Views

- Limited access to DB.
- Tailored schema

Consider a person who needs to know a customer’s loan number but has no need to see the loan amount. This person should see a relation described as:

$$\Pi_{\text{customer-name, loan-number}} (\text{borrower loan})$$

A relation that is made visible to a user as a “virtual relation” is called a view.
A view is defined using the `create view` statement which has the form:

```sql
create view v as <query expression>
```

where `<query expression>` is any legal relational algebra query expression. The view name given as `v`.

Once a view is defined, the view name can be used to refer to the virtual relation that the view generates.

View definition is not the same as creating a new relation by evaluating the query expression. Rather, a view definition causes the saving of an expression to be substituted into queries using the view.
View Examples

- Consider the view (named *all-customer*) consisting of branches and their customers.

  ```
  create view all-customer as
  \( \Pi_{\text{branch-name}, \text{customer-name}} (\text{depositor account}) \cup \Pi_{\text{branch-name}, \text{customer-name}} (\text{borrower loan}) \)
  ```

- We can find all customers of the Perryridge branch by writing:

  ```
  \( \Pi_{\text{customer-name}} (\sigma_{\text{branch-name} = \text{"Perryridge"}} (\text{all-customer})) \)
  ```
Updates Through View

- Database modifications expressed as views must be translated to modifications of the actual relations in the database.
- Consider the person who needs to see all loan data in the loan relation except amount. The view given to the person, branch-loan, is defined as:
  
  \[
  \text{create view} \quad \text{branch-loan as} \\
  \Pi_{\text{branch-name, loan-number}} (\text{loan})
  \]

- Since we allow a view name to appear wherever a relation name is allowed, the person may write:
  
  \[
  \text{branch-loan } \leftarrow \text{branch-loan } \cup \{(\text{“Perryridge”, L-37})\}
  \]
The previous insertion must be represented by an insertion into the actual relation `loan` from which the view `branch-loan` is constructed.

An insertion into `loan` requires a value for `amount`. The insertion can be dealt with by either:

- rejecting the insertion and returning an error message to the user.
- inserting a tuple ("L-37", "Perryridge", `null`) into the `loan` relation
Updates Through Views (Cont.)

- Some updates through views are impossible to translate.
  
  create view v as $\sigma_{\text{branch-name} = \text{“Perryridge”}} (\text{account})$
  
  $v \leftarrow v \cup (L-99, \text{Downtown}, 23)$

- Others cannot be translated uniquely
  
  $\text{all-customer} \leftarrow \text{all-customer} \cup (\text{Perryridge}, \text{John})$
  
  - Have to choose loan or account, and create a new loan/account number!
Views Defined Using Other Views

- One view may be used in the expression defining another view.
- A view relation $v_1$ is said to depend directly on a view relation $v_2$ if $v_2$ is used in the expression defining $v_1$.
- A view relation $v_1$ is said to depend on view relation $v_2$ if either $v_1$ depends directly on $v_2$ or there is a path of dependencies from $v_1$ to $v_2$. 
View Expansion

- Let view \( v_1 \) be defined by an expression \( e_1 \) that may itself contain uses of view relations.
- View expansion of an expression repeats the following replacement step:
  ```
  repeat
  Find any view relation \( v_i \) in \( e_1 \)
  Replace the view relation \( v_i \) by the expression defining \( v_i \)
  until no more view relations are present in \( e_1 \)
  ```
- As long as the view definitions are not recursive, this loop will terminate
Database Design and the Entity/Relationship Model
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??!!!
1. Requirements Specification
   ◦ Determine the requirements of clients

2. Conceptual Design
   ◦ Express client requirements in terms of some high-level model (e.g., E/R model).
   ◦ Confirm with clients that requirements are correct.

3. Functional Requirements
   ◦ Specify required data operations
   ◦ priorities, response times

4. Logical Design
   ◦ Convert E/R model to relational, object-based, XML-based,…

5. Physical Design
   ◦ Specify file organizations, build indexes
What is a Data Model?

- Framework for organizing and interpreting data

Example: E/R Data Model
E/R Data Model

Basics

- **Entities**
  - noun phrases (e.g., Bob Smith, Thayer St. Branch)
  - contained in **entity sets** (e.g. Employee, Branch)
  - have **attributes** (e.g., Employee = (essn, ename, ...))

- **Relationships**
  - verb phrases (e.g., works_at, works_for)
  - relate 2 (binary) or more (n-ary) entities
  - **relationship sets** characterize relationships amongst entity sets
    - e.g., (Bob Smith, Thayer St Branch) ∈ Works_At
E/R Data Model

An Example

Lots of notation to come. Color is irrelevant
E/R Data Model

Types of Attributes

Employee
- essn
- phone
- ename
- children

Works_At
- since
- seniority

Branch
- bname
- bcity

Works_For
- manager
- worker

Attributes:
- Default: ename
- Multivalued: children
- Derived: seniority
E/R Data Model

Types of relationships

Employee
- essn
- phone
- ename
- children

Works_At
- since
- seniority

Branch
- bname
- bcity

Works_For
- manager
- worker

Many-to-One (n:1)
- Works_For

Many-to-Many (n:m)
- Works_At
E/R Data Model

Recursive relationships

Employee
- essn
- phone
- ename
- children

Works_For
- manager
- worker

Works_At
- since
- seniority

Branch
- bname
- bcity

Recursive relationships: Must be declared with roles
**E/R Data Model**

*Design Issue #1: Entity Sets vs. Attributes*

- **To resolve, determine how phones are used**
  - 1. Can many employees share a phone?  
    (If yes, then (b))
  - 2. Can employees have multiple phones?  
    (if yes, then (b), or (a) with multivalued attributes)
  - 3. Else (a)
An Example: How to model bank loans

To resolve, determine how loans are issued

1. Can there be more than one customer per loan?
   - If yes, then (a). Otherwise, loan info must be replicated for each customer (wasteful, potential update anomalies)

2. Is loan a noun or a verb?
   - Both, but more of a noun to a bank. (hence (a) probably more appropriate)
E/R Data Model

Design Issue #3: Relationship Cardinalities

- An Example:
  - Diagram: Customer <--- Borrows ----> Loan

- Variations on Borrows:
  - 1. Can a customer hold multiple loans?
  - 2. Can a loan be jointly held by more than 1 customer?
**E/R Data Model**

**Design Issue #3: Relationship Cardinalities**

- **Customer** \( \rightarrow \) **Borrows** \( \rightarrow \) **Loan**

### Cardinalities of Borrows:

<table>
<thead>
<tr>
<th>Type</th>
<th>Illustrated</th>
<th>Multiple Loans?</th>
<th>Joint Loans?</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-to-One (1:1)</td>
<td>Borrows</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Many-to-one (n:1)</td>
<td>Borrows</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>One-to-many (1:n)</td>
<td>Borrows</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Many-to-many (n:m)</td>
<td>Borrows</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
In general...

- **1:1**

- **n:1**

- **1:n**

- **n:m**
An Example: **Works_At**

**Ternary:**

An example of ternary relationship is shown in the diagram.

**Binary:**

In contrast, a binary relationship is depicted in the lower part of the diagram.

Choose n-ary when possible!
Key = set of attributes identifying individual entities or relationships

A. Superkey:
- any attribute set that distinguishes identities
- e.g., \{essn\}, \{essn, ename, eaddress\}

B. Candidate Key:
- “minimal superkey” (can’t remove attributes and preserve “keyness”)
- e.g., \{essn\}, \{ename, eaddress\}

C. Primary Key:
- candidate key chosen as the key by a DBA
- e.g., \{essn\} (denoted by underline)
Q: What attributes are needed to represent relationships in \textit{Works\_At}?

A: \{essn, bname, since\}
Q: What are the candidate keys of Works_At?

A: \{essn\}
**Q: What are the candidate keys if Works_At is...?**

- **a. 1:n**
  
  A: \{bname\}

- **b. n:m**
  
  A: \{essn, bname\}

- **c. 1:1**
  
  A: \{essn\} or \{bname\}
General Rules for Relationship Set Keys

- If R is:

<table>
<thead>
<tr>
<th>R</th>
<th>Candidate Keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:1</td>
<td>P (E₁) or P (E₂)</td>
</tr>
<tr>
<td>1:n</td>
<td>P (E₂)</td>
</tr>
<tr>
<td>n:1</td>
<td>P (E₁)</td>
</tr>
<tr>
<td>n:m</td>
<td>P (E₁) \cup P (E₂)</td>
</tr>
</tbody>
</table>
E/R Data Model

Existence Dependencies and Weak Entity Sets

- Idea:
  - Existence of one entity depends on another

- Example: Loans and Loan Payments
Existence Dependencies and Weak Entity Sets

- **Weak Entity Sets**
  - existence of payments depends upon loans
  - have no superkeys: different payment records (for different loans) can be identical
  - instead of keys, **discriminators**: discriminate between payments for given loan (e.g., pno)
Identifying Relationships

We say:

- Loan is **dominant** in Loan_Pmt
- Payment is **subordinate** in Loan_Pmt
- Payment is **existence dependent** on Loan
All elements of Payment appear in Loan_Pmt
Q. Is \{att_{b1}, \ldots, att_{bn}\} a superkey of E_2?
A: No

Q. Name a candidate key of E_2
A: \{att_{a1}, att_{b1}\}

Q. Does total participation of E_2 in R \Rightarrow E_2 is existence-dep?
A: No
E/R Data Model

Extensions to the Model: Specialization and Generalization

- **An Example:**
  - Customers can have checking and savings accts
  - Checking ~ Savings (many of the same attributes)

- **Old Way:**

```
Customer
  Has1
    Savings Acct
      acct_no balance interest
  Has2
    Checking Acct
      acct_no balance overdraft
```
E/R Data Model

Extensions to the Model: Specialization and Generalization

- **An Example:**
  - Customers can have checking and savings accts
  - Checking ~ Savings (many of the same attributes)

- **Alternative Way:**

![Diagram showing specialization and generalization with relationships between Customer, Has, Account, Isa, Savings Acct, and Checking Acct.](image-url)
Subclass Distinctions:

1. User-Defined vs. Condition-Defined

User: Membership in subclasses explicitly determined (e.g., Employee, Manager < Person)

Condition: Membership predicate associated with subclasses - e.g:

- Child: age < 18
- Adult: 18 ≤ age
- Senior: age > 65
Subclass Distinctions:

2. Overlapping vs. Disjoint

- Overlapping: Entities can belong to >1 entity set (e.g., Adult, Senior)
- Disjoint: Entities belong to exactly 1 entity set (e.g., Child)
E/R Data Model

Summary

- **Entities, Relationships (sets)**
- **Both can have attributes** (simple, multivalued, derived, composite)
- **Cardinality of relationship sets** (1:1, n:1, n:m)
- **Keys**: superkeys, candidate keys, primary key
  - **DBA chooses primary key for entity sets**
  - **Automatically determined for relationship sets**
- **Weak Entity Sets, Existence Dependence, Total/Partial Participation**
- **Specialization** (E/R + inheritance)
E/R Diagrams and Relations

<table>
<thead>
<tr>
<th>E/R</th>
<th>Relational Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Entity Sets</strong></td>
<td>E = (a₁, ..., aₙ)</td>
</tr>
<tr>
<td>E</td>
<td></td>
</tr>
<tr>
<td>a₁</td>
<td>...</td>
</tr>
<tr>
<td>aₙ</td>
<td></td>
</tr>
</tbody>
</table>
E/R Diagrams and Relations

<table>
<thead>
<tr>
<th>Entity Sets</th>
<th>Relational Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>E = (a₁, ..., aₙ)</td>
</tr>
<tr>
<td>a₁</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>aₙ</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relationship Sets</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>E₁</td>
<td>R</td>
</tr>
<tr>
<td>a₁</td>
<td>b₁</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>aₙ</td>
<td>bₘ</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>E₂</td>
<td></td>
</tr>
<tr>
<td>c₁</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>cₖ</td>
<td></td>
</tr>
</tbody>
</table>

R = (a₁, b₁, c₁, ..., cₖ)

- a₁: E₁’s key
- b₁: E₂’s key
- c₁, ..., cₖ: attributes of R

Not the whole story for Relationship Sets ...
What about…

Could have: \( R = (a_1, b_1, c_1, \ldots, c_k) \) but…

- \( a_1 \) is a key for \( R \)
- \( a_1 \) also a key for \( E_1 = (a_1, \ldots, a_n) \)

Instead:

- Ignore \( R \) as a separate relation
- Add \( b_1, c_1, \ldots, c_k \) to \( E_1 \) instead (i.e.: \( E_1 = (a_1, \ldots, a_n, b_1, c_1, \ldots, c_k) \))

Rule of Thumb
Fewer tables
good, as long as
no redundancy
### E/R Diagrams and Relations

<table>
<thead>
<tr>
<th>Relationship Cardinality</th>
<th>Relational Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>

**Relationship Cardinality**

- \( E_1 \) \( n:m \) \( E_2 \)

**Relational Schema**

- \( E_1 = (a_1, \ldots, a_n) \)
- \( E_2 = (b_1, \ldots, b_m) \)
- \( R = (a_1, b_1, c_1, \ldots, c_n) \)
# E/R Diagrams and Relations

<table>
<thead>
<tr>
<th>Relationship Cardinality</th>
<th>Relational Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>n:m</strong></td>
<td>[ R = (a_1, \ldots, a_n, b_1, \ldots, b_m, c_1, \ldots, c_k) ]</td>
</tr>
<tr>
<td></td>
<td>[ E_1 = (a_1, \ldots, a_n) ]</td>
</tr>
<tr>
<td></td>
<td>[ E_2 = (b_1, \ldots, b_m) ]</td>
</tr>
<tr>
<td><strong>n:1</strong></td>
<td>[ E_1 = (a_1, \ldots, a_n, b_1, c_1, \ldots, c_n) ]</td>
</tr>
<tr>
<td></td>
<td>[ E_2 = (b_1, \ldots, b_m) ]</td>
</tr>
</tbody>
</table>
### E/R Diagrams and Relations

#### Relationship Cardinality

<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) ) &lt;br&gt; ( E_2 = (b_1, \ldots, b_m) ) &lt;br&gt; ( R = (a_1, b_1, c_1, \ldots, c_n) )</td>
</tr>
<tr>
<td>n:1</td>
<td>( E_1 = (a_1, \ldots, a_n, b_1, c_1, \ldots, c_n) ) &lt;br&gt; ( E_2 = (b_1, \ldots, b_m) )</td>
</tr>
<tr>
<td>1:n</td>
<td>( E_1 = (a_1, \ldots, a_n) ) &lt;br&gt; ( E_2 = (b_1, \ldots, b_m, a_1, c_1, \ldots, c_n) )</td>
</tr>
</tbody>
</table>
# E/R Diagrams and Relations

<table>
<thead>
<tr>
<th>Relationship Cardinality</th>
<th>Relational Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="Diagram.png" alt="Diagram" /></td>
<td><img src="Diagram.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relationship</th>
<th>Relational Schema</th>
</tr>
</thead>
</table>
| n:m          | $E_1 = (a_1, \ldots, a_n)$  
$E_2 = (b_1, \ldots, b_m)$  
$R = (a_1, b_1, c_1, \ldots, c_n)$ |
| n:1          | $E_1 = (a_1, \ldots, a_n, b_1, c_1, \ldots, c_n)$  
$E_2 = (b_1, \ldots, b_m)$ |
| 1:n          | $E_1 = (a_1, \ldots, a_n)$  
$E_2 = (b_1, \ldots, b_m, a_1, c_1, \ldots, c_n)$ |
| 1:1          | Treat as n:1 or 1:n |
Q. How many tables does this get translated into?

A. 6 (account, branch, customer, loan, depositor, borrower)
Q. What are the schemas?
# Bank Database

## Account

<table>
<thead>
<tr>
<th>bname</th>
<th>acct_no</th>
<th>balance</th>
</tr>
</thead>
</table>

## Branch

<table>
<thead>
<tr>
<th>bname</th>
<th>bcity</th>
<th>assets</th>
</tr>
</thead>
</table>

## Depositor

<table>
<thead>
<tr>
<th>cname</th>
<th>acct_no</th>
</tr>
</thead>
</table>

## Customer

<table>
<thead>
<tr>
<th>cname</th>
<th>cstreet</th>
<th>ccity</th>
</tr>
</thead>
</table>

## Borrower

<table>
<thead>
<tr>
<th>cname</th>
<th>lno</th>
</tr>
</thead>
</table>

## Loan

<table>
<thead>
<tr>
<th>bname</th>
<th>lno</th>
<th>amt</th>
</tr>
</thead>
</table>
## 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-215</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>900</td>
</tr>
<tr>
<td>Redwood</td>
<td>A-222</td>
<td>700</td>
</tr>
<tr>
<td>Brighton</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-305</td>
</tr>
<tr>
<td>Johnson</td>
<td>A-201</td>
</tr>
<tr>
<td>Jones</td>
<td>A-217</td>
</tr>
<tr>
<td>Lindsay</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>Main</td>
<td>Harrison</td>
</tr>
<tr>
<td>Curry</td>
<td>North</td>
<td>Rye</td>
</tr>
<tr>
<td>Lindsay</td>
<td>Park</td>
<td>Pittsfield</td>
</tr>
<tr>
<td>Turner</td>
<td>Putnam</td>
<td>Stanford</td>
</tr>
<tr>
<td>Williams</td>
<td>Nassau</td>
<td>Princeton</td>
</tr>
<tr>
<td>Adams</td>
<td>Spring</td>
<td>Pittsfield</td>
</tr>
<tr>
<td>Johnson</td>
<td>Alma</td>
<td>Palo Alto</td>
</tr>
<tr>
<td>Glenn</td>
<td>Sand Hill</td>
<td>Woodside</td>
</tr>
<tr>
<td>Brooks</td>
<td>Senator</td>
<td>Brooklyn</td>
</tr>
<tr>
<td>Green</td>
<td>Walnut</td>
<td>Stanford</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>9M</td>
</tr>
<tr>
<td>Redwood</td>
<td>Palo Alto</td>
<td>2.1M</td>
</tr>
<tr>
<td>Perry</td>
<td>Horseneck</td>
<td>1.7M</td>
</tr>
<tr>
<td>Mianus</td>
<td>Horseneck</td>
<td>0.4M</td>
</tr>
<tr>
<td>R.H.</td>
<td>Horseneck</td>
<td>8M</td>
</tr>
<tr>
<td>Pownel</td>
<td>Bennington</td>
<td>0.3M</td>
</tr>
<tr>
<td>N. Town</td>
<td>Rye</td>
<td>3.7M</td>
</tr>
<tr>
<td>Brighton</td>
<td>Brooklyn</td>
<td>7.1M</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-23</td>
</tr>
<tr>
<td>Hayes</td>
<td>L-15</td>
</tr>
<tr>
<td>Jackson</td>
<td>L-14</td>
</tr>
<tr>
<td>Curry</td>
<td>L-93</td>
</tr>
<tr>
<td>Smith</td>
<td>L-11</td>
</tr>
<tr>
<td>Williams</td>
<td>L-17</td>
</tr>
<tr>
<td>Adams</td>
<td>L-16</td>
</tr>
</tbody>
</table>

### Loan

<table>
<thead>
<tr>
<th>bname</th>
<th>ino</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-15</td>
<td>1500</td>
</tr>
<tr>
<td>Downtown</td>
<td>L-14</td>
<td>1500</td>
</tr>
<tr>
<td>Mianus</td>
<td>L-93</td>
<td>500</td>
</tr>
<tr>
<td>R.H.</td>
<td>L-11</td>
<td>900</td>
</tr>
<tr>
<td>Perry</td>
<td>L-16</td>
<td>1300</td>
</tr>
</tbody>
</table>
# E/R Diagrams & Relations

<table>
<thead>
<tr>
<th>E/R</th>
<th>Relational Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Weak Entity Sets</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>E/R</th>
<th>Relational Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>E₁ = (a₁, ..., aₙ)</td>
<td></td>
</tr>
<tr>
<td>E₂ = (a₁, b₁, ..., bₘ)</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></td>
<td><strong>Multivalued Attributes</strong></td>
</tr>
</tbody>
</table>

**Emp** = (ssn, name)

**Emp-Depts** = (ssn, dept)

<table>
<thead>
<tr>
<th>Emp</th>
<th>Emp-Depts</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssn</td>
<td>dept</td>
</tr>
<tr>
<td>001</td>
<td>Acct</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ssn</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>Smith</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ssn</th>
<th>dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>Sales</td>
</tr>
</tbody>
</table>

Emp

Emp-Depts
E/R Diagrams & Relations

Subclasses

Method 1:
\[ E = (a_1, \ldots, a_n) \]
\[ E_1 = (a_1, b_1, \ldots, b_m) \]
\[ E_2 = (a_1, c_1, \ldots, c_k) \]
## E/R Diagrams & Relations

<table>
<thead>
<tr>
<th>E/Rb</th>
<th>Relational Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Subclasses</strong></td>
<td><strong>Method 1:</strong></td>
</tr>
<tr>
<td><img src="image" alt="Diagram" /></td>
<td>$E = (a_1, \ldots, a_n)$</td>
</tr>
<tr>
<td></td>
<td>$E_1 = (a_1, b_1, \ldots, b_m)$</td>
</tr>
<tr>
<td></td>
<td>$E_2 = (a_1, c_1, \ldots, c_k)$</td>
</tr>
<tr>
<td><strong>Method 2:</strong></td>
<td>$E_1 = (a_1, \ldots, a_n, b_1, \ldots, b_m)$</td>
</tr>
<tr>
<td></td>
<td>$E_2 = (a_1, \ldots, a_n, c_1, \ldots, c_k)$</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)

**Q:** When is method 2 not possible?

**A:** When subclassing is partial
Subclass Distinctions:

3. **Total vs. Partial Membership**

- **Total:** Every entity of superclass belongs to a subclass 
  e.g.,

  - Person
    - Isa
    - Child: age < 18
    - Adult: age $\geq$ 18
    - Senior: age $\geq$ 65

- **Partial:** Some entities of superclass do not belong to any 
  subclass (e.g., Suppose Adult condition is age $\geq$ 21)
E/R Data Model

Extensions to the Model: Aggregation

- E/R: No relationships between relationships
  - E.g.: Associate loan officers with Borrow relationship set

associate Loan Officer with Loan?
- What if we want a loan officer for every (customer, loan) pair?