**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, an attribute is a **function** 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 created and 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*.

**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 http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter2/_13288_tex2html_wrap752.gif(possibly non-distinct) sets.

If http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter2/_13288_tex2html_wrap754.gifare entity sets, then a relationship set R is a **subset** of

http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter2/_13288_displaymath750.gif

where http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter2/_13288_tex2html_wrap756.gifis 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.

**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.

**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,

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

**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.

[next](http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter2/node8.html)[up](http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter2/node1.html)[previous](http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter2/node6.html)  
**Next:** [The Entity Relationship](http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter2/node8.html) **Up:** [The Entity-Relationship Model](http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter2/node1.html) **Previous:** [Keys](http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter2/node6.html)

# 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 http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter2/_13288_tex2html_wrap758.gifdepends on the mapping cardinality and the presence of descriptive attributes.

With no descriptive attributes:

* **many-to-many:** all attributes in http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter2/_13288_tex2html_wrap758.gif.
* **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).

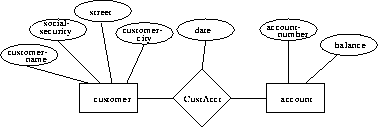
# The Entity Relationship Diagram

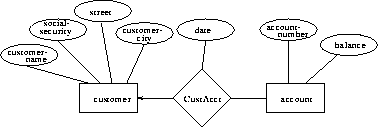
We can express the overall logical structure of a database **graphically** with an E-R diagram.

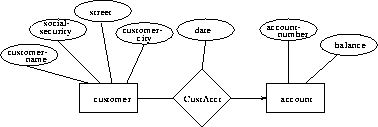
Its components are:

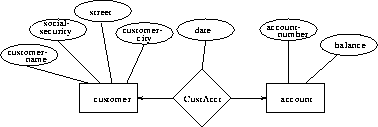
* **rectangles** representing entity sets.
* **ellipses** representing attributes.
* **diamonds** representing relationship sets.
* **lines** linking attributes to entity sets and entity sets to relationship sets.

In the text, lines may be directed (have an arrow on the end) to signify mapping cardinalities for relationship sets. Figures [2.8](http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter2/node8.html#fig281n) to [2.10](http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter2/node8.html#fig2911) show some examples.

     
**Figure 2.7:** An E-R diagram

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

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

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

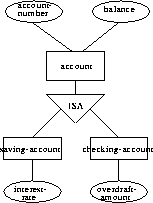
# Generalization

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](http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter2/node14.html#fig219gen).

     
**Figure 2.19:** Generalization

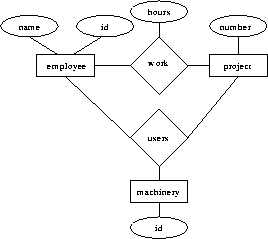
* Generalization hides differences and emphasizes similarities.
* Distinction made through **attribute inheritance**.
* Attributes of higher-level entity are inherited by lower-level entities.
* Two methods for conversion to a table form:
  + Create a table for the high-level entity, plus tables for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

# Aggregation

The E-R model cannot express relationships among relationships.

When would we need such a thing?

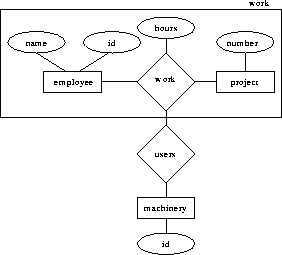
Consider a DB with information about employees who work on a particular project and use a number of machines doing that work. We get the E-R diagram shown in Figure [2.20](http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter2/node15.html#fig220redn).

     
**Figure 2.20:** E-R diagram with redundant relationships

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](http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter2/node15.html#fig221agg) shows the E-R diagram with aggregation.

     
**Figure 2.21:** 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.