A.Balance > 2,200,000*;*

Final Query Result Table

Name	Balance
Alfred Hitchcock	4,400,200
Jason Bourn	3,025,000

Useful Keyword: DISTINCT

- Suppose

Intermediate Query Result Table

Number	Name	Balance	Type	Accnt	TxID	Date	Amount
7003003	Alfred Hitchcock	4,400,200	Savings	7003003	3	26-Aug-2017	100
7003009	Jason Bourn	3,025,000	Checking	7003009	4	26-Aug-2017	840
7003009	Jason Bourn	3,025,000	Checking	7003009	5	27-Aug-2017	1000

*SELECT **DISTINCT** A.Name, A.Balance*
FROM Account A, Deposit D
WHERE A.Number = D.Account AND A.Balance > 2,200,000*;*

- DISTINCT removes duplicate rows

Final Query Result Table

Name	Balance
Alfred Hitchcock	4,400,200
Jason Bourn	3,025,000
Jason Bourn	3,025,000

w/o DISTINCT

Final Query Result Table

Name	Balance
Alfred Hitchcock	4,400,200
Jason Bourn	3,025,000

w/ DISTINCT

Exercise

Account

Number	Name	Balance	Type
7003001	Jane Smith	1,000,000	Savings
7003003	Alfred Hitchcock	4,400,200	Savings
7003005	Takumi Fujiwara	2,230,000	Checking
7003007	Brian Mills	1,200,000	Savings
7003009	Jason Bourn	3,025,000	Checking

Deposit

Accnt	TxID	Date	Amount
7003007	1	16-Aug-2017	200
7003001	2	19-Aug-2017	400
7003003	3	26-Aug-2017	100
7003009	4	26-Aug-2017	840

SELECT A.Number, A.Name, D.Date, D.Amount
FROM Account A, Deposit D
WHERE A.Number = D.Account AND D.Amount > 100*;*

- How many rows will be in the query result?
- How may columns will be in the query result?

Notes on Queries

- A query is expressed against the schema
- But, the query is executed against the instance, that is against the data
- The result of a query is always a table
  - resulting table do not always have a name
  - Attributes deduced from input tables
  - result may not have any rows

Notes on Queries

- Self Joins – Here, A1 and A2 refer to copies of the same instance

```
SELECT A1.Number, A2.Number
FROM Account A1, Account A2
WHERE A1.Balance = A2.Balance AND
        A1.Number > A2.Number;
```

Notes on Queries

Account

Number	Name	Balance	Type
7003001	Jane Smith	1,000,000	Savings
7003003	Alfred Hitchcock	4,400,200	Savings
7003005	Takumi Fujiwara	2,230,000	Checking
7003007	Brian Mills	1,200,000	Savings
7003009	Jason Bourn	3,025,000	Checking

MyTable

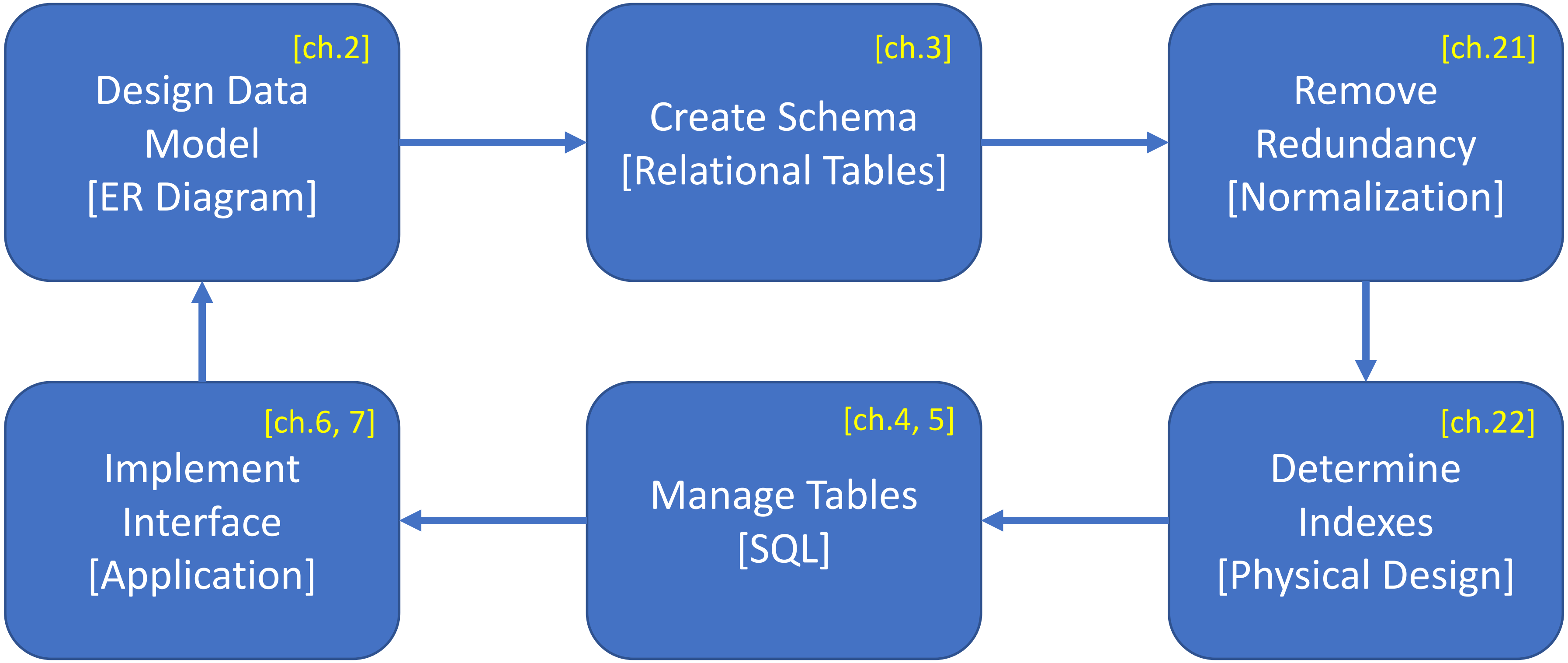
Owner	Amount
Takumi Fujiwara	2,230,000
Jason Bourn	3,025,000

- Naming attributes and query result

  SELECT Name As Owner, Balance as Amount INTO MyTable
  FROM Account
  WHERE Type = "Checking";

- MyTable can be used as a table in subsequent queries
- Remember to delete the temporary tables!

Typical Database Lifecycle



For Next Week

- Review – Quiz on the material
  - Ch. 1
  - Ch. 3 to 3.3
  - Ch. 5 to 5.2
- Reading assignments
  - Ch. 4 to 4.2
  - Ch. 5.5
- Be sure you understand
  - the basic terminology
  - Basic SQL Queries (SELECT, FROM, WHERE)