

Introduction to Databases

Lecture 1, January 6, 2009

Instructor: Lois Delcambre

imd@cs.pdx.edu

Grader: TBD

Class e-mail list (for both 386 and 586 students):

cs386-586@cecs.pdx.edu

URGENT! visit the following web page and register:

<https://mailhost.cecs.pdx.edu/mailman/listinfo/cs386-586>

Register multiple times if you want to use multiple e-mail addresses.

I use this membership list to find your e-mail address – so list your name so that I can recognize it.

Class web page (single page for CS386 and CS586)

Syllabus available at:

www.cs.pdx.edu/~lmd/cs386

Will contain complete class schedule: reading assignments, 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 – usually at least 24 hours ahead.

General structure of the class and the grading is shown but
the details aren't fixed yet. I hope to have the grade
structure and the order of topics fixed by the 2nd class.

Approximate Structure of the Class

- **Weekly assignments (35%)**; work in teams of 1 or 2
- **Test 1 (27.5%) OPEN BOOK (closed notes)**:
In class; work by yourself.
- **Test 2 (27.5%) OPEN BOOK (closed notes)**:
In class, work by yourself.
- **Undergraduates/CS386: exercises** to be done in-class – worth **10% of your grade**. Evidence of having worked the exercises – due at the end of the term. (Graduate students are encouraged to do the exercises, too. But they don't need to turn them in and they won't receive any credit toward their grade.)
- **Graduates/CS586**: Prepare and present a **tutorial** on a DB-related project. This is worth **10% of your grade**. Work in teams of 2 or 3.

Communication Mechanisms

- **Communication from students:**
 - E-mail to instructors, graders, class mail list
 - Ask questions in class
 - Ask questions before and after class
- **Communication to students:**
 - Model answers sometimes posted on the web page.
 - Questions with answer (deemed of general interest) are sent to the cs386-586@cs.pdx.edu e-mail list.
- **In person and telephone meetings by request.**
- **Office hours – before and after class**

Introduction to Databases

Data – Digital Information

- Digital Information is **precious**
- Modern business, culture and society **could not exist** in its present form without digital information.
- Think about how often you use digital information in **your life**.
- If most modern businesses lost their current and backup information they would be candidates for **bankruptcy**. (We need to protect it!)
- **Digital information can be combined and summarized in many ways – to serve many different purposes.**

Digital Information may or may not be Regularly Structured

Regularly structured data: set the structure once and then have many, many instances (records) that use that structure.

Examples of regularly structured data:

- Employee, payroll, bank account
- Data captured on a web form

Examples of unstructured (or loosely structured data or semi-structured) data:

- Documents, video, audio, images, ...

Consider a music file

- Is it regular structured?
- Do we attach regularly structured data to it?
(What are some examples?)
- This is very common – to have some of the data of interest with regular structure. The attributes of interest have a name.

Structured vs. Unstructured Data

- In this class we study relational database management system (**DBMS**); they are designed to store, manage, and retrieve regularly structured data. We use **SQL** to manage and retrieve data from a database (DB).
- In contrast, unstructured data (e.g., in documents) is managed by information retrieval systems - including search engines working over the web – that look for certain words or characters in text; in any text and rank results.

Characteristics of Databases

Data is vast - Much too large to fit in memory

- Library of congress: 20 terabytes
- Photos uploaded to Facebook each month: 20 terabytes
- Amazon.com: 42 terabytes
- Data processed by Google's servers each hour: 1 Petabyte

1 terabyte \approx 1,000,000,000,000 bytes

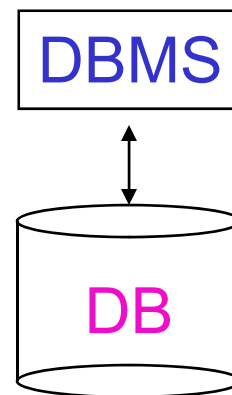
1 petabyte \approx 1,000,000,000,000,000 bytes

Why study databases?

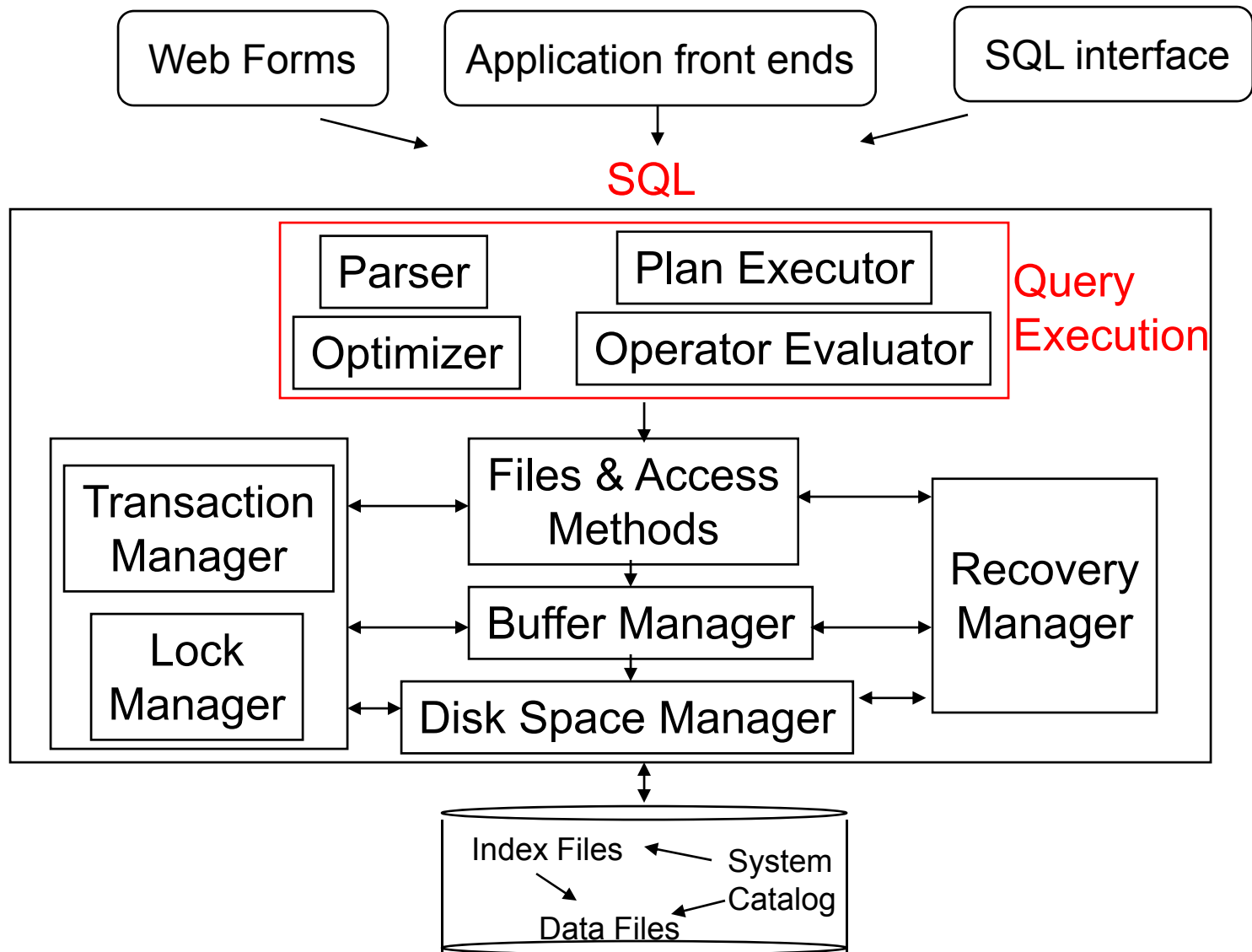
- Because the database field has made a number of contributions to basic computer science:
 - because of its focus on data...and disks...
 - because of the formalization of concepts
- Because DBMS software is highly successful as a commercial technology (Oracle, Informix, MS Access...)
- Because DB research is highly active

What's a DB?

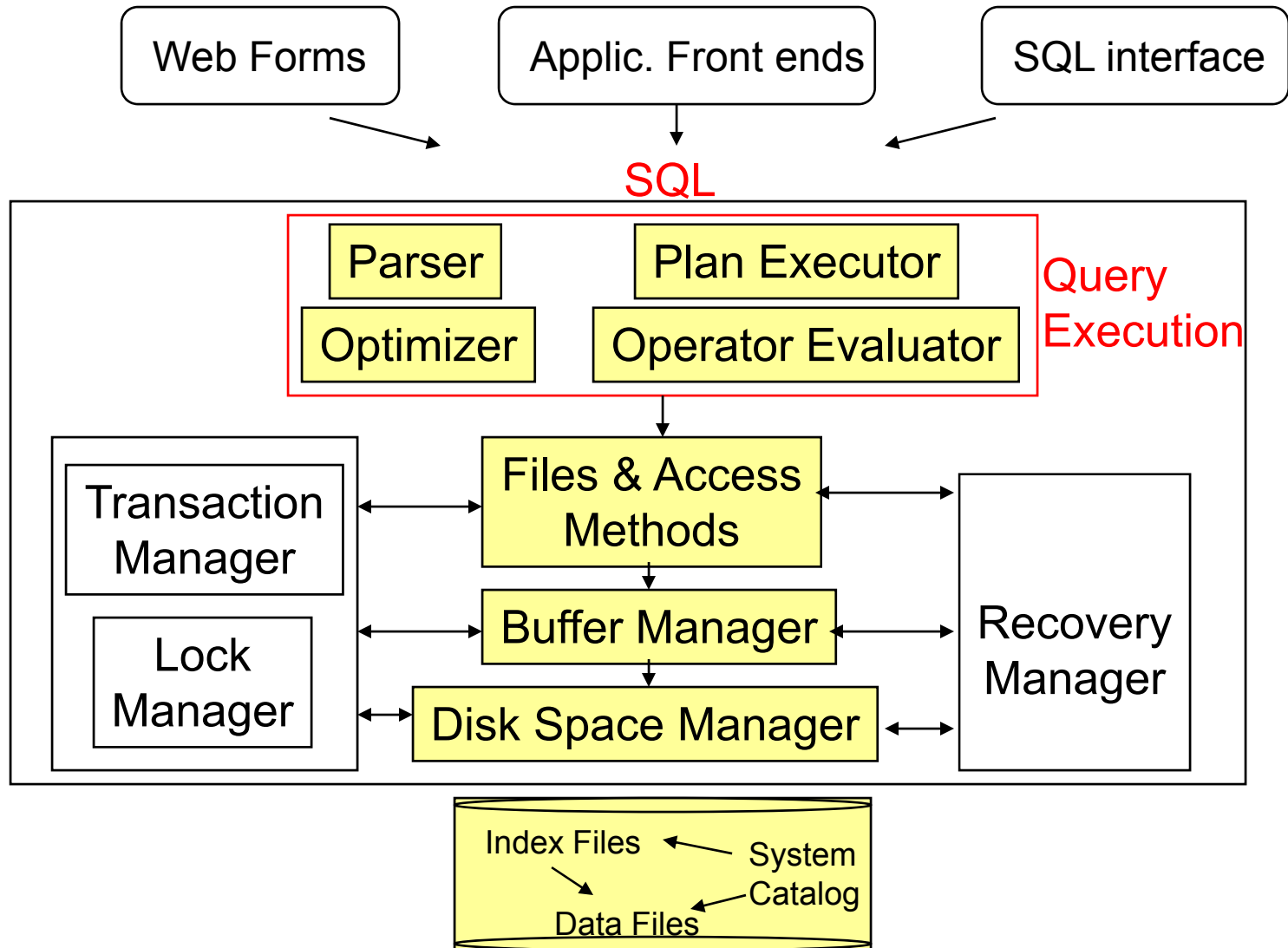
- **database** (DB)- a collection of persistent data
- **database management system** (DBMS) - a software system that supports the definition, population, and query of a database.



Database Architecture (Figure 1.3, p. 20)



Query Processing! (shown in yellow)



What is computer science?

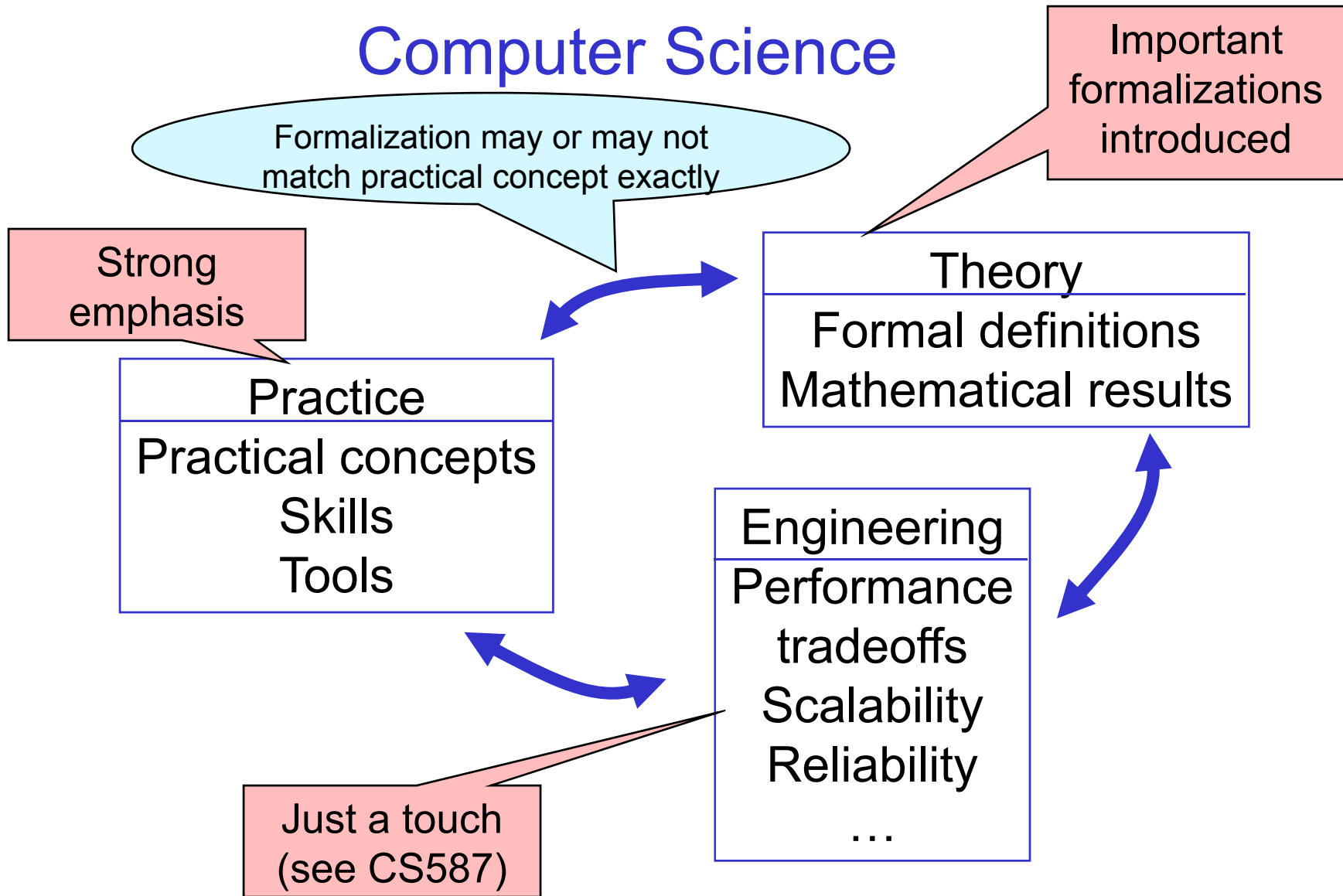
All computer science students must learn to integrate **theory** and **practice**, to recognize the importance of **abstraction**, and to appreciate the value of **good engineering design**.

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



Week 1 Lecture Material

- Introduce:
 - Database terminology
 - Difference between schema and data
 - SQL query language
 - Relational data model

from a practical point of view (only, for tonight);
we'll look at the formal definition soon!

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 has been defined to help keep track of bank accounts.

Introduction to Relational Databases

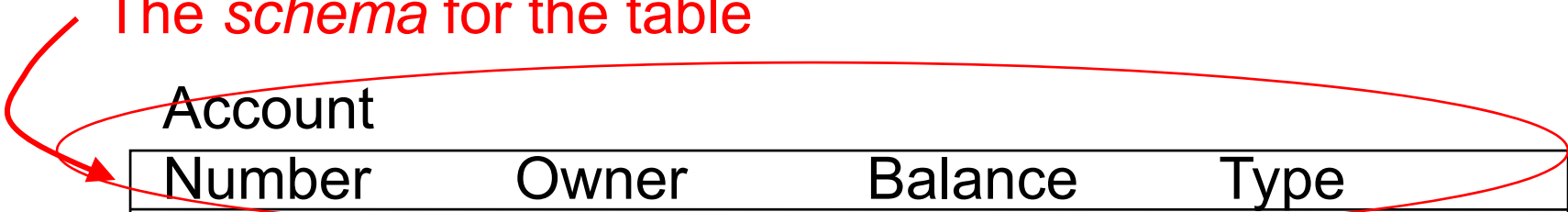
The *name* of the table

The name of the *attributes* (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

Introduction to Relational Databases

The *schema* for the table



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

Terminology for 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

Each entry in the table is called a **row** or a **tuple**.
Sometimes an entry in the table is called a record.
The **instance** is the current set of rows (or tuples).

Introduction to Relational Databases

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

Introduction to Relational Databases

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

Terminology for Relational Databases

The *intension* of the table

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 current instance is the *extension* of the table.

The extension is also called the *extent* of the table.

Terminology for Relational Databases

Degree or *arity* of a table is the number of attributes

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

Relational Database Example (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	11/2/00	10,000.00

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

Relational Database Example (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	11/2/00	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.

Relational Database Example (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	11/2/00	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 attribute or two (or more) attributes.

Relational Database Example (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	11/2/00	10,000.00
	106	5	12/5/00	555.00

Is this legal?

If not, how do we prevent it from happening?

Relational Database Example (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	11/2/00	10,000.00
	106	5	12/5/00	555.00

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

Relational Database Example (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 may or may not be part of the key 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

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	11/2/00	10,000.00

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

Deposit.Account
is **not** part
of key for
Deposit.

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?

Keys for a Table

One possibility:

Person Table with Id as the key

<u>Id</u>	Name	Age	Salary
1	Jones	28	\$50,000
2	Smith	28	\$60,000

Keys, Table Names, Attribute Names

Tell us what the table is

Another possibility:

Sales Commission Table, by client company, per day

<u>Salesperson</u>	<u>Company</u>	<u>Day</u>	Commission
1	Jones	28	\$50,000
2	Smith	28	\$60,000

Relational Database Domains for Attributes

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

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

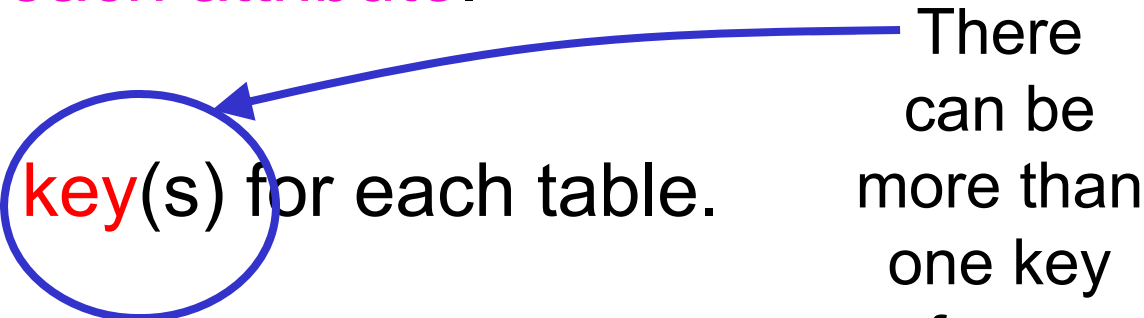
Number must be a 3-digit number

Owner must be a 30-character string

Type must be “checking” or “savings”

The allowable values for an attribute is called the **domain** of the attribute.

Specification of a Relational Schema

- Select the tables, with a **name for each table**.
- Select **attributes for each table** and give the **domain for each attribute**.
- Specify the **key(s)** for each table.  There can be more than one key for a table.
- Specify all appropriate **foreign keys**.

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, Course, Quarter, Section, Grade)

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, Course, Quarter, 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, Course, Quarter, Section, Grade)

Foreign keys in the Completed table are shown above.

What are the limitations of this schema?

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

Course (Number, Name, Description)

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

Student (Number, Name, Major, Advisor)

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

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 ", 10000, " 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	<u>Number</u>	Owner	Balance	Type
---------	---------------	-------	---------	------

DEPOSIT	Account	<u>Transaction-id</u>	Date	Amount
---------	---------	-----------------------	------	--------

CHECK	<u>Account</u>	<u>Check-number</u>	Date	Amount
-------	----------------	---------------------	------	--------

Database Schema (second version) - Allow multiple owners for an account

What are the foreign keys here?

ACCOUNT	<u>Number</u>	Owner	Balance	Type
---------	---------------	------------------	---------	------

DEPOSIT	Account	<u>ID</u>	Date	Amount
---------	---------	-----------	------	--------

Change attribute name to "ID" to be more consistent.

CHECK	<u>Account</u>	<u>Check-number</u>	Date	Amount
-------	----------------	---------------------	------	--------

ATMWITHDRAWAL	<u>ID</u>	CustID	AcctNo	Amount	WithdrawDate
---------------	-----------	--------	--------	--------	--------------

CUSTOMER	<u>ID</u>	Name	Phone	Address
----------	-----------	------	-------	---------

ACCT-OWNER	<u>AccountID</u>	<u>CustomerID</u>
------------	------------------	-------------------

ATMWithdrawal table

TransactionID	CustId	AcctNo	Amount	WithdrawDate
1	1	102	\$25.00	11/1/2000 9:45:00
2	1	102	\$150.00	11/10/2000 13:15:00
3	2	101	\$40.00	11/1/2000 10:05:00
4	2	100	\$40.00	11/1/2000 10:07:00
5	2	100	\$200.00	11/8/2000 14:14:00

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

ATMWithdrawal table

TransactionID	CustId	AcctNo	Amount	WithdrawDate
1	1	102	\$25.00	11/1/2000 9:45:00
2	1	102	\$150.00	11/10/2000 13:15:00
3	2	101	\$40.00	11/1/2000 10:05:00
4	2	100	\$40.00	11/1/2000 10:07:00
5	2	100	\$200.00	11/8/2000 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	11/1/2000 9:45:00
2	1	102	\$150.00	11/10/2000 13:15:00
3	2	101	\$40.00	11/1/2000 10:05:00
4	2	100	\$40.00	11/1/2000 10:07:00
5	2	100	\$200.00	11/8/2000 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	11/1/2000 9:45:00

ATMWithdrawal table

TransactionID	CustId	AcctNo	Amount	WithdrawDate
1	1	102	\$25.00	11/1/2000 9:45:00
2	1	102	\$150.00	11/10/2000 13:15:00
3	2	101	\$40.00	11/1/2000 10:05:00
4	2	100	\$40.00	11/1/2000 10:07:00
5	2	100	\$200.00	11/8/2000 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	11/1/2000 9:45:00

ATMWithdrawal table

TransactionID	CustId	AcctNo	Amount	WithdrawDate
1	1	102	\$25.00	11/1/2000 9:45:00
2	1	102	\$150.00	11/10/2000 13:15:00
3	2	101	\$40.00	11/1/2000 10:05:00
4	2	100	\$40.00	11/1/2000 10:07:00
5	2	100	\$200.00	11/8/2000 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	11/1/2000 9:45:00
3	2	101	\$40.00	11/1/2000 10:05:00

ATMWithdrawal table

TransactionID	CustId	AcctNo	Amount	WithdrawDate
1	1	102	\$25.00	11/1/2000 9:45:00
2	1	102	\$150.00	11/10/2000 13:15:00
3	2	101	\$40.00	11/1/2000 10:05:00
4	2	100	\$40.00	11/1/2000 10:07:00
5	2	100	\$200.00	11/8/2000 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	11/1/2000 9:45:00
3	2	101	\$40.00	11/1/2000 10:05:00
4	2	100	\$40.00	11/1/2000 10:07:00

ATMWithdrawal table

TransactionID	CustId	AcctNo	Amount	WithdrawDate
1	1	102	\$25.00	11/1/2000 9:45:00
2	1	102	\$150.00	11/10/2000 13:15:00
3	2	101	\$40.00	11/1/2000 10:05:00
4	2	100	\$40.00	11/1/2000 10:07:00
5	2	100	\$200.00	11/8/2000 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	11/1/2000 9:45:00
3	2	101	\$40.00	11/1/2000 10:05:00
4	2	100	\$40.00	11/1/2000 10:07:00

Intermediate Query Answer table

TransactionID	CustId	AcctNo	Amount	WithdrawDate
1	1	102	\$25.00	11/1/2000 9:45:00
3	2	101	\$40.00	11/1/2000 10:05:00
4	2	100	\$40.00	11/1/2000 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	11/1/00 9:45:00 AM
2	1	102	\$150.00	11/10/00 1:15:00 PM
3	2	101	\$40.00	11/1/00 10:05:00 AM
4	2	100	\$40.00	11/1/00 10:07:00 AM
5	2	100	\$200.00	11/8/00 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).

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

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

ATMWithdrawal					
TransactionId	CustId	AcctNo	Amount	WithdrawDate	
1	1	102	\$25.00	11/1/00 9:45:00 AM	
2	1	102	\$150.00	11/10/00 1:15:00 PM	
3	2	101	\$40.00	11/1/00 10:05:00 AM	
4	2	100	\$40.00	11/1/00 10:07:00 AM	
5	2	100	\$200.00	11/8/00 2:14:00 PM	

Query Answer is:

TransactionId	CustId	AcctNo	Amount	WithdrawDate	
3	2	101	\$40.00	11/1/00 10:05:00 AM	

Example Query (first version of schema)

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
Table in DB	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
Query answer	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
Table in DB	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
Query answer	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
Table in DB	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
Query answer	J. Smith
	W. Wei
	M. Jones
	H. Martin

Another 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" AND  
Type = "savings";
```

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" AND
Type = "savings";
```

Query
answer is
empty.

	Number	Owner	Balance	Type
--	--------	-------	---------	------

Queries with Empty Answers

- An empty answer is not an error.
- An empty answer can be very informative. You might find out, for example, that there aren't any accounts with balance over \$90,000 in your bank.
- If a query will **always**, necessarily produce an empty answer, it would be nice if you or the query optimizer could recognize it (and not run it).

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" AND  
Type = "savings";
```

	Number	Owner	Balance	Type
--	--------	-------	---------	------

This query will ALWAYS have an empty answer, no matter what data is in the table instance.

Another query that will always be empty

Student (id, name, age, gender)

```
SELECT *  
FROM Student  
WHERE gender = "m" and gender = "f";
```

This query has the same form as the prior one.
There is NO reason to execute this query.

How an SQL query is evaluated

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

SELECT
FROM
WHERE

AcctNo, Amount
ATMWithdrawal
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.

SQL query using two tables (first version of schema)

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

How does this work?

Which rows, from which tables,
are evaluated in the WHERE clause?

What about this one:

```
SELECT      *
FROM        Account A, Deposit D;
```

SQL query using two tables

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

“A” is a correlation name for **Account**

“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.

You can always use the table name as a correlation name: Account.name, for example.

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	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	11/2/00	10,000.00

```

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

```

We must check every combination of one row from Customer with one row from CheckingAccount!

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	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	11/2/00	10,000.00

No! Throw it away.

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

notice the attributes

Number	Owner	Balance	Type	Account	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	

No! Throw
it away.

Deposit			
Account	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	11/2/00	10,000.00

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

Number	Owner	Balance	Type	Account	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	

No! Throw
it away.

Deposit			
Account	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	11/2/00	10,000.00

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

Number	Owner	Balance	Type	Account	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	

No! Throw
it away.

Deposit			
Account	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	11/2/00	10,000.00

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

Number	Owner	Balance	Type	Account	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			
Account	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	11/2/00	10,000.00

Yes! Place in query answer.

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

Number	Owner	Balance	Type	Account	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	

Yes! Place in query answer.

Deposit			
Account	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	11/2/00	10,000.00

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

Number	Owner	Balance	Type	Account	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			
Account	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	11/2/00	10,000.00

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

Number	Owner	Balance	Type	Account	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			
Account	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	11/2/00	10,000.00

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

Number	Owner	Balance	Type	Account	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			
Account	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	11/2/00	10,000.00

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

Number	Owner	Balance	Type	Account	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			
Account	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	11/2/00	10,000.00

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

Number	Owner	Balance	Type	Account	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			
Account	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	11/2/00	10,000.00

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

Number	Owner	Balance	Type	Account	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. Why?

Deposit			
Account	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	11/2/00	10,000.00

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

Number	Owner	Balance	Type	Account	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			
Account	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	11/2/00	10,000.00

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

Number	Owner	Balance	Type	Account	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			
Account	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	11/2/00	10,000.00

No! The first three fail.

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

Number	Owner	Balance	Type	Account	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			
Account	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	11/2/00	10,000.00

Yes! Place in query answer.

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

Number	Owner	Balance	Type	Account	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	11/2/00	10,000.00

Intermediate result

(after processing the FROM & WHERE clauses)

Number	Owner	Balance	Type	Account	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	11/2/00	10,000.00

Process the SELECT

SELECT A.Owner, A.Balance
FROM Account A, Deposit D
WHERE D.Account = 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	Account	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	11/2/00	10,000.00

Process the SELECT

```
SELECT  DISTINCT A.Owner, A.Balance
FROM    Account A, Deposit D
WHERE   D.Account = 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	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	11/2/00	10,000.00

```
SELECT      A.Number, A.Owner, D.Date, D.Amount
FROM        Account AS A, Deposit AS D
WHERE       A.Number = D.Account 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	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	11/2/00	10,000.00

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

	Number	Owner	Date	Amount
	102	W. Wei	10/22/00	500.00
	104	M. Jones	10/29/00	1000.00
	105	H. Martin	11/2/00	10000.00

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).

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 may or may not have any rows.



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

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

“A1”

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

“A2”

```


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

```

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!

We say that relational query languages are **closed** when they have this property.

Typical Steps to Define/Use a DB

1. Organize structured data into an **ER Diagram**, Ch. 2
2. Transform an ER diagram into a **Schema of Tables**, Ch. 3
3. Eliminate **anomalies** from tables (**normalization**), Ch. 21
4. Structuring tables **efficiently** (**physical design**), Ch. 22
5. **Managing** those tables using the intergalactic standard language (**SQL**), Ch. 5 Note: the DBMS manages the tables internally using **Relational Algebra**, Ch. 4
6. Protect those tables during **concurrent** use (**ACID** properties), Ch. 16
7. Access those tables through a **web-based interface** (some scripting language)

But first ... we are going to learn how to write SQL queries!