Chapter 2

The Entity-Relationship Model

The **E-R** (entity-relationship) data model views the real world as a set of basic **objects** (entities) and **relationships** among these objects.

It is intended primarily for the DB design process by allowing the specification of an **enterprise scheme**. This represents the overall logical structure of the DB.

2.1 Entities and Entity Sets

- An **entity** is an object that exists and is distinguishable from other objects. For instance, John Harris with S.I.N. 890-12-3456 is an entity, as he can be uniquely identified as one particular person in the universe.
- An entity may be **concrete** (a person or a book, for example) or **abstract** (like a holiday or a concept).
- An **entity set** is a set of entities of the same type (e.g., all persons having an account at a bank).
- Entity sets **need not be disjoint**. For example, the entity set *employee* (all employees of a bank) and the entity set *customer* (all customers of the bank) may have members in common.
- An entity is represented by a set of **attributes**.
  - E.g. name, S.I.N., street, city for "customer" entity.
  - The **domain** of the attribute is the set of permitted values (e.g. the telephone number must be seven positive integers).
- Formally, <u>an attribute is a **function**</u> which maps an entity set into a domain.
  - Every entity is described by a set of (attribute, data value) pairs.
  - There is one pair for each attribute of the entity set.
  - E.g. a particular *customer* entity is described by the set {(name, Harris), (S.I.N., 890-123-456), (street, North), (city, Georgetown)}.

An analogy can be made with the programming language notion of type definition.

- The concept of an **entity set** corresponds to the programming language **type definition**.
- A variable of a given type has a particular value at a point in time.
- Thus, a programming language variable corresponds to an **entity** in the E-R model.

Figure 2-1 shows two entity sets.

We will be dealing with five entity sets in this section:

- *branch*, the set of all branches of a particular bank. Each branch is described by the attributes *branch-name*, *branch-city* and *assets*.
- *customer*, the set of all people having an account at the bank. Attributes are *customer-name*, *S.I.N.*, *street* and *customer-city*.
- *employee*, with attributes *employee-name* and *phone-number*.
- *account*, the set of all accounts maintained in the bank. Attributes are *account-number* and *balance*.
- *transaction*, the set of all account transactions executed in the bank. Attributes are *transaction-number*, *date* and *amount*.

2.2 Relationships & Relationship Sets

A **relationship** is an association between several entities.

A **relationship set** is a set of relationships of the same type.

Formally it is a mathematical relation on $n \geq 2$ (possibly non-distinct) sets.

If $E_1, E_2, \ldots, E_n$ are entity sets, then a relationship set R is a **subset** of

$$\{(e_1, e_2, \ldots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in En\}$$

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

For example, consider the two entity sets *customer* and *account*. (Fig. 2.1 in the text). We define the relationship *CustAcct* to denote the association between customers and their accounts. This is a **binary** relationship set (see Figure 2.2 in the text).

Going back to our formal definition, the relationship set *CustAcct* is a subset of all the possible customer and account pairings.

This is a binary relationship. Occasionally there are relationships involving more than two entity sets.

The **role** of an entity is the function it plays in a relationship. For example, the relationship *works-for* could be ordered pairs of *employee* entities. The first employee takes the role of manager, and the second one will take the role of worker.

A relationship may also have **descriptive** attributes. For example, *date* (last date of account access) could be an attribute of the *CustAcct* relationship set.

2.3 Attributes

It is possible to define a set of entities and the relationships among them in a number of different ways. The main difference is in how we deal with attributes.

- Consider the entity set *employee* with attributes *employee-name* and *phone-number*.
- We could argue that the phone be treated as an entity itself, with attributes *phone-number* and *location*.
- Then we have two entity sets, and the relationship set *EmpPhn* defining the association between employees and their phones.
- This new definition allows employees to have several (or zero) phones.
- New definition may more accurately reflect the real world.
- We cannot extend this argument easily to making *employee-name* an entity.

The question of what constitutes an entity and what constitutes an attribute depends mainly on the structure of the real world situation being modeled, and the semantics associated with the attribute in question.

2.4 Mapping Constraints

An E-R scheme may define certain constraints to which the contents of a database must conform.

- **Mapping Cardinalities:** express the number of entities to which another entity can be associated via a relationship. For binary relationship sets between entity sets A and B, the mapping cardinality must be one of:

  1. **One-to-one**: An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A. (Figure 2.3)
  2. **One-to-many**: An entity in A is associated with any number in B. An entity in B is associated with at most one entity in A. (Figure 2.4)
  3. **Many-to-one**: An entity in A is associated with at most one entity in B. An entity in B is associated with any number in A. (Figure 2.5)
  4. **Many-to-many**: Entities in A and B are associated with any number from each other. (Figure 2.6)

  The appropriate mapping cardinality for a particular relationship set depends on the real world being modeled. (Think about the *CustAcct* relationship...)

- **Existence Dependencies:** if the existence of entity X depends on the existence of entity Y, then X is said to be **existence dependent** on Y. (Or we say that Y is the **dominant** entity and X is the **subordinate** entity.)

  For example,

  - Consider *account* and *transaction* entity sets, and a relationship *log* between them.
  - This is one-to-many from account to transaction.
  - If an *account* entity is deleted, its associated *transaction* entities must also be deleted.
  - Thus *account* is dominant and *transaction* is subordinate.

2.5 Keys

Differences between entities must be expressed in terms of attributes.

- A **superkey** is a set of one or more attributes which, taken collectively, allow us to identify uniquely an entity in the entity set.
- For example, in the entity set *customer*, *customer-name* and *S.I.N.* is a superkey.
- Note that *customer-name* alone is not, as two customers could have the same name.
- A superkey may contain extraneous attributes, and we are often interested in the smallest superkey. A superkey for which no subset is a superkey is called a **candidate key**.
- In the example above, *S.I.N.* is a candidate key, as it is minimal, and uniquely identifies a customer entity.
- A **primary key** is a candidate key (there may be more than one) chosen by the DB designer to identify entities in an entity set.

An entity set that does not possess sufficient attributes to form a primary key is called a **weak entity set.** One that does have a primary key is called a **strong entity set**.

For example,

- The entity set *transaction* has attributes *transaction-number*, *date* and *amount.*
- Different transactions on different accounts could share the same number.

- These are not sufficient to form a primary key (uniquely identify a transaction).
- Thus *transaction* is a weak entity set.

For a weak entity set to be meaningful, it must be part of a one-to-many relationship set. This relationship set should have no descriptive attributes. (Why?)

The idea of strong and weak entity sets is related to the existence dependencies seen earlier.

- Member of a strong entity set is a dominant entity.
- Member of a weak entity set is a subordinate entity.

A weak entity set does not have a primary key, but we need a means of distinguishing among the entities.

The **discriminator** of a weak entity set is a set of attributes that allows this distinction to be made.

The **primary key of a weak entity set** is formed by taking the primary key of the strong entity set on which its existence depends (see Mapping Constraints) plus its **discriminator**.

To illustrate:

- *transaction* is a weak entity. It is existence-dependent on *account*.
- The primary key of *account* is *account-number*.
- *transaction-number* distinguishes transaction entities within the same account (and is thus the discriminator).
- So the primary key for *transaction* would be *(account-number, transaction-number)*.

Just Remember: The primary key of a weak entity is found by taking the primary key of the strong entity on which it is existence-dependent, plus the discriminator of the weak entity set.

2.6 Primary Keys for Relationship Sets

The attributes of a relationship set are the attributes that comprise the primary keys of the entity sets involved in the relationship set.

For example:

- *S.I.N.* is the primary key of *customer*, and
- *account-number* is the primary key of *account*.
- The attributes of the relationship set *custacct* are then *(account-number, S.I.N.)*.

This is enough information to enable us to relate an account to a person.

If the relationship has descriptive attributes, those are also included in its attribute set. For example, we might add the attribute *date* to the above relationship set, signifying the date of last access to an account by a particular customer.

Note that this attribute cannot instead be placed in either entity set as it relates to both a customer and an account, and the relationship is many-to-many.

The primary key of a relationship set R depends on the mapping cardinality and the presence of descriptive attributes.

With no descriptive attributes:

- **many-to-many:** all attributes in R.
- **one-to-many:** primary key for the "many" entity.

Descriptive attributes may be added, depending on the mapping cardinality and the semantics involved (see text).

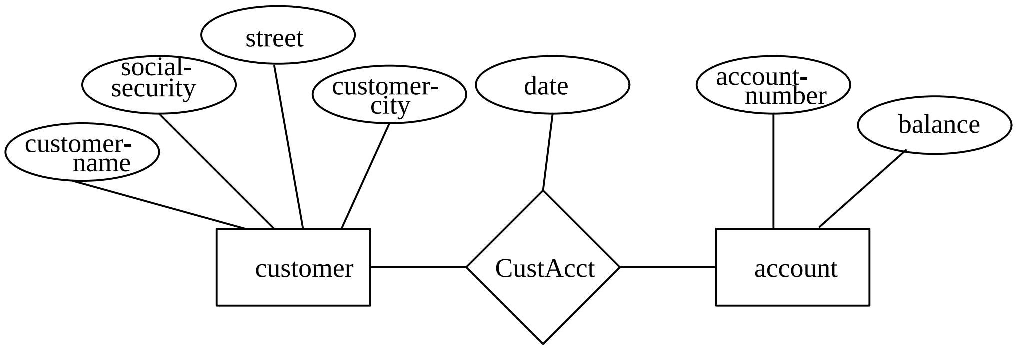

Figure 2.7: An E-R diagram

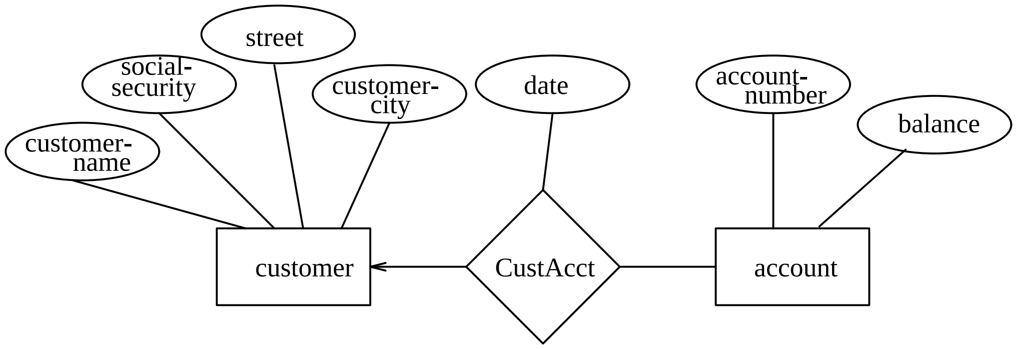

Figure 2.8: One-to-many from *customer* to *account*

2.7 The Entity Relationship Diagram

We can express the overall logical structure of a database **graphically** with an E-R diagram.

Its components are:

- **rectangles** representing entity sets.
- **ellipses** representing attributes.
- **diamonds** representing relationship sets.
- **lines** linking attributes to entity sets and entity sets to relationship sets.

In the text, lines may be directed (have an arrow on the end) to signify mapping cardinalities for relationship sets. Figures 2.8 to 2.10 show some examples.

Go back and review mapping cardinalities. They express the number of entities to which an entity can be associated via a relationship.

The arrow positioning is simple once you get it straight in your mind, so do some examples. Think of the arrow head as pointing to the entity that "one" refers to.

2.8 Other Styles of E-R Diagram

The text uses one particular style of diagram. Many variations exist.

Some of the variations you will see are:

- Diamonds being omitted – a link between entities indicates a relationship.

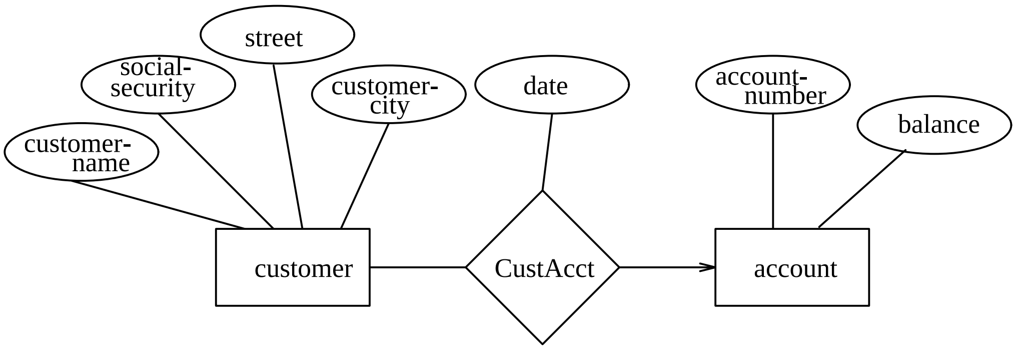

Figure 2.9: Many-to-one from *customer* to *account*

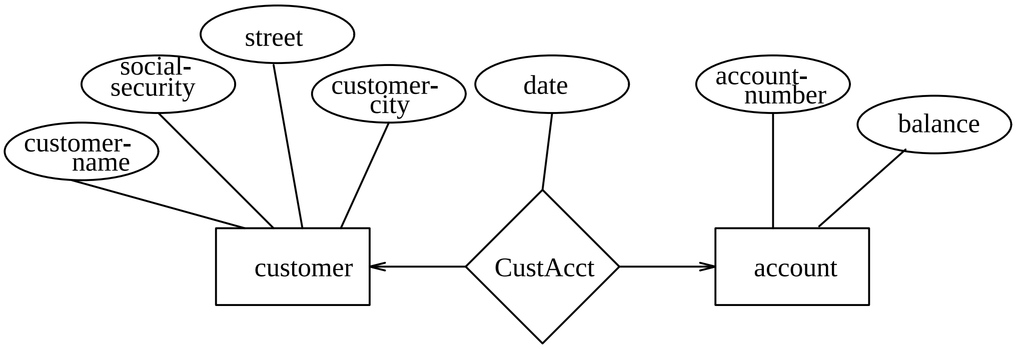

Figure 2.10: One-to-one from *customer* to *account*

- Less symbols, clearer picture.
- What happens with descriptive attributes?
- In this case, we have to create an **intersection entity** to possess the attributes.

- Numbers instead of arrowheads indicating cardinality.
  - Symbols, 1, n and m used.
  - E.g. 1 to 1, 1 to n, n to m.
  - Easier to understand than arrowheads.

- A range of numbers indicating **optionality** of relationship. (See Elmasri & Navathe, p 58.)
  - E.g (0,1) indicates minimum zero (optional), maximum 1.
  - Can also use (0,n), (1,1) or (1,n).
  - Typically used on near end of link – confusing at first, but gives more information.
  - E.g. entity 1 (0,1) —— (1,n) entity 2 indicates that entity 1 is related to between 0 and 1 occurrences of entity 2 (optional).
  - Entity 2 is related to at least 1 and possibly many occurrences of entity 1 (mandatory).

- **Multivalued** attributes may be indicated in some manner.
  - Means attribute can have more than one value.
  - E.g. hobbies.
  - Has to be normalized later on.

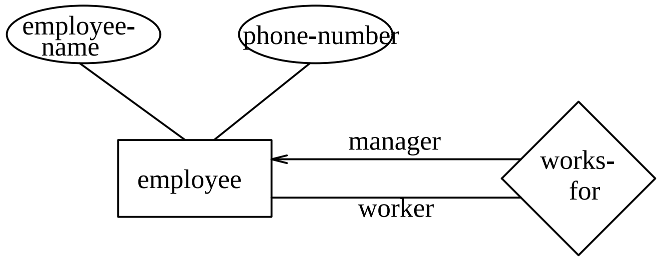

Figure 2.11: E-R diagram with role indicators

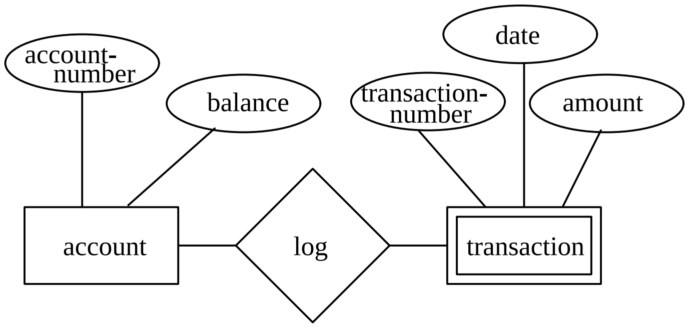

Figure 2.12: E-R diagram with a weak entity set

- **Extended E-R diagrams** allowing more details/constraints in the real world to be recorded. (See Elmasri & Navathe, chapter 21.)
  - Composite attributes.
  - Derived attributes.
  - Subclasses and superclasses.
  - Generalization and specialization.

Roles in E-R Diagrams

The function that an entity plays in a relationship is called its **role**. Roles are normally explicit and not specified.

They are useful when the meaning of a relationship set needs clarification.

For example, the entity sets of a relationship may not be distinct. The relationship *works-for* might be ordered pairs of *employees* (first is manager, second is worker).

In the E-R diagram, this can be shown by labelling the lines connecting entities (rectangles) to relationships (diamonds). (See figure 2.11).

Weak Entity Sets in E-R Diagrams

A weak entity set is indicated by a doubly-outlined box. For example, the previously-mentioned weak entity set *transaction* is dependent on the strong entity set *account* via the relationship set *log*.

Figure 2.12) shows this example.

Nonbinary Relationships

Non-binary relationships can easily be represented. Figure 2.13) shows an example.

This E-R diagram says that a customer may have several accounts, each located in a specific bank branch, and that an account may belong to several different customers.

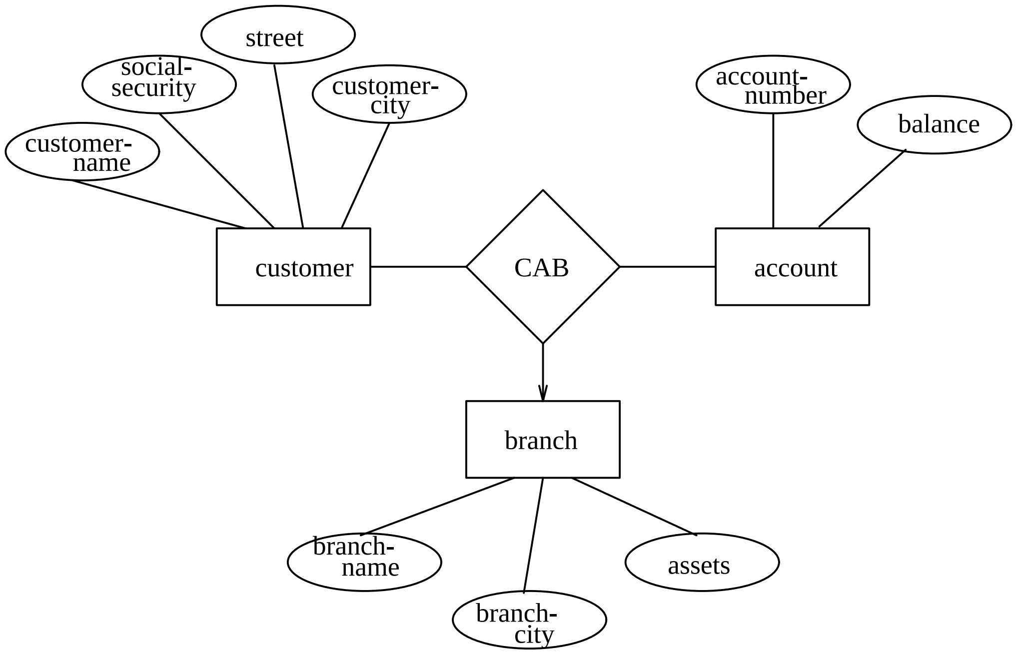

Figure 2.13: E-R diagram with a ternary relationship

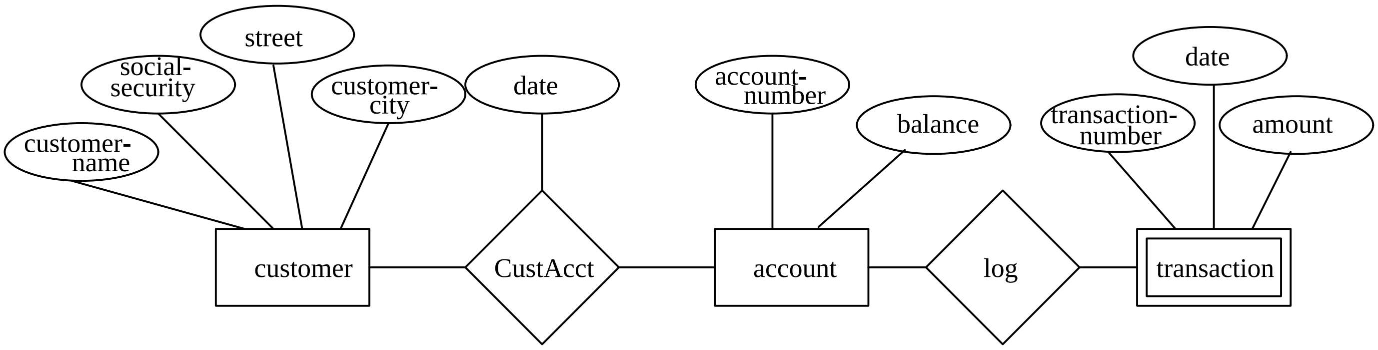

Figure 2.14: E-R diagram with strong and weak entity sets

2.9 Reducing E-R Diagrams to Tables

A database conforming to an E-R diagram can be represented by a collection of tables. We'll use the E-R diagram of Figure 2.14) as our example.

For each entity set and relationship set, there is a **unique** table which is assigned the name of the corresponding set. Each table has a number of columns with unique names. (E.g. Figs. 2.14 - 2.18 in the text).

2.9.1 Representation of Strong Entity Sets

We use a table with one column for each attribute of the set. Each row in the table corresponds to one entity of the entity set. For the entity set *account*, see the table of figure 2.14.

We can add, delete and modify rows (to reflect changes in the real world).

A row of a table will consist of an n-tuple where n is the number of attributes.

Actually, the table contains a subset of the set of all possible rows. We refer to the set of all possible rows as the **cartesian product** of the sets of all attribute values.

We may denote this as

$$D_1 \times D_2 \text{ or } \times_{i=1}^{2} D_i$$

for the account table, where D_1 and D_2 denote the set of all account numbers and all account balances, respectively.

In general, for a table of n columns, we may denote the cartesian product of $D_1, D_2, \ldots, D_n$ by

$$\times_{i=1}^{n} D_i$$

2.9.2 Representation of Weak Entity Sets

For a weak entity set, we add columns to the table corresponding to the primary key of the strong entity set on which the weak set is dependent.

For example, the weak entity set *transaction* has three attributes: *transaction-number*, *date* and *amount*. The primary key of *account* (on which *transaction* depends) is *account-number*. This gives us the table of figure 2.16.

2.9.3 Representation of Relationship Sets

Let R be a relationship set involving entity sets $E_1, E_2, \ldots, E_m$.

The table corresponding to the relationship set R has the following attributes:

$$\bigcup_{i=1}^{m} \text{primary-key}(E_i)$$

If the relationship has k descriptive attributes, we add them too:

$$\bigcup_{i=1}^{m} \text{primary-key}(E_i) \bigcup \{a_1, a_2, \ldots, a_k\}$$

An example:

- The relationship set *CustAcct* involves the entity sets *customer* and *account*.
- Their respective primary keys are *S.I.N.* and *account-number*.
- *CustAcct* also has a descriptive attribute, *date*.
- This gives us the table of figure 2.17.

Non-binary Relationship Sets

The ternary relationship of Figure 2.13 gives us the table of figure 2.18. As required, we take the primary keys of each entity set. There are no descriptive attributes in this example.

Linking a Weak to a Strong Entity

These relationship sets are many-to-one, and have no descriptive attributes. The primary key of the weak entity set is the primary key of the strong entity set it is existence-dependent on, plus its discriminator.

The table for the relationship set would have the same attributes, and is thus redundant.

2.10 Generalization

Consider extending the entity set *account* by classifying accounts as being either *savings-account* or *chequing-account*.

Each of these is described by the attributes of *account* plus additional attributes. (*savings* has *interest-rate* and *chequing* has *overdraft-amount*.)

We can express the similarities between the entity sets by **generalization**. This is the process of forming containment relationships between a **higher-level** entity set and one or more **lower-level** entity sets.

In E-R diagrams, generalization is shown by a triangle, as shown in Figure 2.19.

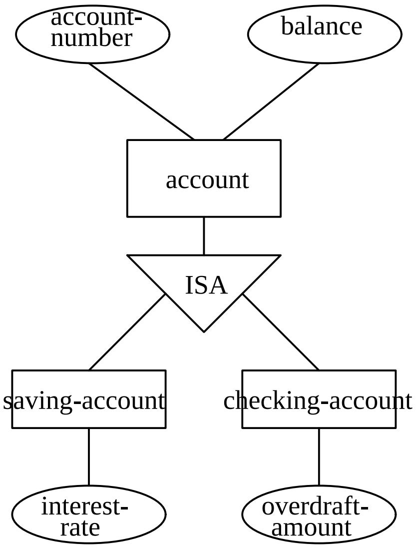

Figure 2.19: Generalization

- Generalization hides differences and emphasizes similarities.
- Distinction made through **attribute inheritance**.
- Attributes of higher-level entity are inherited by lower-level entities.
- Two methods for conversion to a table form:
  - Create a table for the high-level entity, plus tables for the lower-level entities containing also their specific attributes.
  - Create only tables for the lower-level entities.

2.11 Aggregation

The E-R model cannot express relationships among relationships.

When would we need such a thing?

Consider a DB with information about employees who work on a particular project and use a number of machines doing that work. We get the E-R diagram shown in Figure 2.20.

Relationship sets *work* and *uses* could be combined into a single set. However, they shouldn't be, as this would obscure the logical structure of this scheme.

The solution is to use **aggregation**.

- An abstraction through which relationships are treated as higher-level entities.
- For our example, we treat the relationship set *work* and the entity sets *employee* and *project* as a higher-level **entity set** called *work*.
- Figure 2.21 shows the E-R diagram with aggregation.

Transforming an E-R diagram with aggregation into tabular form is easy. We create a table for each entity and relationship set as before.

The table for relationship set *uses* contains a column for each attribute in the primary key of *machinery* and *work*.

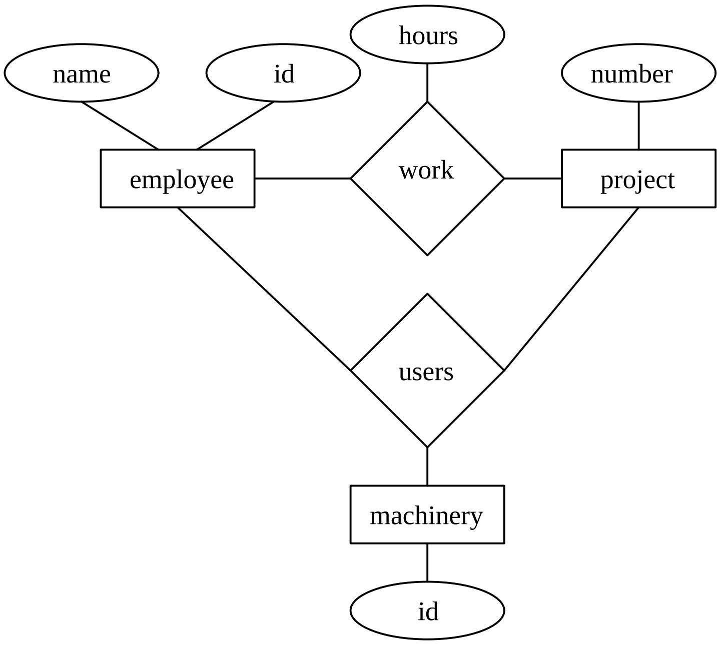

Figure 2.20: E-R diagram with redundant relationships

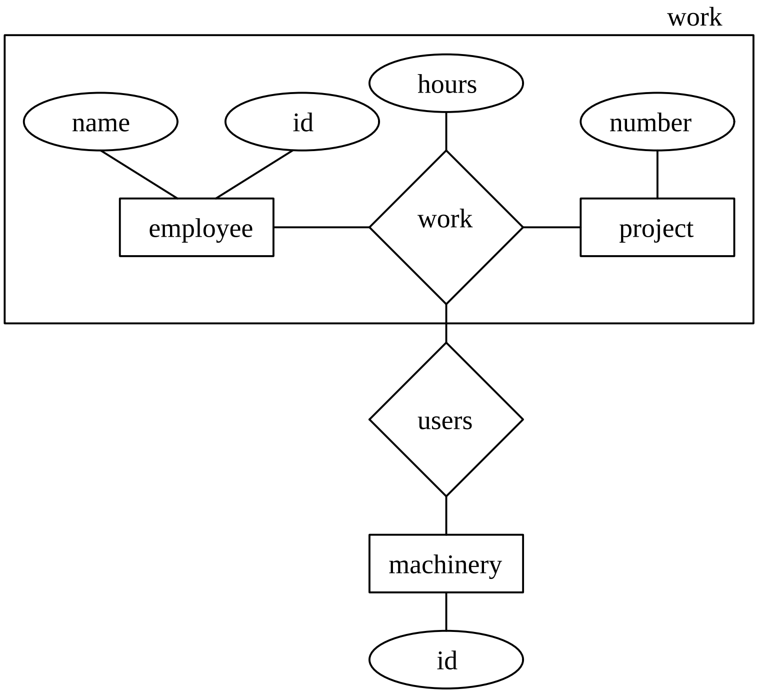

Figure 2.21: E-R diagram with aggregation

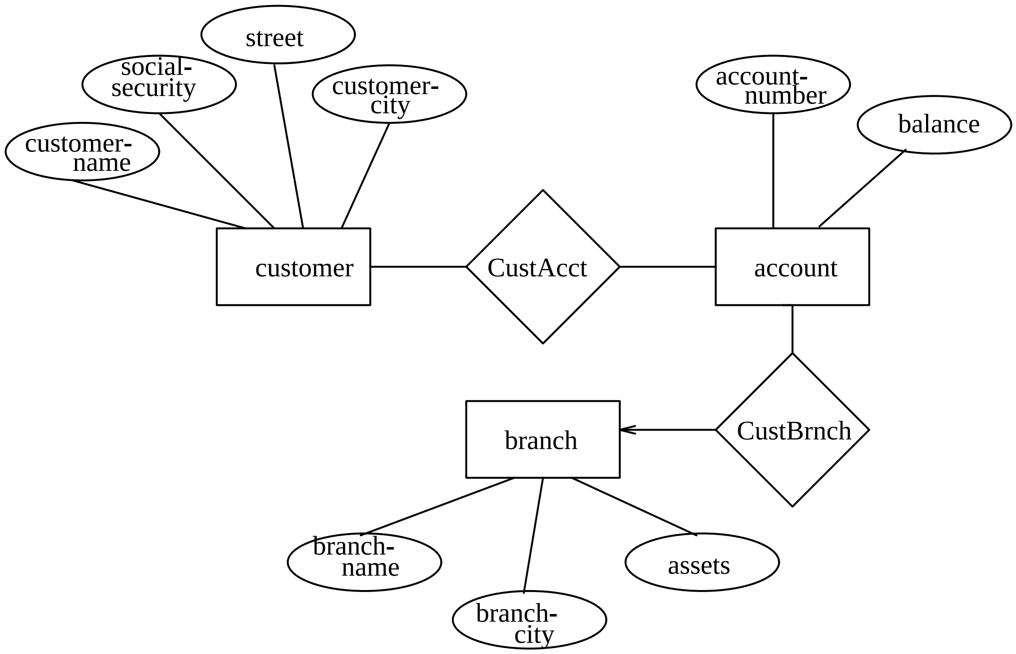

Figure 2.22: Representation of Figure 2.13 using binary relationships

2.12 Design of an E-R Database Scheme

The E-R data model provides a wide range of choice in designing a database scheme to accurately model some real-world situation.

Some of the decisions to be made are

- Using a ternary relationship versus two binary relationships.
- Whether an entity set or a relationship set best fit a real-world concept.
- Whether to use an attribute or an entity set.
- Use of a strong or weak entity set.
- Appropriateness of generalization.
- Appropriateness of aggregation.

2.12.1 Mapping Cardinalities

The ternary relationship of Figure 2.13 could be replaced by a pair of binary relationships, as shown in Figure 2.22.

However, there is a distinction between the two representations:

- In Figure 2.13, relationship between a customer and account can be made only if there is a corresponding branch.
- In Figure 2.22, an account can be related to either a customer or a branch alone.
- The design of figure 2.13 is more appropriate, as in the banking world we expect to have an account relate to both a customer and a branch.

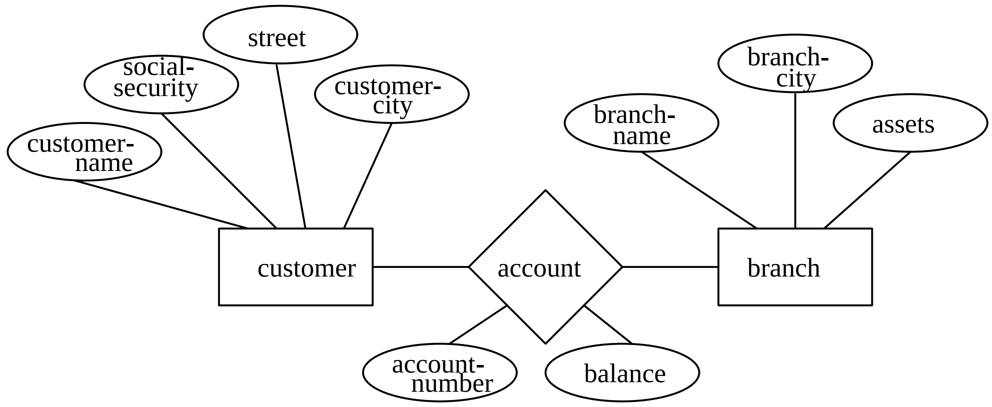

Figure 2.23: E-R diagram with *account* as a relationship set

2.12.2 Use of Entity or Relationship Sets

It is not always clear whether an object is best represented by an entity set or a relationship set.

- Both Figure 2.13 and Figure 2.22 show *account* as an entity.
- Figure 2.23 shows how we might model an account as a relationship between a customer and a branch.
- This new representation cannot model adequately the situation where customers may have joint accounts. (Why not?)
- If every account is held by only one customer, this method works.

2.12.3 Use of Extended E-R Features

We have seen weak entity sets, generalization and aggregation. Designers must decide when these features are appropriate.

- Strong entity sets and their dependent weak entity sets may be regarded as a single "object" in the database, as weak entities are existence-dependent on a strong entity.
- It is possible to treat an aggregated entity set as a single unit without concern for its inner structure details.
- Generalization contributes to modularity by allowing common attributes of similar entity sets to be represented in one place in an E-R diagram.

Excessive use of the features can introduce unnecessary complexity into the design.