CMSC424: Database Design

Instructor: Amol Deshpande
amol@cs.umd.edu

Databases

- Data Models
  - Conceptual representation of the data
- Data Retrieval
  - How to ask questions of the database
  - How to answer those questions
- Data Storage
  - How/where to store data, how to access it
- Data Integrity
  - Manage crashes, concurrency
  - Manage semantic inconsistencies
Outline

- Overview of modeling
- Entity-relationship Model (E/R model)
- Relational Model
- Converting from E/R to Relational
- Extra slides

Data Modeling

- Goals:
  - Conceptual representation of the data
  - “Reality” meets “bits and bytes”
  - Must make sense, and be usable by other people
- We will study:
  - Entity-relationship Model
  - Relational Model
    - Note the difference !!
  - May study XML-based models or object-oriented models
- Why so many models?
Data Modeling

- Why so many models?

- Tradeoff between:
  - Descriptive capabilities
  - How many concepts it has?
  - What can’t it capture?
  - Is it reasonably easy for humans to use and reason about?
  - Performance
  - Can it be implemented reasonably efficiently?

Motivation

- You’ve just been hired by Bank of America as their DBA for their online banking web site.
- You are asked to create a database that monitors:
  - customers
  - accounts
  - loans
  - branches
  - transactions, ...

- Now what??!!!
  - Start with modeling the information that needs to be stored..
**Database Design Steps**

- **Entity-relationship Model**
  - Typically used for conceptual database design

- **Relational Model**
  - Typically used for logical database design

**Three Levels of Modeling**

- Conceptual DB design
  - Conceptual Data Model
- Logical DB design
  - Logical Data Model
- Physical DB design
  - Physical Data Model

**Outline**

- Overview of modeling
- Entity-relationship Model (E/R model)
- Relational Model
- Converting from E/R to Relational
- Extra slides
Entity-Relationship Model

- Two key concepts
  - **Entities:**
    - An object that *exists* and is *distinguishable* from other objects
    - Examples: Bob Smith, BofA, CMSC424
    - Have *attributes* (people have names and addresses)
    - Form *entity sets* with other entities of the same type that share the same properties
      - Set of all people, set of all classes
    - Entity sets may overlap
      - Customers and Employees

- Relationships:
  - Relate 2 or more entities
    - E.g. Bob Smith *has account at* College Park Branch
  - Form *relationship sets* with other relationships of the same type that share the same properties
    - Customers *have accounts at* Branches
  - Can have attributes:
    - *has account at* may have an attribute *start-date*
  - Can involve more than 2 entities
    - Employee *works at* Branch *at* Job
ER Diagram: Starting Example

- Rectangles: entity sets
- Diamonds: relationship sets
- Ellipses: attributes

Rest of the class

- Details of the ER Model
  - How to represent various types of constraints/semantic information etc.

- Design issues

- A detailed example
Next: Relationship Cardinalities

- We may know:
  - One customer can only open one account
  - OR
  - One customer can open multiple accounts

- Representing this is important

- Why?
  - Better manipulation of data
    - If former, can store the account info in the customer table
  - Can enforce such a constraint
    - Application logic will have to do it; NOT GOOD
  - Remember: If not represented in conceptual model, the domain knowledge may be lost

Mapping Cardinalities

- Express the number of entities to which another entity can be associated via a relationship set
- Most useful in describing binary relationship sets
Mapping Cardinalities

- One-to-One
- One-to-Many
- Many-to-One
- Many-to-Many

Express the number of entities to which another entity can be associated via a relationship set

- Most useful in describing binary relationship sets

- N-ary relationships?
  - More complicated
  - Details in the book
Next: Types of Attributes

- Simple vs Composite
  - Single value per attribute?

- Single-valued vs Multi-valued
  - E.g. Phone numbers are multi-valued

- Derived
  - If date-of-birth is present, age can be derived
  - Can help in avoiding redundancy, enforcing constraints etc…

Types of Attributes

![Entity-Relationship Diagram]

- customer
  - cust-id
  - cust-street
  - cust-city
- account
  - number
  - balance
- has
  - access-date
- cust-name
Types of Attributes

Composite Attribute

Next: Keys

- Key = set of attributes that uniquely identifies an entity or a relationship
**Entity Keys**

Possible Keys:
- \{cust-id\}
- \{cust-name, cust-city, cust-street\}
- \{cust-id, age\}

cust-name ?? Probably not.

Domain knowledge dependent !!

---

**Entity Keys**

- **Superkey**
  - any attribute set that can distinguish entities

- **Candidate key**
  - a minimal superkey
    - Can’t remove any attribute and preserve key-ness
      - \{cust-id, age\} not a candidate key
      - \{cust-name, cust-city, cust-street\} is
        - assuming cust-name is not unique

- **Primary key**
  - Candidate key chosen as *the* key by DBA
  - Underlined in the ER Diagram
Entity Keys

- `cust-id` is a natural primary key
- Typically, SSN forms a good primary key
- Try to use a candidate key that rarely changes
  - e.g. something involving address not a great idea

Relationship Set Keys

- What attributes are needed to represent a relationship completely and uniquely?
  - Union of primary keys of the entities involved, and relationship attributes

  - `{cust-id, access-date, account number}` describes a relationship completely
Is \{\text{cust-id, access-date, account number}\} a candidate key?

- No. Attribute \text{access-date} can be removed from this set without losing key-ness
- In fact, union of primary keys of associated entities is always a superkey

Is \{\text{cust-id, account-number}\} a candidate key?

- Depends
Is \{\text{cust-id, account-number}\} a candidate key?

- Depends

- If one-to-one relationship, either \{\text{cust-id}\} or \{\text{account-number}\} sufficient
  - Since a given customer can only have one account, she can only participate in one relationship
  - Ditto account

- If one-to-many relationship (as shown), \{\text{account-number}\} is a candidate key
  - A given customer can have many accounts, but at most one account holder per account allowed
Relationship Set Keys

- General rule for binary relationships
  - one-to-one: primary key of either entity set
  - one-to-many: primary key of the entity set on the many side
  - many-to-many: union of primary keys of the associate entity sets

- n-ary relationships
  - More complicated rules

***

- What have we been doing

- Why?

- Understanding this is important
  - Rest are details !!
  - That’s what books/manuals are for.
Next: Recursive Relationships

- Sometimes a relationship associates an entity set to itself

Recursive Relationships

Must be declared with roles
Next: Weak Entity Sets

- An entity set without enough attributes to have a primary key
- E.g. Transaction Entity
  - Attributes:
    - transaction-number, transaction-date, transaction-amount, transaction-type
    - transaction-number: may not be unique across accounts

Weak Entity Sets

- A weak entity set must be associated with an identifying or owner entity set
- Account is the owner entity set for Transaction
Weak Entity Sets

Still need to be able to distinguish between different weak entities associated with the same strong entity

Discriminator: A set of attributes that can be used for that
Weak Entity Sets

- Primary key:
  - Primary key of the associated strong entity +
  - discriminator attribute set
  - For Transaction:
    - \(\text{\{account-number, transaction-number\}}\)

More...

- Read Chapter 6 for:
  - Semantic data constraints
  - Specialization/Generalization/Aggregation
  - Generalization: opposite of specialization
  - Lower- and higher-level entities
  - Attribute inheritance
  - Homework 1 !!
Example Design

- We will model a university database
  - Main entities:
    - Professor
    - Projects
    - Departments
    - Graduate students
    - etc...

- Diagram:
  - Professor: SSN, name, area, rank
  - Project: proj-number, sponsor, start, budget
  - Dept: dept-no, name, office, homepage
  - Grad: name, age, degree
Thoughts...

- Nothing about actual data
  - How is it stored?
- No talk about the query languages
  - How do we access the data?
- Semantic vs Syntactic Data Models
  - Remember: E/R Model is used for conceptual modeling
  - Many conceptual models have the same properties
- They are much more about representing the knowledge than about database storage/querying
Thoughts...

- Basic design principles
  - Faithful
    - Must make sense
  - Satisfies the application requirements
  - Models the requisite domain knowledge
    - If not modeled, lost afterwards
  - Avoid redundancy
    - Potential for inconsistencies
  - Go for simplicity
- Typically an iterative process that goes back and forth

Design Issues

- Entity sets vs attributes
  - Depends on the semantics of the application
  - Consider telephone
- Entity sets vs Relationship sets
  - Consider loan
- N-ary vs binary relationships
  - Possible to avoid n-ary relationships, but there are some cases where it is advantageous to use them
- It is not an exact science !!
Recap

- Entity-relationship Model
  - Intuitive diagram-based representation of domain knowledge, data properties etc...
  - Two key concepts:
    - Entities
    - Relationships
  - We also looked at:
    - Relationship cardinalities
    - Keys
    - Weak entity sets
    - ...

- Entity-relationship Model
  - No standardized model (as far as I know)
    - You will see different types of symbols/constructs
  - Powerful constructs
    - Relational model (next) is not as powerful
  - Easy to reason about/understand/construct
  - Not as easy to implement
    - Came after the relational model, so no real implementation was ever done
    - Mainly used in the design phase
Outline

- Overview of modeling
- Entity-relationship Model (E/R model)
- Relational Model
- Converting from E/R to Relational
- Extra slides

Relational Data Model

Introduced by Ted Codd (late 60’s – early 70’s)

- Before = “Network Data Model” (Cobol as DDL, DML)
- Very contentious: Database Wars (Charlie Bachman vs. Ted Codd)

Relational data model contributes:

1. Separation of logical, physical data models (data independence)
2. Declarative query languages
3. Formal semantics
4. Query optimization (key to commercial success)

1st prototypes:

- Ingres → CA
- Postgres → Illustra → Informix → IBM
- System R → Oracle, DB2
Key Abstraction: Relation

<table>
<thead>
<tr>
<th>bname</th>
<th>acct_no</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>A-101</td>
<td>500</td>
</tr>
<tr>
<td>Brighton</td>
<td>A-201</td>
<td>900</td>
</tr>
<tr>
<td>Brighton</td>
<td>A-217</td>
<td>500</td>
</tr>
</tbody>
</table>

Terms:

- Tables (aka: Relations)

Why called Relations?

Mathematical relations

Given sets: \( R = \{1, 2, 3\}, \ S = \{3, 4\} \)

- \( R \times S = \{ (1, 3), (1, 4), (2, 3), (2, 4), (3, 3), (3, 4) \} \)
- A relation on \( R, S \) is any subset (\( \subseteq \)) of \( R \times S \)
  - (e.g. \( \{ (1, 4), (3, 4) \} \))

Database relations

Given attribute domains

- \( \text{Branches} = \{ \text{Downtown, Brighton, …} \} \)
- \( \text{Accounts} = \{ \text{A-101, A-201, A-217, …} \} \)
- \( \text{Balances} = R \)

\( \text{Account} \subseteq \text{Branches} \times \text{Accounts} \times \text{Balances} \)

\( \{ (\text{Downtown, A-101, 500}), (\text{Brighton, A-201, 900}), (\text{Brighton, A-217, 500}) \} \)
**Relations**

Account =

<table>
<thead>
<tr>
<th>bname</th>
<th>acct_no</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>A-101</td>
<td>500</td>
</tr>
<tr>
<td>Brighton</td>
<td>A-201</td>
<td>900</td>
</tr>
<tr>
<td>Brighton</td>
<td>A-217</td>
<td>500</td>
</tr>
</tbody>
</table>

Considered equivalent to…

\{ (Downtown, A-101, 500),
  (Brighton, A-201, 900),
  (Brighton, A-217, 500) \}

Relational database semantics defined in terms of mathematical relations

**Terms:**

- Tables (aka: Relations)
- Rows (aka: tuples)
- Columns (aka: attributes)
- Schema (e.g.: Acct_Schema = (bname, acct_no, balance))
Definitions

Relation Schema (or Schema)
A list of attributes and their domains
We will require the domains to be atomic

Programming language equivalent: A variable (e.g. x)

Relation Instance
A particular instantiation of a relation with actual values
Will change with time

Programming language equivalent: Value of a variable

<table>
<thead>
<tr>
<th>Name</th>
<th>Stock No</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>A-101</td>
<td>500</td>
</tr>
<tr>
<td>Brighton</td>
<td>A-201</td>
<td>900</td>
</tr>
<tr>
<td>Brighton</td>
<td>A-217</td>
<td>500</td>
</tr>
</tbody>
</table>

So...

- That’s the basic relational model
- That’s it?
  - What about semantic information?
    - Relationships between entities?
  - What about the constraints?
  - How do we represent one-to-one vs many-to-one relationships?
- Most are embedded in the schema
  - Schema may not allow many-to-many relationships
  - Some are explicitly specified using *integrity constraints*
Keys and Relations

- Recall:
  - Keys: Sets of attributes that allow us to identify entities
  - Very loosely speaking, tuples === entities

- Just as in E/R Model:
  - Superkeys, candidate keys, and primary keys

Keys

- Superkey
  - set of attributes of table for which every row has distinct set of values

- Candidate key
  - Minimal such set of attributes

- Primary key
  - DB Chosen Candidate key
  - Plays a very important role
    - E.g. relations typically sorted by this
Keys

- Also act as integrity constraints
  - i.e., guard against illegal/invalid instance of given schema

  e.g., Branch = (bname, bcity, assets)

<table>
<thead>
<tr>
<th>bname</th>
<th>bcity</th>
<th>assets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brighton</td>
<td>Brooklyn</td>
<td>5M</td>
</tr>
<tr>
<td>Brighton</td>
<td>Boston</td>
<td>3M</td>
</tr>
</tbody>
</table>

Invalid

In fact, keys are one of the primary ways to enforce constraints/structure

Consider a one-to-many relationship e.g.
  - Between customers and accounts
  - The relational model will be:
    - Customers(custid, custname, ...)
    - Accounts(accountid, custid, balance, ...)
  - Allows for multiple accounts per customer, but not multiple customers per account
    - Not possible to store such information

In other words, constraints will lead to less representation power
  - Contrast with:
    - Customers(custid, custname, ...)
    - Accounts(accountid, balance, ...)
    - CustomerHasAccounts(custid, accountid)
More on Keys

- Determining Primary Keys
  - If relation schema derived from E-R diagrams, we can determine the primary keys using the original entity and relationship sets
  - Otherwise, same way we do it for E-R diagrams
    - Find candidate keys (minimal sets of attributes that can uniquely identify a tuple)
    - Designate one of them to be primary key

- Foreign Keys
  - If a relation schema includes the primary key of another relation schema, that attribute is called the foreign key.
Recap: Relational Data Model

Key Abstraction: Relation

Mathematical relations

Given sets:  \( R = \{1, 2, 3\} \),  \( S = \{3, 4\} \)
- \( R \times S = \{(1, 3), (1, 4), (2, 3), (2, 4), (3, 3), (3, 4)\} \)
- A relation on \( R, S \) is any subset (\( \subseteq \)) of \( R \times S \) (e.g: \( \{(1, 4), (3, 4)\}\))

Database relations

Given attribute domains

<table>
<thead>
<tr>
<th>Branches</th>
<th>Accounts</th>
<th>Balances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>A-101, A-201, A-217, ...</td>
<td>R</td>
</tr>
<tr>
<td>Brighton</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Account \( \subseteq \) Branches \( \times \) Accounts \( \times \) Balances

<table>
<thead>
<tr>
<th>bname</th>
<th>acct_no</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>A-101</td>
<td>500</td>
</tr>
<tr>
<td>Brighton</td>
<td>A-201</td>
<td>900</td>
</tr>
<tr>
<td>Brighton</td>
<td>A-217</td>
<td>500</td>
</tr>
</tbody>
</table>

Recap: Terms and Definitions

1. **Tables** = Relations
2. **Columns** = Attributes
3. **Rows** = Tuples
4. **Relation Schema (or Schema)**
   1. A list of attributes and their domains
   2. We will require the domains to be atomic
   3. E.g. account(assets, account-number, branch-name, balance)
5. **Relation Instance**
   1. A particular instantiation of a relation with actual values
   2. Will change with time
Recap: Thoughts

- Very few constructs
  - Just a “relation”, and “keys”
  - Normalization
    - We will discuss this in more detail later

- Most constraints are embedded in the schema itself
  - Particularly relationship-cardinality constraints

- Not as easy to reason about
  - In an E/R diagram, changing a relationship from one-to-one to many-to-many just requires changing the arrow types
  - More complicated to do with relations
    - Need to construct a new relation, add foreign keys etc...

Outline

- Overview of modeling
- Entity-relationship Model (E/R model)
- Relational Model
- Converting from E/R to Relational
- Extra slides
E/R Diagrams → Relations

› Convert entity sets into a relational schema with the same set of attributes

\[
\text{Customer} \\
\text{cname} \quad \text{ccity} \quad \text{cstreet} \rightarrow \text{Customer\_Schema}(\text{cname}, \text{ccity}, \text{cstreet})
\]

\[
\text{Branch} \\
\text{bname} \quad \text{bcity} \quad \text{assets} \rightarrow \text{Branch\_Schema}(\text{bname}, \text{bcity}, \text{assets})
\]

E/R Diagrams → Relations

› Convert relationship sets also into a relational schema
› Remember: A relationship is completely described by primary keys of associate entities and its own attributes

\[
\text{Account\_Schema}(\text{acct-no}, \text{balance})
\]

\[
\text{Depositor\_Schema}(\text{cname}, \text{acct-no}, \text{access-date})
\]

\[
\text{Customer\_Schema}(\text{cname}, \text{ccity}, \text{cstreet})
\]

Well… Not quite. We can do better.
It depends on the relationship cardinality
E/R Diagrams $\Rightarrow$ Relations

- Say One-to-Many Relationship from Customer to Account
  $\Rightarrow$ Many accounts per customer

Account (acct-no, balance, access-date)

Customer (cname, ccity, cstreet)

E/R Diagrams $\Rightarrow$ Relations

<table>
<thead>
<tr>
<th>E/R</th>
<th>Relational Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Sets $E_i$</td>
<td>$E = {a_1, \ldots, a_n}$</td>
</tr>
</tbody>
</table>
E/R Diagrams \(\rightarrow\) Relations

<table>
<thead>
<tr>
<th>E/R</th>
<th>Relational Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Sets</td>
<td>(E = (a_1, \ldots, a_n))</td>
</tr>
<tr>
<td>Relationship Sets</td>
<td>(R = (a_{12}, b_{12}, c_1, \ldots, c_n))</td>
</tr>
<tr>
<td>(a_i): (E_i)'s key</td>
<td>(b_i): (E_2)'s key</td>
</tr>
<tr>
<td>(c_1, \ldots, c_k): attributes of (R)</td>
<td></td>
</tr>
</tbody>
</table>

*Not the whole story for Relationship Sets ...*

---

E/R Diagrams \(\rightarrow\) Relations

<table>
<thead>
<tr>
<th>Relationship Cardinality</th>
<th>Relational Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>n:m</td>
<td>(E_1 = (a_1, \ldots, a_n))</td>
</tr>
<tr>
<td>(E_2 = (b_1, \ldots, b_m))</td>
<td></td>
</tr>
<tr>
<td>(R = (a_{12}, b_{12}, c_1, \ldots, c_n))</td>
<td></td>
</tr>
</tbody>
</table>
E/R Diagrams → Relations

<table>
<thead>
<tr>
<th>Relationship Cardinality</th>
<th>Relational Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>E₁ = (a₁, ..., aₙ)</td>
</tr>
<tr>
<td></td>
<td>E₂ = (b₁, ..., bₘ)</td>
</tr>
<tr>
<td></td>
<td>R = (a₁, b₁, c₁, ..., cₙ)</td>
</tr>
<tr>
<td>n:1</td>
<td>E₁ = (a₁, ..., aₙ, b₁, c₁, ..., cₙ)</td>
</tr>
<tr>
<td></td>
<td>E₂ = (b₁, ..., bₘ)</td>
</tr>
<tr>
<td>1:n</td>
<td>E₁ = (a₁, ..., aₙ)</td>
</tr>
<tr>
<td></td>
<td>E₂ = (b₁, ..., bₘ, a₁, c₁, ..., cₙ)</td>
</tr>
</tbody>
</table>
**E/R Diagrams ➔ Relations**

<table>
<thead>
<tr>
<th>Relationship Cardinality</th>
<th>Relational Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>n:m</td>
<td>E₁ = (a₁, …, aₙ)</td>
</tr>
<tr>
<td></td>
<td>E₂ = (b₁, …, bₚ)</td>
</tr>
<tr>
<td></td>
<td>R = (a₁, b₁, c₁, …, cₜ)</td>
</tr>
<tr>
<td>n:1</td>
<td>E₁ = (a₁, …, aₙ)</td>
</tr>
<tr>
<td></td>
<td>E₂ = (b₁, …, bₚ)</td>
</tr>
<tr>
<td>1:n</td>
<td>E₁ = (a₁, …, aₙ)</td>
</tr>
<tr>
<td></td>
<td>E₂ = (b₁, …, bₚ, a₁, c₁, …, cₜ)</td>
</tr>
<tr>
<td>1:1</td>
<td>Treat as n:1 or 1:n</td>
</tr>
</tbody>
</table>

Translating E/R Diagrams to Relations

Q. How many tables does this get translated into?

* A. 6 (account, branch, customer, loan, depositor, borrower)
## Bank Database

### Account
<table>
<thead>
<tr>
<th>bname</th>
<th>acct_no</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>A-101</td>
<td>500</td>
</tr>
<tr>
<td>Mianus</td>
<td>A-115</td>
<td>700</td>
</tr>
<tr>
<td>Perry</td>
<td>A-102</td>
<td>400</td>
</tr>
<tr>
<td>R.H.</td>
<td>A-305</td>
<td>350</td>
</tr>
<tr>
<td>Brighton</td>
<td>A-201</td>
<td>800</td>
</tr>
<tr>
<td>Redwood</td>
<td>A-223</td>
<td>700</td>
</tr>
<tr>
<td>Dayton</td>
<td>A-217</td>
<td>750</td>
</tr>
</tbody>
</table>

### Depositor
<table>
<thead>
<tr>
<th>cname</th>
<th>acct_no</th>
</tr>
</thead>
<tbody>
<tr>
<td>Johnson</td>
<td>A-101</td>
</tr>
<tr>
<td>Smith</td>
<td>A-215</td>
</tr>
<tr>
<td>Hayes</td>
<td>A-102</td>
</tr>
<tr>
<td>Turner</td>
<td>A-300</td>
</tr>
<tr>
<td>Johnson</td>
<td>A-201</td>
</tr>
<tr>
<td>Jones</td>
<td>A-117</td>
</tr>
<tr>
<td>Lindsey</td>
<td>A-222</td>
</tr>
</tbody>
</table>

### Customer
<table>
<thead>
<tr>
<th>cname</th>
<th>cstreet</th>
<th>ccity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jones</td>
<td>Main</td>
<td>Harrison</td>
</tr>
<tr>
<td>Smith</td>
<td>North</td>
<td>Rye</td>
</tr>
<tr>
<td>Hayes</td>
<td>Park</td>
<td>Pittsfield</td>
</tr>
<tr>
<td>Turner</td>
<td>Pulsom</td>
<td>Stanford</td>
</tr>
<tr>
<td>Williams</td>
<td>Adams</td>
<td>Spring</td>
</tr>
<tr>
<td>Adams</td>
<td>Adams</td>
<td>Spring</td>
</tr>
<tr>
<td>Turner</td>
<td>Turner</td>
<td>Pittsfield</td>
</tr>
<tr>
<td>Smith</td>
<td>Smith</td>
<td>Pitfield</td>
</tr>
<tr>
<td>Turner</td>
<td>Turner</td>
<td>Senator</td>
</tr>
<tr>
<td>Stones</td>
<td>Stone</td>
<td>Woodside</td>
</tr>
<tr>
<td>Becker</td>
<td>Becker</td>
<td>Brooklyn</td>
</tr>
<tr>
<td>Becker</td>
<td>Becker</td>
<td>Brooklyn</td>
</tr>
<tr>
<td>Green</td>
<td>Green</td>
<td>Walnut</td>
</tr>
</tbody>
</table>

### Borrower
<table>
<thead>
<tr>
<th>cname</th>
<th>lno</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jones</td>
<td>L-17</td>
</tr>
<tr>
<td>Smith</td>
<td>L-13</td>
</tr>
<tr>
<td>Hayes</td>
<td>L-15</td>
</tr>
<tr>
<td>Turner</td>
<td>L-14</td>
</tr>
<tr>
<td>Smith</td>
<td>L-11</td>
</tr>
<tr>
<td>Adams</td>
<td>L-15</td>
</tr>
<tr>
<td>Williams</td>
<td>L-17</td>
</tr>
</tbody>
</table>

### Loan
<table>
<thead>
<tr>
<th>bname</th>
<th>lno</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>L-17</td>
<td>1000</td>
</tr>
<tr>
<td>Redwood</td>
<td>L-23</td>
<td>2000</td>
</tr>
<tr>
<td>Perry</td>
<td>L-18</td>
<td>1500</td>
</tr>
<tr>
<td>Downtown</td>
<td>L-14</td>
<td>1600</td>
</tr>
<tr>
<td>Marcus</td>
<td>L-44</td>
<td>500</td>
</tr>
<tr>
<td>Perry</td>
<td>L-16</td>
<td>1200</td>
</tr>
</tbody>
</table>

### Branch
<table>
<thead>
<tr>
<th>bname</th>
<th>bcity</th>
<th>assets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>Brooklyn</td>
<td>3.1M</td>
</tr>
<tr>
<td>Redwood</td>
<td>Pittsfield</td>
<td>1.7M</td>
</tr>
<tr>
<td>Perry</td>
<td>Homestead</td>
<td>0.4M</td>
</tr>
<tr>
<td>Mianus</td>
<td>Homestead</td>
<td>1.9M</td>
</tr>
<tr>
<td>R.H.</td>
<td>Homestead</td>
<td>0.8M</td>
</tr>
<tr>
<td>Palme</td>
<td>Bennington</td>
<td>0.3M</td>
</tr>
<tr>
<td>N. Town</td>
<td>Ryd</td>
<td>2.1M</td>
</tr>
<tr>
<td>Brighton</td>
<td>Brooklyn</td>
<td>7.1M</td>
</tr>
</tbody>
</table>

### E/R Diagrams & Relations

**Weak Entity Sets**

$E_1 = (a_{1}, \ldots, a_{n})$

$E_2 = (a_{1}, b_{1}, \ldots, b_{m})$
### E/R Diagrams & Relations

<table>
<thead>
<tr>
<th>E/R</th>
<th>Relational Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Multivalued Attributes</strong></td>
<td></td>
</tr>
<tr>
<td>Employee</td>
<td></td>
</tr>
<tr>
<td>Emp = (ssn, name)</td>
<td></td>
</tr>
<tr>
<td>Emp-Phones = (ssn, phone)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ssn name ssn phone</td>
</tr>
<tr>
<td>001 Smith 001 4-1234</td>
<td></td>
</tr>
<tr>
<td>... ... 001 4-5678</td>
<td></td>
</tr>
<tr>
<td>Emp Emp-Phones</td>
<td></td>
</tr>
</tbody>
</table>

---

### E/R Diagrams & Relations

<table>
<thead>
<tr>
<th>E/R</th>
<th>Relational Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Subclasses</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Method 1:</td>
</tr>
<tr>
<td>E = (a₁, ..., aₙ)</td>
<td></td>
</tr>
<tr>
<td>E₁ = (a₁, b₁, ..., bₘ)</td>
<td></td>
</tr>
<tr>
<td>E₂ = (a₁, c₁, ..., cₖ)</td>
<td></td>
</tr>
</tbody>
</table>
Subclasses example:

- **Method 1:**
  - **Account** = (acct_no, balance)
  - **SAccount** = (acct_no, interest)
  - **CAccount** = (acct_no, overdraft)

- **Method 2:**
  - **SAccount** = (acct_no, balance, interest)
  - **CAccount** = (acct_no, balance, overdraft)
Outline

- Overview of modeling
- Entity-relationship Model (E/R model)
- Relational Model
- Converting from E/R to Relational
- Extra slides

Extra slides...

- E/R modeling stuff not covered in class follows...
Next: Data Constraints

- Representing semantic data constraints
  - We already saw constraints on relationship cardinalities

Participation Constraint

- Given an entity set E, and a relationship R it participates in:
  - If every entity in E participates in at least one relationship in R, it is total participation
  - partial otherwise
Participation Constraint

How many relationships can an entity participate in?

Cardinality Constraints

Minimum - 0
Maximum - no limit
Minimum - 1
Maximum - 1
Next: Specialization

- Consider entity person:
  - Attributes: name, street, city
- Further classification:
  - customer
    - Additional attributes: customer-id, credit-rating
  - employee
    - Additional attributes: employee-id, salary

- Note similarities to object-oriented programming

Finally: Aggregation

- No relationships between relationships
  - E.g.: Associate account officers with has account relationship set
Finally: Aggregation

- Associate an account officer with each account?
  - What if different customers for the same account can have different account officers?

```
customer has account

? account officer

employee
```

Finally: Aggregation

- Solution: Aggregation

```
customer has account

account officer

employee
```