Chapter 6: Entity-Relationship Model
Chapter 6: Entity-Relationship Model

- Design Process
- Modeling
- Constraints
- E-R Diagram
- Design Issues
- Weak Entity Sets
- Extended E-R Features
- Design of the Bank Database
- Reduction to Relation Schemas
- Database Design
- UML
Modeling

- A *database* can be modeled as:
  - a collection of entities,
  - relationship among entities.
- An **entity** is an object that exists and is distinguishable from other objects.
  - Example: specific person, company, event, plant
- Entities have **attributes**
  - Example: people have *names* and *addresses*
- An **entity set** is a set of entities of the same type that share the same properties.
  - Example: set of all persons, companies, trees, holidays
## Entity Sets `customer` and `loan`

<table>
<thead>
<tr>
<th>customer_id</th>
<th>customer_</th>
<th>customer_</th>
<th>customer_</th>
<th>loan_</th>
<th>amount</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>name</td>
<td>street</td>
<td>city</td>
<td>number</td>
<td></td>
</tr>
<tr>
<td>321-12-3123</td>
<td>Jones</td>
<td>Main</td>
<td>Harrison</td>
<td></td>
<td></td>
</tr>
<tr>
<td>019-28-3746</td>
<td>Smith</td>
<td>North</td>
<td>Rye</td>
<td></td>
<td></td>
</tr>
<tr>
<td>677-89-9011</td>
<td>Hayes</td>
<td>Main</td>
<td>Harrison</td>
<td></td>
<td></td>
</tr>
<tr>
<td>555-55-5555</td>
<td>Jackson</td>
<td>Dupont</td>
<td>Woodside</td>
<td></td>
<td></td>
</tr>
<tr>
<td>244-66-8800</td>
<td>Curry</td>
<td>North</td>
<td>Rye</td>
<td></td>
<td></td>
</tr>
<tr>
<td>963-96-3963</td>
<td>Williams</td>
<td>Nassau</td>
<td>Princeton</td>
<td></td>
<td></td>
</tr>
<tr>
<td>335-57-7991</td>
<td>Adams</td>
<td>Spring</td>
<td>Pittsfield</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A **relationship** is an association among several entities

Example:

- **Hayes**
- **depositor**
- **A-102**

*customer* entity | *relationship set* | *account* entity

A **relationship set** is a mathematical relation among \( n \geq 2 \) entities, each taken from entity sets

\[
\{(e_1, e_2, \ldots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}
\]

where \((e_1, e_2, \ldots, e_n)\) is a relationship

- Example:
  - \((\text{Hayes, A-102}) \in \text{depositor}\)
Relationship Set borrower

<table>
<thead>
<tr>
<th>borrower</th>
<th>customer</th>
<th>loan</th>
</tr>
</thead>
<tbody>
<tr>
<td>321-12-3123</td>
<td>Jones</td>
<td>Main</td>
</tr>
<tr>
<td>019-28-3746</td>
<td>Smith</td>
<td>North</td>
</tr>
<tr>
<td>677-89-9011</td>
<td>Hayes</td>
<td>Main</td>
</tr>
<tr>
<td>555-55-5555</td>
<td>Jackson</td>
<td>Dupont</td>
</tr>
<tr>
<td>244-66-8800</td>
<td>Curry</td>
<td>North</td>
</tr>
<tr>
<td>963-96-3963</td>
<td>Williams</td>
<td>Nassau</td>
</tr>
<tr>
<td>335-57-7991</td>
<td>Adams</td>
<td>Spring</td>
</tr>
</tbody>
</table>
A **relationship** set can also be an entity set.

For instance, the **depositor** relationship set between entity sets `customer` and `account` may have the attribute `access-date`.
Degree of a Relationship Set

- Refers to number of entity sets that participate in a relationship set.

- Relationship sets that involve two entity sets are **binary** (or degree two). Generally, most relationship sets in a database system are binary.

- Relationship sets may involve more than two entity sets.

  - Example: Suppose employees of a bank may have jobs (responsibilities) at multiple branches, with different jobs at different branches. Then there is a ternary relationship set between entity sets *employee, job, and branch*

- Relationships between more than two entity sets are rare. Most relationships are binary. (More on this later.)
Attributes

- An entity is represented by a set of attributes, that is descriptive properties possessed by all members of an entity set.
  Example:
  
  \[
  \text{customer} = (\text{customer\_id, customer\_name, customer\_street, customer\_city}) \\
  \text{loan} = (\text{loan\_number, amount})
  \]

- **Domain** – the set of permitted values for each attribute

- Attribute types:
  - *Simple* and *composite* attributes.
  - *Single-valued* and *multi-valued* attributes
    - Example: multivalued attribute: \text{phone\_numbers}
  - *Derived* attributes
    - Can be computed from other attributes
    - Example: \text{age, given date\_of\_birth}
Composite Attributes

Composite Attributes

name

Component Attributes

first_name  middle_initial  last_name

Component Attributes

address

street  city  state  postal_code

Component Attributes

street_number  street_name  apartment_number
Mapping Cardinality Constraints

- Express the number of entities to which another entity can be associated via a relationship set.
- Most useful in describing binary relationship sets.
- For a binary relationship set the mapping cardinality must be one of the following types:
  - One to one
  - One to many
  - Many to one
  - Many to many
Mapping Cardinalities

(a) One to one

Note: Some elements in A and B may not be mapped to any elements in the other set

(b) One to many
Mapping Cardinalities

Many to one

Many to many

Note: Some elements in A and B may not be mapped to any elements in the other set
A **super key** of an entity set is a set of one or more attributes whose values uniquely determine each entity.

A **candidate key** of an entity set is a minimal super key

- *Customer_id* is candidate key of *customer*
- *account_number* is candidate key of *account*

Although several candidate keys may exist, one of the candidate keys is selected to be the **primary key**.
Keys for Relationship Sets

- The combination of primary keys of the participating entity sets forms a super key of a relationship set.
  - \((\text{customer}_\text{id}, \text{account}_\text{number})\) is the super key of \textit{depositor}.
  - \textit{NOTE: this means a pair of entity sets can have at most one relationship in a particular relationship set.}
    - Example: if we wish to track all \textit{access\_dates} to each account by each customer, we cannot assume a relationship for each access. We can use a multivalued attribute though.

- Must consider the mapping cardinality of the relationship set when deciding the what are the candidate keys

- Need to consider semantics of relationship set in selecting the \textit{primary key} in case of more than one candidate key.
Rectangles represent entity sets.

Diamonds represent relationship sets.

Lines link attributes to entity sets and entity sets to relationship sets.

Ellipses represent attributes
- Double ellipses represent multivalued attributes.
- Dashed ellipses denote derived attributes.

Underline indicates primary key attributes (will study later)
E-R Diagram With Composite, Multivalued, and Derived Attributes

- First_name
- Last_name
- Middle_initial
- Customer_id
- Phone_number
- Date_of_birth
- Age
- Street_name
- Street_number
- Apartment_number
- Street
- City
- State
- Zip_code
- Address

Customer
Roles

- Entity sets of a relationship need not be distinct.
- The labels “manager” and “worker” are called roles; they specify how employee entities interact via the works_for relationship set.
- Roles are indicated in E-R diagrams by labeling the lines that connect diamonds to rectangles.
- Role labels are optional, and are used to clarify semantics of the relationship.
Cardinality Constraints

- We express cardinality constraints by drawing either a directed line (→), signifying “one,” or an undirected line (—), signifying “many,” between the relationship set and the entity set.

- One-to-one relationship:
  - A customer is associated with at most one loan via the relationship \textit{borrower}
  - A loan is associated with at most one customer via \textit{borrower}
In the one-to-many relationship a loan is associated with at most one customer via borrower, a customer is associated with several (including 0) loans via borrower.
In a many-to-one relationship a loan is associated with several (including 0) customers via borrower, a customer is associated with at most one loan via borrower.
**Many-To-Many Relationship**

- A customer is associated with several (possibly 0) loans via borrower.
- A loan is associated with several (possibly 0) customers via borrower.
Participation of an Entity Set in a Relationship Set

- Total participation (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set
  - E.g. participation of loan in borrower is total
    - every loan must have a customer associated to it via borrower
- Partial participation: some entities may not participate in any relationship in the relationship set
  - Example: participation of customer in borrower is partial
Alternative Notation for Cardinality Limits

- Cardinality limits can also express participation constraints

```
+--------------------------+                +----------------+                +-----------------+
|  customer_name           |                | loan_number     |                | amount           |
| customer_id              |                | customer_city   |                |                 |
| customer_street          |                | customer_city   |                |                 |
|                          |                | loan_number     |                |                 |
+--------------------------+                +----------------+                +-----------------+
```

```graphviz
digraph G {
  node [shape=ellipse];
  customer [label=customer,shape=box];
  borrower [label=1..1,shape=trapezium];
  loan [label=loan,shape=box];
  customer -> borrower [label=0..*];
  borrower -> loan [label=1..1];
}
```
Diagram with a Ternary Relationship

- employee
  - employee_id
  - employee_name
  - employee_street
- telephone_number
- title
- level
- job
- works_on
- branch
  - branch_name
  - branch_city
  - assets
Cardinality Constraints on Ternary Relationship

- We allow at most one arrow out of a ternary (or greater degree) relationship to indicate a cardinality constraint.
- E.g. an arrow from `works_on` to `job` indicates each employee works on at most one job at any branch.
- If there is more than one arrow, there are two ways of defining the meaning.
  - E.g. a ternary relationship `R` between `A`, `B` and `C` with arrows to `B` and `C` could mean:
    1. each `A` entity is associated with a unique entity from `B` and `C` or
    2. each pair of entities from `(A, B)` is associated with a unique `C` entity, and each pair `(A, C)` is associated with a unique `B`
  - Each alternative has been used in different formalisms.
  - To avoid confusion we outlaw more than one arrow.
Design Issues

- **Use of entity sets vs. attributes**
  Choice mainly depends on the structure of the enterprise being modeled, and on the semantics associated with the attribute in question.

- **Use of entity sets vs. relationship sets**
  Possible guideline is to designate a relationship set to describe an action that occurs between entities.

- **Binary versus n-ary relationship sets**
  Although it is possible to replace any nonbinary \((n\text{-ary, for } n > 2)\) relationship set by a number of distinct binary relationship sets, a \(n\)-ary relationship set shows more clearly that several entities participate in a single relationship.

- **Placement of relationship attributes**
Some relationships that appear to be non-binary may be better represented using binary relationships

- E.g. A ternary relationship *parents*, relating a child to his/her father and mother, is best replaced by two binary relationships, *father* and *mother*
  - Using two binary relationships allows partial information (e.g. only mother being known)
- But there are some relationships that are naturally non-binary
  - Example: *works_on*
In general, any non-binary relationship can be represented using binary relationships by creating an artificial entity set.

- Replace $R$ between entity sets $A$, $B$ and $C$ by an entity set $E$, and three relationship sets:
  1. $R_A$, relating $E$ and $A$
  2. $R_B$, relating $E$ and $B$
  3. $R_C$, relating $E$ and $C$

- Create a special identifying attribute for $E$
- Add any attributes of $R$ to $E$
- For each relationship $(a_i, b_i, c_i)$ in $R$, create
  1. a new entity $e_i$ in the entity set $E$
  2. add $(e_i, a_i)$ to $R_A$
  3. add $(e_i, b_i)$ to $R_B$
  4. add $(e_i, c_i)$ to $R_C$

![Diagram](image)
Converting Non-Binary Relationships (Cont.)

- Also need to translate constraints
  - Translating all constraints may not be possible
  - There may be instances in the translated schema that cannot correspond to any instance of \( R \)
    - Exercise: *add constraints to the relationships* \( R_A, R_B \) and \( R_C \) to ensure that a newly created entity corresponds to exactly one entity in each of entity sets \( A, B \) and \( C \)
  - We can avoid creating an identifying attribute by making \( E \) a weak entity set (described shortly) identified by the three relationship sets
Can make access-date an attribute of account, instead of a relationship attribute, if each account can have only one customer.

- That is, the relationship from account to customer is many to one, or equivalently, customer to account is one to many.
How about doing an ER design interactively on the board? Suggest an application to be modeled.
An entity set that does not have a primary key is referred to as a **weak entity set**.

The existence of a weak entity set depends on the existence of a **identifying entity set**

- it must relate to the identifying entity set via a total, one-to-many relationship set from the identifying to the weak entity set
- Identifying relationship depicted using a double diamond

The **discriminator** (or partial key) of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set.

The primary key of a weak entity set is formed by the primary key of the strong entity set on which the weak entity set is existence dependent, plus the weak entity set’s discriminator.
Weak Entity Sets (Cont.)

- We depict a weak entity set by double rectangles.
- We underline the discriminator of a weak entity set with a dashed line.
- payment_number – discriminator of the payment entity set
- Primary key for payment – (loan_number, payment_number)
Note: the primary key of the strong entity set is not explicitly stored with the weak entity set, since it is implicit in the identifying relationship.

If loan_number were explicitly stored, payment could be made a strong entity, but then the relationship between payment and loan would be duplicated by an implicit relationship defined by the attribute loan_number common to payment and loan.
More Weak Entity Set Examples

- In a university, a course is a strong entity and a course_offering can be modeled as a weak entity.
- The discriminator of course_offering would be semester (including year) and section_number (if there is more than one section).
- If we model course_offering as a strong entity we would model course_number as an attribute.
  Then the relationship with course would be implicit in the course_number attribute.
Extended E-R Features: Specialization

- Top-down design process; we designate subgroupings within an entity set that are distinctive from other entities in the set.
- These subgroupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set.
- Depicted by a triangle component labeled ISA (E.g. customer “is a” person).

Attribute inheritance – a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked.
Specialization Example
Extended ER Features: Generalization

- A bottom-up design process – combine a number of entity sets that share the same features into a higher-level entity set.
- Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way.
- The terms specialization and generalization are used interchangeably.
Specialization and Generalization (Cont.)

- Can have multiple specializations of an entity set based on different features.
- E.g. `permanent_employee` vs. `temporary_employee`, in addition to `officer` vs. `secretary` vs. `teller`
- Each particular employee would be
  - a member of one of `permanent_employee` or `temporary_employee`,
  - and also a member of one of `officer`, `secretary`, or `teller`
- The ISA relationship also referred to as **superclass - subclass** relationship
Design Constraints on a Specialization/Generalization

- Constraint on which entities can be members of a given lower-level entity set.
  - condition-defined
    - Example: all customers over 65 years are members of senior-citizen entity set; senior-citizen ISA person.
  - user-defined

- Constraint on whether or not entities may belong to more than one lower-level entity set within a single generalization.
  - Disjoint
    - an entity can belong to only one lower-level entity set
    - Noted in E-R diagram by writing disjoint next to the ISA triangle
  - Overlapping
    - an entity can belong to more than one lower-level entity set
Design Constraints on a Specialization/Generalization (Cont.)

- **Completeness constraint** -- specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within a generalization.
  - **total**: an entity must belong to one of the lower-level entity sets
  - **partial**: an entity need not belong to one of the lower-level entity sets
Consider the ternary relationship `works_on`, which we saw earlier

Suppose we want to record managers for tasks performed by an employee at a branch
Aggregation (Cont.)

- Relationship sets `works_on` and `manages` represent overlapping information
  - Every `manages` relationship corresponds to a `works_on` relationship
  - However, some `works_on` relationships may not correspond to any `manages` relationships
    - So we can’t discard the `works_on` relationship

- Eliminate this redundancy via aggregation
  - Treat relationship as an abstract entity
  - Allows relationships between relationships
  - Abstraction of relationship into new entity

- Without introducing redundancy, the following diagram represents:
  - An employee works on a particular job at a particular branch
  - An employee, branch, job combination may have an associated manager
E-R Diagram With Aggregation

- Employee
- Works_on
- Job
- Branch
- Manages
- Manager
E-R Design Decisions

- The use of an attribute or entity set to represent an object.
- Whether a real-world concept is best expressed by an entity set or a relationship set.
- The use of a ternary relationship versus a pair of binary relationships.
- The use of a strong or weak entity set.
- The use of specialization/generalization – contributes to modularity in the design.
- The use of aggregation – can treat the aggregate entity set as a single unit without concern for the details of its internal structure.
E-R Diagram for a Banking Enterprise
How about doing another ER design interactively on the board?
Summary of Symbols Used in E-R Notation

- **E**: entity set
- **A**: attribute
- **R**: relationship set
- **multivalued attribute**
- **derived attribute**
- **identifying relationship set for weak entity set**
- **primary key**
- **many-to-many relationship**
- **many-to-one relationship**
- **one-to-one relationship**
- **role indicator**
- **ISA (specialization or generalization)**
- **total generalization**
- **disjoint generalization**
Primary keys allow entity sets and relationship sets to be expressed uniformly as relation schemas that represent the contents of the database.

A database which conforms to an E-R diagram can be represented by a collection of schemas.

For each entity set and relationship set there is a unique schema that is assigned the name of the corresponding entity set or relationship set.

Each schema has a number of columns (generally corresponding to attributes), which have unique names.
A strong entity set reduces to a schema with the same attributes.

A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set

$$payment = (loan\_number, payment\_number, payment\_date, payment\_amount)$$
Representing Relationship Sets as Schemas

A many-to-many relationship set is represented as a schema with attributes for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.

Example: schema for relationship set borrower

\[ \text{borrower} = (\text{customer\_id, loan\_number}) \]
Redundancy of Schemas

- Many-to-one and one-to-many relationship sets that are total on the many-side can be represented by adding an extra attribute to the “many” side, containing the primary key of the “one” side.

- Example: Instead of creating a schema for relationship set `account_branch`, add an attribute `branch_name` to the schema arising from entity set `account`.

![Diagram](image-url)
Redundancy of Schemas (Cont.)

- For one-to-one relationship sets, either side can be chosen to act as the "many" side
  - That is, extra attribute can be added to either of the tables corresponding to the two entity sets
- If participation is partial on the "many" side, replacing a schema by an extra attribute in the schema corresponding to the "many" side could result in null values
- The schema corresponding to a relationship set linking a weak entity set to its identifying strong entity set is redundant.
  - Example: The payment schema already contains the attributes that would appear in the loan_payment schema (i.e., loan_number and payment_number).
Composite and Multivalued Attributes

- Composite attributes are flattened out by creating a separate attribute for each component attribute
  - Example: given entity set customer with composite attribute name with component attributes first_name and last_name the schema corresponding to the entity set has two attributes name.first_name and name.last_name

- A multivalued attribute $M$ of an entity $E$ is represented by a separate schema $EM$
  - Schema $EM$ has attributes corresponding to the primary key of $E$ and an attribute corresponding to multivalued attribute $M$
  - Example: Multivalued attribute dependent_names of employee is represented by a schema:
    
    $employee\_dependent\_names = (employee\_id, dname)$
  
    - Each value of the multivalued attribute maps to a separate tuple of the relation on schema $EM$
    
      - For example, an employee entity with primary key 123-45-6789 and dependents Jack and Jane maps to two tuples:
        (123-45-6789, Jack) and (123-45-6789, Jane)
Representing Specialization via Schemas

- **Method 1:**
  - Form a schema for the higher-level entity
  - Form a schema for each lower-level entity set, include primary key of higher-level entity set and local attributes

<table>
<thead>
<tr>
<th>schema</th>
<th>attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>person</td>
<td>name, street, city</td>
</tr>
<tr>
<td>customer</td>
<td>name, credit_rating</td>
</tr>
<tr>
<td>employee</td>
<td>name, salary</td>
</tr>
</tbody>
</table>

- Drawback: getting information about an employee requires accessing two relations, the one corresponding to the low-level schema and the one corresponding to the high-level schema.
Method 2:

- Form a schema for each entity set with all local and inherited attributes

<table>
<thead>
<tr>
<th>schema</th>
<th>attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>person</td>
<td>name, street, city</td>
</tr>
<tr>
<td>customer</td>
<td>name, street, city, credit_rating</td>
</tr>
<tr>
<td>employee</td>
<td>name, street, city, salary</td>
</tr>
</tbody>
</table>

- If specialization is total, the schema for the generalized entity set (person) not required to store information
  - Can be defined as a “view” relation containing union of specialization relations
  - But explicit schema may still be needed for foreign key constraints

- Drawback: street and city may be stored redundantly for people who are both customers and employees
To represent aggregation, create a schema containing

- primary key of the aggregated relationship,
- the primary key of the associated entity set
- any descriptive attributes
For example, to represent aggregation manages between relationship works_on and entity set manager, create a schema

\[\text{manages (employee\_id, branch\_name, title, manager\_name)}\]

Schema works_on is redundant provided we are willing to store null values for attribute manager\_name in relation on schema manages
- **UML**: Unified Modeling Language
- UML has many components to graphically model different aspects of an entire software system
- UML Class Diagrams correspond to E-R Diagram, but several differences.
Summary of UML Class Diagram Notation

1. Entity sets and attributes
   - `customer-name`
   - `customer-street`
   - `customer-id`
   - `customer-city`
   - `customer`

   - `customer-id`
   - `customer-name`
   - `customer-street`
   - `customer-city`

2. Relationships
   - E1 \(\text{role}_1\) R \(\text{role}_2\) E2
   - E1 \(\text{role}_1\) R \(\text{role}_2\) E2
   - E1 \(\text{role}_1\) R \(\text{role}_2\) E2
   - E1 \(\text{role}_1\) R \(\text{role}_2\) E2

   - \(R\) \(a_1\) \(a_2\)
Entity sets are shown as boxes, and attributes are shown within the box, rather than as separate ellipses in E-R diagrams.

Binary relationship sets are represented in UML by just drawing a line connecting the entity sets. The relationship set name is written adjacent to the line.

The role played by an entity set in a relationship set may also be specified by writing the role name on the line, adjacent to the entity set.

The relationship set name may alternatively be written in a box, along with attributes of the relationship set, and the box is connected, using a dotted line, to the line depicting the relationship set.

Non-binary relationships drawn using diamonds, just as in ER diagrams.
UML Class Diagram Notation (Cont.)

3. Cardinality constraints

4. Generalization and Specialization

*Note reversal of position in cardinality constraint depiction

*Generalization can use merged or separate arrows independent of disjoint/overlapping
Cardinality constraints are specified in the form $l..h$, where $l$ denotes the minimum and $h$ the maximum number of relationships an entity can participate in.

Beware: the positioning of the constraints is exactly the reverse of the positioning of constraints in E-R diagrams.

The constraint 0..* on the $E_2$ side and 0..1 on the $E_1$ side means that each $E_2$ entity can participate in at most one relationship, whereas each $E_1$ entity can participate in many relationships; in other words, the relationship is many to one from $E_2$ to $E_1$.

Single values, such as 1 or * may be written on edges; The single value 1 on an edge is treated as equivalent to 1..1, while * is equivalent to 0..*.
End of Chapter 2
E-R Diagram for Exercise 2.10

- author: name, address, written-by (URL)
- publisher: name, address, phone, published-by (URL)
- customer: name, address, email, phone
- shopping-basket: basketID
- basket-of: basketID
- stocks: number, address, phone
- warehouse: code
E-R Diagram for Exercise 2.15

(a) 

(b) 

(c)
E-R Diagram for Exercise 2.22
E-R Diagram for Exercise 2.15

(a) Diagram with entities A, B, and C connected through relations R.

(b) Diagram with entities A, E, and C connected through relations R_B and R_A.

(c) Diagram with entities B, A, and C connected through relations R_1, R_2, and R_3.
If the existence of entity $x$ depends on the existence of entity $y$, then $x$ is said to be *existence dependent* on $y$.

- $y$ is a *dominant entity* (in example below, *loan*)
- $x$ is a *subordinate entity* (in example below, *payment*)

If a *loan* entity is deleted, then all its associated *payment* entities must be deleted also.
<table>
<thead>
<tr>
<th>loan_number</th>
<th>amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>L-11</td>
<td>900</td>
</tr>
<tr>
<td>L-14</td>
<td>1500</td>
</tr>
<tr>
<td>L-15</td>
<td>1500</td>
</tr>
<tr>
<td>L-16</td>
<td>1300</td>
</tr>
<tr>
<td>L-17</td>
<td>1000</td>
</tr>
<tr>
<td>L-23</td>
<td>2000</td>
</tr>
<tr>
<td>L-93</td>
<td>500</td>
</tr>
</tbody>
</table>
Figure 6.27

Diagram showing relationships between account, branch-name, branch-city, account-branch, balance, and assets.
Figure 6.29
Alternative E-R Notations

Figure 6.24

entity set E with
attributes A1, A2, A3
and primary key A1

many_to_many
relationship

many_to_one
relationship

one_to_one
relationship