Introduction to Databases

Week 1, September 27/29

Instructor: David Maier
maier@cs.pdx.edu
Guest Lecturer: Patrick Leyshock
TA: Jim Miller
jgm2@pdx.edu

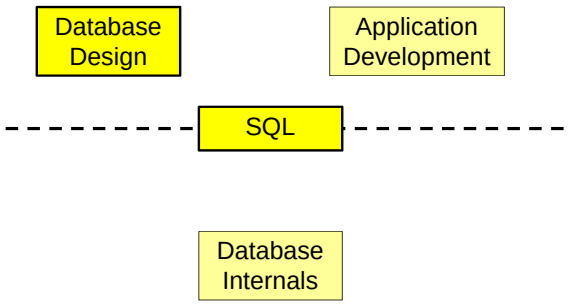
Class e-mail list:
cs486-list@cs.pdx.edu

In order to join the mail list, please visit the following web page and register:
https://mailhost.cecs.pdx.edu/mailman/listinfo/cs486-list
Please give at least your last name and initial, in addition to your email.

Why is This Course in the Curriculum?

- PSU has traditionally had multiple DB profs
- It teaches valued job skills
- It integrates CS concepts
  Languages, data structures, concurrency
- The (digital) world runs on data
- Example of the practical power (query optimizers) of an underlying theory (relational algebra)

Where Does This Course Focus



Class web page

Syllabus available at:
www.cs.pdx.edu/~maier/486

Contains complete class schedule including reading assignments, HW assignments, suggested answers for completed assignments, handouts for lectures, and so forth.

New information appears frequently, so **reload** the page. Handouts of slides will be posted on the web page sometime before class – hopefully a day ahead. (One set per week)

General structure of the class and the grading is set but the details may be modified, if necessary.

Overview of the Syllabus

- Seven Assignments, due Wednesdays (32% or 30%):
  Seven weekly assignments, most worth 4% of your grade.
  Work by yourself or with a partner. Must be your own work.
- Seven Quizzes on Mondays(18% or 12%):
  Each quiz (except for the one that is dropped) counts for 3% (468) or 2% (586) of your grade. In class, almost every Monday.
  Work by yourself.
  NO MAKEUPS FOR QUIZZES!
- Project (8%) For 586 students
  Design and implement a database
- First Exam (25%) OPEN BOOK and NOTES:
  In class; work by yourself. Ask questions only of the instructor or exam monitor.
- Second Exam (25%) OPEN BOOK and NOTES:
  During finals week, work by yourself. Ask questions only of the instructor or exam monitor.

Course Text

Database Systems: The Complete Book

Second Edition

By Garcia-Molina, Ullman, Widom, Pearson, 2009, ISBN 0-13-187325-3.

- Should be able to get used copies
- Make sure it's the correct edition

Also useful to have a SQL reference

Okay to do the reading after topics are covered in class (but should do it before the next quiz)

Academic Integrity

You are responsible for knowing the PSU Academic Integrity Policy

We have no sense of humor about violations of it.

Communication Mechanisms

- Communication from students:
  - E-mail to instructor, TA, class mail list
  - Ask questions in class
  - Ask questions after class
- Communication to students:
  - Course web page.
  - Questions with answer (deemed of general interest) are sent to the cs486-list@cs.pdx.edu e-mail list.
- In person and telephone meetings by request.
  - My office hours: Monday, 1p until at least 2p
  - TA office hours: See course page

Class Courtesy

Please …

- Be prompt
- Cell phones off
  Prof. gets to answer any incoming calls …
- No headphones or earbuds
- One person talking at a time (except during group exercises)

What is computer science?

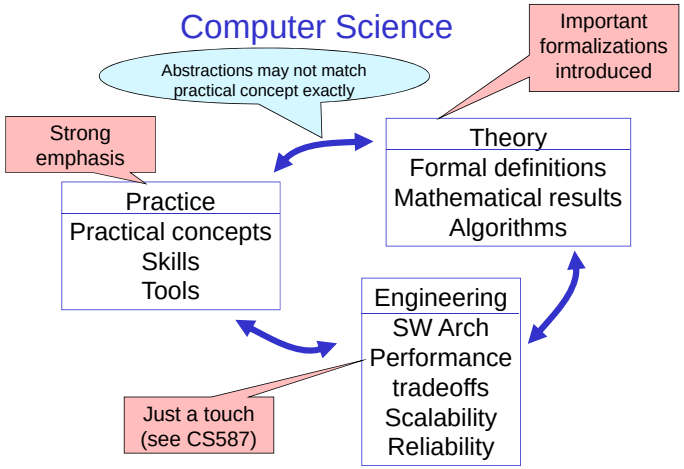
All computer science students must learn to integrate theory and practice, to recognize the importance of abstraction, and to appreciate the value of good engineering design.

Final Report of the Joint ACM/IEEE-CS Task Force on Computing Curricula 2001 for Computer Science - a joint undertaking of the Computer Society of the Institute for Electrical and Electronic Engineers (IEEE-CS) and the Association for Computing Machinery (ACM).

This volume outlines a set of recommendations for undergraduate programs in computer science.

http://www.computer.org/education/cc2001/final/index.htm

Computer Science



Practice and Theory

Practice

- Tables, columns, rows, keys
- SQL
- Application structure
- Logical & physical database design
- Transactions
- Security

Theory

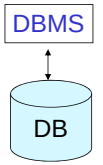
- Relational model: relations, attributes, tuples
- Relational algebra, equivalences
- Functional dependencies, normalization
- Schedules, serializability

Engineering

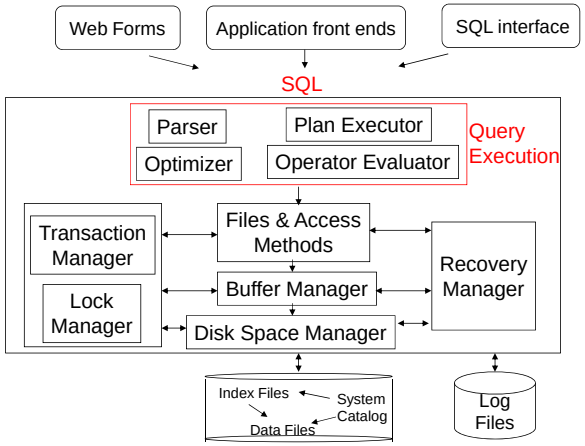
- Basic system structure
- Storage and Indexing
- Query evaluation (operators, optimization)

What's a Database?

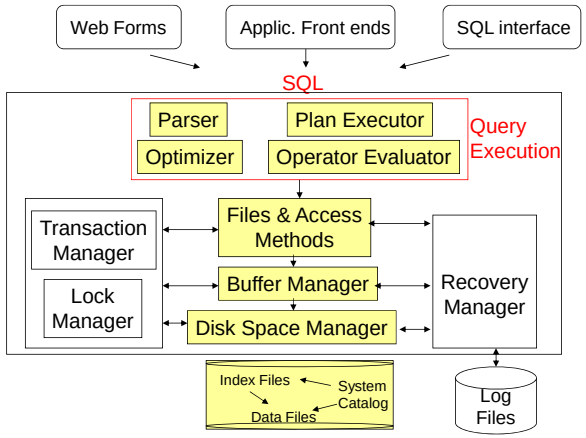
- database (DB) - a collection of persistent data
  Persistent: Lifetime not bound to a process
- database management system (DBMS) - a software system that supports the definition, population, and query of a database.



Database Architecture



Query Processing! (shown in yellow)



This Week's Lectures

- Introduce:
  - Database terminology
  - Difference between schema and data
  - Keys and foreign keys
  - SQL query language

  from a mostly practical point of view (this week)!

Introduction to Relational Databases

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Imagine that this table (or relation) has been defined to help keep track of bank accounts.

Table Structure

The *name* of the table

The name of the *columns* (*attributes*)

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Table Schema

The *schema* for the table

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

The schema sets the structure of the table. You can think of the schema as the *definition* of the table. (Note, the schema specifies more information than what is shown.)

Table Rows

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Each entry in the table is called a ***row*** (***tuple***).
Sometimes an entry in the table is called a record.

Table Instance

An *instance* of the table...

the current contents or data in the table.

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Another Table Instance

Another *instance* of the table
(two rows added, one (103) deleted)

Account

Number	Owner	Balance	Type
101	J. Smith	1,000.00	checking
102	W. Wei	2,000.00	checking
104	M. Jones	1,000.00	checking
105	H. Martin	10,000.00	checking
107	W. Yu	7,500.00	savings
109	R. Jones	432.55	checking

new

Intension vs. Extension

The *intension* of the table

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

The *extension* of the table. Also called the *extent*.

"Size" of a Table

Degree or arity of a table is the number of columns

Degree of this relation (or table) is 4 because there are 4 attributes

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Cardinality of this instance is 5 (because there are 5 rows)

Cardinality of a table = the number of rows in the current instance

Database (One or More Tables)

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Deposit

Account	Transaction-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

Check

Account	Check-number	Date	Amount
101	924	10/23/00	125.00
101	925	10/24/00	23.98

Table Keys

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Deposit

Account	Transaction-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

Check

Account	Check-number	Date	Amount
101	924	10/23/98	125.00
101	925	10/24/98	23.98

Each table has a key…. where the values must be unique.

Table Keys (cont.)

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Deposit

Account	Transaction-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

Check

Account	Check-number	Date	Amount
101	924	10/23/98	125.00
101	925	10/24/98	23.98

Key may consist of one column or two (or more) columns.

Connections between Tables

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

Deposit	Account	Transaction-id	Date	Amount
	102	1	10/22/00	500.00
	102	2	10/29/00	200.00
	104	3	10/29/00	1000.00
	105	4	11/02/00	10,000.00
	106	5	12/05/00	555.00

Is this legal?

If not, how do we prevent it from happening?

Foreign Key

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

Deposit	Account	Transaction-id	Date	Amount
	102	1	10/22/00	500.00
	102	2	10/29/00	200.00
	104	3	10/29/00	1000.00
	105	4	11/02/00	10,000.00
	106	5	12/05/00	555.00

We say that Deposit.Account is a *foreign key* that *references* Account.Number. If the DBMS enforces this constraint, we have ***referential integrity***.

Foreign Keys (cont.)

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

Check	Account	Check-number	Date	Amount
	101	924	10/23/98	125.00
	101	925	10/24/98	23.98

Are there any foreign keys in the Check table?

Yes, Check.Account is a foreign key that references Account.Number.

Foreign keys might or might not be part of the key key for the referring table

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

foreign key

Deposit	Account	key Transaction-id	Date	Amount
	102	1	10/22/00	500.00
	102	2	10/29/00	200.00
	104	3	10/29/00	1000.00
	105	4	110/2/00	10,000.00

Deposit.Account is not part of key for Deposit.

foreign key

Check	Account	key Check-number	Date	Amount
	101	924	10/23/98	125.00
	101	925	10/24/98	23.98

Check.Account is part of key for Check.

Keys for a Table

Consider the following sample data from a table:

1	Jones	28	$50,000.00

Can you tell what the key for this table is?

Keys for a Table

Consider the following sample data from a table:

1	Jones	28	$50,000
2	Smith	28	$60,000

Can you tell what the key for this table is? (Or what it isn't?)

Keys for a Table

One possibility:

Person table with Id as the key

Id	Name	Age	Salary
1	Jones	28	$50,000
2	Smith	28	$60,000

Use underline to indicate key columns

Keys, Table Names, Column Names tell us something about the meaning of a table

Another possibility:

Sales Table, by client company, per day

Salesperson	Customer	Day	Volume
1	Jones	28	$50,000
2	Smith	28	$60,000

Database Domains for Columns

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	...			

For every column of every table, the schema specifies allowable values. For example,

Number must be a 3-digit number
Owner must be a 30-character string
Type must be "checking" or "savings"

The set of allowable values for an column is called the domain of the column.

Specification of a Relational Schema

- Select the tables, with a name for each table.
- Select columns for each table and give the domain for each column.
- Specify the key(s) for each table.
- Specify all appropriate foreign keys.

There can be more than one key for a table.

Another Example Database
(Keys are underlined. Each table has one key.)

Teacher (Number, Name, Office, E-mail)

Course (Number, Name, Description)

Class-Offering (Quarter, Course, Section, Teacher, TimeDays)

Student (Number, Name, Major, Advisor)

Completed (Student, Quarter, Course, Section, Grade)

Do Name and Number mean the same thing everywhere?

Example Database (cont.)
(with some foreign keys shown informally, with arrows)

Teacher (Number, Name, Office, E-mail)

Course (Number, Name, Description)

Taught-By (Quarter, Course, Section, Teacher, TimeDays)

Student (Number, Name, Major, Advisor)

Completed (Student, Quarter, Course, Section, Grade)

What foreign keys are present in the Completed table?

Example Database (cont.)

(with foreign keys shown informally, with arrows)

Teacher (Number, Name, Office, E-mail)

Course (Number, Name, Description)

Taught-By (Quarter, Course, Section, Teacher, TimeDays)

Student (Number, Name, Major, Advisor)

Completed (Student, Quarter, Course, Section, Grade)

Foreign keys in the Completed table are shown above.

New Example

Recipe (id, name, servings, prep-time)

Ingredient (id, name)

Possible Keys

Recipe (id, name, servings, prep-time)

Ingredient (id, name)

But…one recipe uses lots of ingredients and one ingredient can be used in lots of recipes..

What can we do?

Possible tables

Recipe (id, name, servings, prep-time, ingred-id)

Will this work?

Ingredient (id, name, recipe-id)

Will this work?

Should we do both of these?

Possible tables

Recipe (id, name, servings, prep-time)

Used-In (recipe-id, ingredient-id)

What's the key for this table?

Ingredient (id, name)

Another Possibility

Recipe (id, name, servings, prep-time)

Used-In (recipe-id, ingredient-id, quantity)

What's the key for this table?

Ingredient (id, name)

In general, we always need to introduce a new table for a many-to-many relationship

Table Design Exercise

- Create some table schemas to represent information about food carts.
- After you have your tables, I will distribute some information, and we'll see how well it fits your tables.

SQL – the language we use to talk to the Database Management System

SQL can be used for lots of purposes including:

To define tables -

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

To query the database –

```
SELECT    *
FROM      Account
WHERE     Type = "checking ";
```

SQL (cont.)

To insert rows into a table:

```
INSERT INTO Account
VALUES (106, " H. Martinez ", 10,000, " savings ");
```

and so forth

SQL is a standard…
- and there have been a series of SQL standards: 1986, 1989, 1992 (SQL2), 1999 (SQL3), …

But DBMS products differ in how much of the standard they support … and how many extra features they have.

Database Schema (first version)

ACCOUNT	Number	Owner	Balance	Type

DEPOSIT	AcctNo	Transaction-id	Date	Amount

CHECK	AcctNo	Check-number	Date	Amount

Database Schema (second version)

What are the foreign keys here?

ACCOUNT	Number	Owner	Balance	Type

DEPOSIT	AcctNo	Transaction-id	Date	Amount

CHECK	AcctNo	Check-number	Date	Amount

ATMWITHDRAWAL	TransactionID	CustId	AcctNo	Amount	WithdrawDate

CUSTOMER	ID	Name	Phone	Address

ATMWithdrawal table

TransactionID	CustId	AcctNo	Amount	WithdrawDate
1	1	102	$25.00	11/01/2000 9:45:00
2	1	102	$150.00	11/10/2000 13:15:00
3	2	101	$40.00	11/01/2000 10:05:00
4	2	100	$40.00	11/01/2000 10:07:00
5	2	100	$200.00	11/08/2000 14:14:00

```
SELECT AcctNo, Amount
FROM   ATMWithdrawal
WHERE  Amount < 50;
```

ATMWithdrawal table

TransactionID	CustId	AcctNo	Amount	WithdrawDate
1	1	102	$25.00	11/01/2000 9:45:00
2	1	102	$150.00	11/10/2000 13:15:00
3	2	101	$40.00	11/01/2000 10:05:00
4	2	100	$40.00	11/01/2000 10:07:00
5	2	100	$200.00	11/08/2000 14:14:00

```
SELECT AcctNo, Amount
FROM   ATMWithdrawal
WHERE  Amount < 50;
```

This is the WHERE clause.
The WHERE clause is evaluated for each row in the table.

ATMWithdrawal table

TransactionID	CustId	AcctNo	Amount	WithdrawDate
1	1	102	$25.00	11/01/2000 9:45:00
2	1	102	$150.00	11/10/2000 13:15:00
3	2	101	$40.00	11/01/2000 10:05:00
4	2	100	$40.00	11/01/2000 10:07:00
5	2	100	$200.00	11/08/2000 14:14:00

Is the amount field of this row less than $50? YES!

Amount < 50

Intermediate Query Answer table

TransactionID	CustId	AcctNo	Amount	WithdrawDate
1	1	102	$25.00	11/01/2000 9:45:00

ATMWithdrawal table

TransactionID	CustId	AcctNo	Amount	WithdrawDate
1	1	102	$25.00	11/01/2000 9:45:00
2	1	102	$150.00	11/10/2000 13:15:00
3	2	101	$40.00	11/01/2000 10:05:00
4	2	100	$40.00	11/01/2000 10:07:00
5	2	100	$200.00	11/08/2000 14:14:00

Is the amount field of this record less than $50? NO!

Amount < 50

Ignore this record!

Intermediate Query Answer table

TransactionID	CustId	AcctNo	Amount	WithdrawDate
1	1	102	$25.00	11/01/2000 9:45:00

ATMWithdrawal table

TransactionID	CustId	AcctNo	Amount	WithdrawDate
1	1	102	$25.00	11/01/2000 9:45:00
2	1	102	$150.00	11/10/2000 13:15:00
3	2	101	$40.00	11/01/2000 10:05:00
4	2	100	$40.00	11/01/2000 10:07:00
5	2	100	$200.00	11/08/2000 14:14:00

Is the amount field of this record less than $50? YES!

Amount < 50

Intermediate Query Answer table

TransactionID	CustId	AcctNo	Amount	WithdrawDate
1	1	102	$25.00	11/01/2000 9:45:00
3	2	101	$40.00	11/01/2000 10:05:00

ATMWithdrawal table

TransactionID	CustId	AcctNo	Amount	WithdrawDate
1	1	102	$25.00	11/01/2000 9:45:00
2	1	102	$150.00	11/10/2000 13:15:00
3	2	101	$40.00	11/01/2000 10:05:00
4	2	100	$40.00	11/01/2000 10:07:00
5	2	100	$200.00	11/08/2000 14:14:00

Is the amount field of this record less than $50? YES!

Amount < 50

Intermediate Query Answer table

TransactionID	CustId	AcctNo	Amount	WithdrawDate
1	1	102	$25.00	11/01/2000 9:45:00
3	2	101	$40.00	11/01/2000 10:05:00
4	2	100	$40.00	11/01/2000 10:07:00

ATMWithdrawal table

TransactionID	CustId	AcctNo	Amount	WithdrawDate
1	1	102	$25.00	11/01/2000 9:45:00
2	1	102	$150.00	11/10/2000 13:15:00
3	2	101	$40.00	11/01/2000 10:05:00
4	2	100	$40.00	11/01/2000 10:07:00
5	2	100	$200.00	11/08/2000 14:14:00

Is the amount field of this record less than $50? NO!

Amount < 50

Ignore this record!

Intermediate Query Answer table

TransactionID	CustId	AcctNo	Amount	WithdrawDate
1	1	102	$25.00	11/01/2000 9:45:00
3	2	101	$40.00	11/01/2000 10:05:00
4	2	100	$40.00	11/01/2000 10:07:00

Intermediate Query Answer table

TransactionID	CustId	AcctNo	Amount	WithdrawDate
1	1	102	$25.00	11/01/2000 9:45:00
3	2	101	$40.00	11/01/2000 10:05:00
4	2	100	$40.00	11/01/2000 10:07:00

```
SELECT AcctNo, Amount
FROM   ATMWithdrawal
WHERE  Amount < 50;
```

Consider the attributes listed in the SELECT clause.
Throw away attributes that are not listed.
Thus the final query answer is:

Final Query Answer table

AcctNo	Amount
102	$25.00
101	$40.00
100	$40.00

Another SQL Query (using one table)

ATMWithdrawal

TransactionId	CustId	AcctNo	Amount	WithdrawDate
1	1	102	$25.00	11/01/00 9:45:00 AM
2	1	102	$150.00	11/10/00 1:15:00 PM
3	2	101	$40.00	11/01/00 10:05:00 AM
4	2	100	$40.00	11/01/00 10:07:00 AM
5	2	100	$200.00	11/08/00 2:14:00 PM

```
SELECT *
FROM   ATMWithdrawal
WHERE TransactionId = 3;
```

The five rows are considered, one by one, to see if TransactionId = 3 (to see if the WHERE clause evaluates to true).

Note: "*" in SELECT clause means "all attributes"

```
SELECT *
FROM   ATMWithdrawal
WHERE TransactionId = 3;
```

ATMWithdrawal

TransactionId	CustId	AcctNo	Amount	WithdrawDate
1	1	102	$25.00	11/01/00 9:45:00 AM
2	1	102	$150.00	11/10/00 1:15:00 PM
3	2	101	$40.00	11/01/00 10:05:00 AM
4	2	100	$40.00	11/01/00 10:07:00 AM
5	2	100	$200.00	11/08/00 2:14:00 PM

Query Answer is:

TransactionId	CustId	AcctNo	Amount	WithdrawDate
3	2	101	$40.00	11/01/00 10:05:00 AM

Example Query

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

```
SELECT    *
FROM      Account
WHERE     Type = "checking";
```


Example Query with Answer

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

```
SELECT    *
FROM      Account
WHERE     Type = "checking";
```

	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

Another Query

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

```
SELECT    *
FROM      Account
WHERE     Type = "savings";
```

…with its Query Answer

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

```
SELECT    *
FROM      Account
WHERE     Type = "savings";
```

	Number	Owner	Balance	Type
	103	J. Smith	5000.00	savings

Yet Another Query (what's different?)

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

```
SELECT    Owner
FROM      Account
WHERE     Type = "checking";
```


…the query answer

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

```
SELECT    Owner
FROM      Account
WHERE     Type = "checking";
```

	Owner
	J. Smith
	W. Wei
	M. Jones
	H. Martin

Example (Useless) Query

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

```
SELECT    *
FROM      Account
WHERE     Type = "checking" AND
          Type = "savings";
```

Example (Useless) Query with Answer

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

```
SELECT    *
FROM      Account
WHERE     Type = "checking" AND
          Type = "savings";
```

Query answer is empty. But that's allowed.

	Number	Owner	Balance	Type

So… why is this a "useless" query?

How an SQL query is evaluated

Third, the SELECT clause tells us which columns to keep in the query answer.

```
SELECT    AcctNo, Amount
FROM      ATMWithdrawal
WHERE     Amount < 50;
```

First, the FROM clause tells us the input tables.

Second, the WHERE clause is evaluated for all possible combinations from the input tables.

Quick Exercise

Using Account table

ACCOUNT	Number	Owner	Balance	Type

Write SQL queries that determine

1. What types of accounts does Smith have?
2. What is the account number of savings accounts with less than $1000

And find the answers to your queries

Is Query 2. a useless query?

SQL query using two tables

```
SELECT    A.Owner, A.Balance
FROM      Account A, Deposit D
WHERE     D.AcctNo = A.Number and A.Balance > 1000;
```

How does this work?
Which rows, from which tables,
are evaluated in the WHERE clause?

SQL query using two tables

```
SELECT    A.Owner, A.Balance
FROM      Account A, Deposit D
WHERE     D.AcctNo = A.Number and A.Balance > 1000;
```

"A" is a correlation name for Account
and
"D" is a correlation name for Deposit.

Correlation names are like local variables – they hold one tuple or row from the corresponding table.
You choose correlation names when you write the query.

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Deposit

AcctNo	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

```
SELECT    A.Owner, A.Balance
FROM      Account A, Deposit D
WHERE     D.AcctNo = A.Number and A.Balance > 1000;
```

We must check every combination of one row from Account with one row from Deposit.

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Deposit

AcctNo	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

No! Throw it away.

```
WHERE     D.AcctNo = A.Number and A.Balance > 1000;
```

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount

notice the columns

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Deposit

AcctNo	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

No! Throw it away.

```
WHERE     D.AcctNo = A.Number and A.Balance > 1000;
```

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Deposit

AcctNo	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

No! Throw it away.

WHERE D.AcctNo = A.Number and A.Balance > 1000;

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Deposit

AcctNo	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

No! Throw it away.

WHERE D.AcctNo = A.Number and A.Balance > 1000;

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Deposit

AcctNo	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

Yes! Place in query answer.

WHERE D.AcctNo = A.Number and A.Balance > 1000;

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Deposit

AcctNo	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

Yes! Place in query answer.

WHERE D.AcctNo = A.Number and A.Balance > 1000;

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

No! Throw it away.

Deposit

AcctNo	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

WHERE D.AcctNo = A.Number and A.Balance > 1000;

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

No! Throw it away.

Deposit

AcctNo	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

WHERE D.AcctNo = A.Number and A.Balance > 1000;

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

All combinations fail! →

Deposit

AcctNo	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

WHERE D.AcctNo = A.Number and A.Balance > 1000;

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

No! Throw it away.

Deposit

AcctNo	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

WHERE D.AcctNo = A.Number and A.Balance > 1000;

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Deposit

AcctNo	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

No! Throw it away.

WHERE D.AcctNo = A.Number and A.Balance > 1000;

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Deposit

AcctNo	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

No! Throw it away. Why?

WHERE D.AcctNo = A.Number and A.Balance > 1000;

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Deposit

AcctNo	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

No! Throw it away.

WHERE D.AcctNo = A.Number and A.Balance > 1000;

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Deposit

AcctNo	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

The first three fail.

WHERE D.AcctNo = A.Number and A.Balance > 1000;

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Yes! Place in query answer.

Deposit

AcctNo	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

```
WHERE    D.AcctNo = A.Number and A.Balance > 1000;
```

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00
105	H. Martin	10,000.00	checking	105	4	11/02/00	10,000.00

Intermediate result
(after processing the FROM & WHERE clauses)

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00
105	H. Martin	10,000.00	checking	105	4	11/02/00	10,000.00

Process the SELECT

```
SELECT    A.Owner, A.Balance
FROM      Account A, Deposit D
WHERE     D.AcctNo = A.Number and A.Balance > 1000;
```

Final query answer:
(notice that W. Wei appears twice)

Owner	Balance
W. Wei	2000.00
W. Wei	2000.00
H. Martin	10,000.00

Intermediate result
(after processing the FROM & WHERE clauses)

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00
105	H. Martin	10,000.00	checking	105	4	11/02/00	10,000.00

Process the SELECT

```
SELECT    DISTINCT  A.Owner, A.Balance
FROM      Account A, Deposit D
WHERE     D.AcctNo = A.Number and A.Balance > 1000;
```

If we use the word DISTINCT, then duplicates are removed from the query answer.
W. Wei only appears once.

Owner	Balance
W. Wei	2000.00
H. Martin	10,000.00

Another SQL query using two tables

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

Deposit	AcctNo	Transaction-id	Date	Amount
	102	1	10/22/00	500.00
	102	2	10/29/00	200.00
	104	3	10/29/00	1000.00
	105	4	11/02/00	10,000.00

```
SELECT    A.Number, A.Owner
FROM      Account AS A, Deposit AS D
WHERE     A.Number = D.AcctNo and D.Amount > 300;
```

How many rows will be in the query answer?
How many columns will be in the query answer?

SQL query using two tables(cont.)

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

Deposit	AcctNo	Transaction-id	Date	Amount
	102	1	10/22/00	500.00
	102	2	10/29/00	200.00
	104	3	10/29/00	1000.00
	105	4	11/02/00	10,000.00

```
SELECT  A.Number, A.Owner
FROM    Account AS A, Deposit AS D
WHERE   A.Number = D.AcctNo and D.Amount > 300;
```

	Number	Owner
	102	W. Wei
	104	M. Jones
	105	H. Martin

Queries

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

Notice that a query is expressed against the schema.

```
SELECT  Owner
FROM    Account
WHERE   Type = "checking";
```

But the query runs or executes against the instance (the data)

And may give different answers on different instances

	Owner
	J. Smith
	W. Wei
	M. Jones
	H. Martin

Comments on Queries

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

Notice that the answer to a query is always a **table**!
It doesn't always have a name (for the table).
The attribute names are deduced from the input tables (or supplied by the query author). It might or Might not have any rows.

	Owner
	J. Smith
	W. Wei
	M. Jones
	H. Martin

Creating temporary tables using INTO

We can create a name for the query answer:

```
SELECT  Owner INTO temp3
FROM    Account
WHERE   Type = "checking";
```

temp3	Owner
	J. Smith
	W. Wei
	M. Jones
	H. Martin

temp3 can be used as a table in subsequent queries!
REMEMBER TO DELETE YOUR TEMPORARY TABLES!!

Comments on Queries

Because the answer to a relational query is always a **table**

……

we can use the answer from one query as input to another query.

This means that we can create arbitrarily complex queries!

A relational query languages is **closed** if it has this property.