Database Design

Entity-Relationship Diagrams

Database Design

What it is:

- Starting from scratch, design the database schema: relation, attributes, keys, foreign keys, constraints etc.

Why it’s hard:

- The database will be in operation for years.
- Updating the schema in production is very hard:
  - schema change modifications are expensive (why?)
  - making the change without introducing any bugs is hard
    - this part is, by far, the most important consideration in practice

Database Design

The database design process can be divided into six basic steps.

1. *Requirements Analysis:* in this step, we must point out

   - What data is stored in the database,
   - What applications must be built on top of it,
   - What operations are most frequent and subject to performance requirements.

   Often, this is an informal process involving discussions with user groups and studying the current environment. Examining existing applications expected to be replaced or complemented by the database system.

Database Design

2. *Conceptual Database Design:* develop a high-level description of the data to be stored in the database, along with the constraints that are known to hold on this data.

3. *Logical Database Design*: convert the conceptual database design into a database schema within the data model of the chosen DBMS.

4. *Schema Refinement*: the schemas developed in step 3 are analyzed for potential problems, then are *normalized*. *Normalization* of a database is based upon some elegant and powerful mathematical theory.

5. *Physical Database Design:* potential workloads and access patterns are simulated to identify potential weaknesses in the conceptual database. This will often cause the creation of additional indices and/or clustering relations. In critical situations, the entire conceptual model will need restructuring.

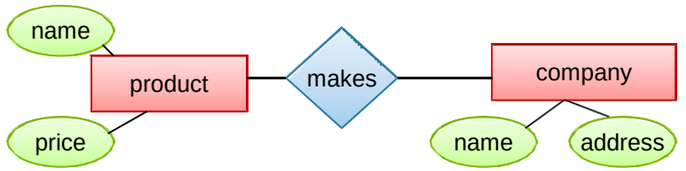
Database Design

6. *Security Design:* Different user groups are identified and their different roles are analyzed so that access patterns to the data can be defined.

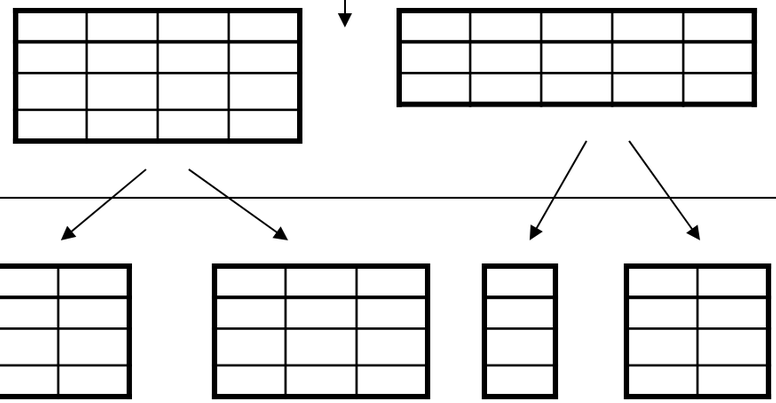
There is often a 7th step in this process with the last step being a *tuning phase*, during which the database is made operational (although it may be through a simulation) and further refinements are made as the system is “tweaked” to provide the expected environment.

Database Design Process

Conceptual Model:



Relational Model:
Tables + constraints
And also functional dep.

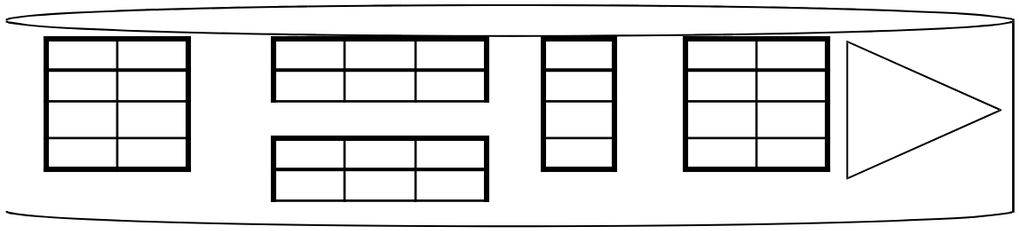


Normalization:
Eliminates anomalies

Conceptual Schema

Physical storage details

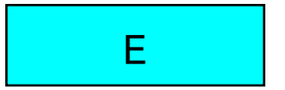
Physical Schema



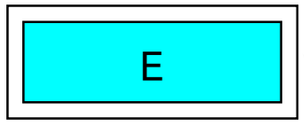
Entity - Relationship Diagrams

- The E-R Diagram employs three basic notions: *entity sets*, *relationship sets*, and *attributes*.
- An ***entity*** is a “thing” or “object” in the real world that is distinguishable from all other objects. An entity may be either concrete, such as a person or a book, or it may be abstract, such as a bank loan, or a holiday, or a concept.
- An entity is represented by a set of ***attributes***. Attributes are descriptive properties or characteristics possessed by an entity. For each attribute, there is a permitted set of values, called the ***domain.***
- An ***entity set*** is a set of entities of the same type that share the same attributes. For example, the set of all persons who are customers at a particular bank can be defined as the entity set *customers*.
- A database contains a collection of entity sets.

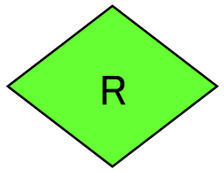
ERD notations



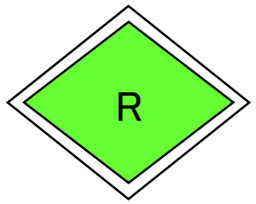
entity set



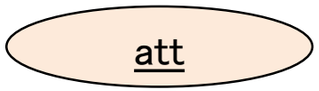
weak entity set



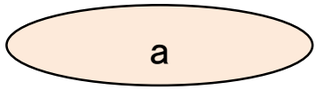
relationship



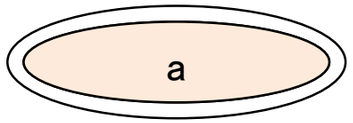
identifying relationship
for a weak entity set



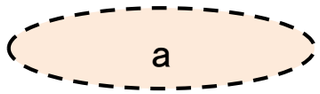
primary key



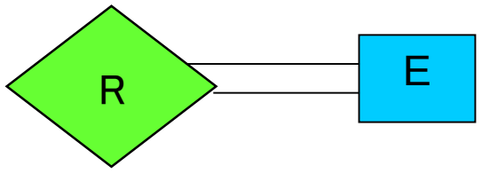
attribute



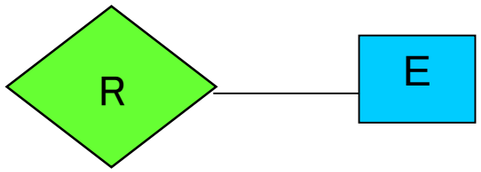
multi-valued attribute



derived attribute

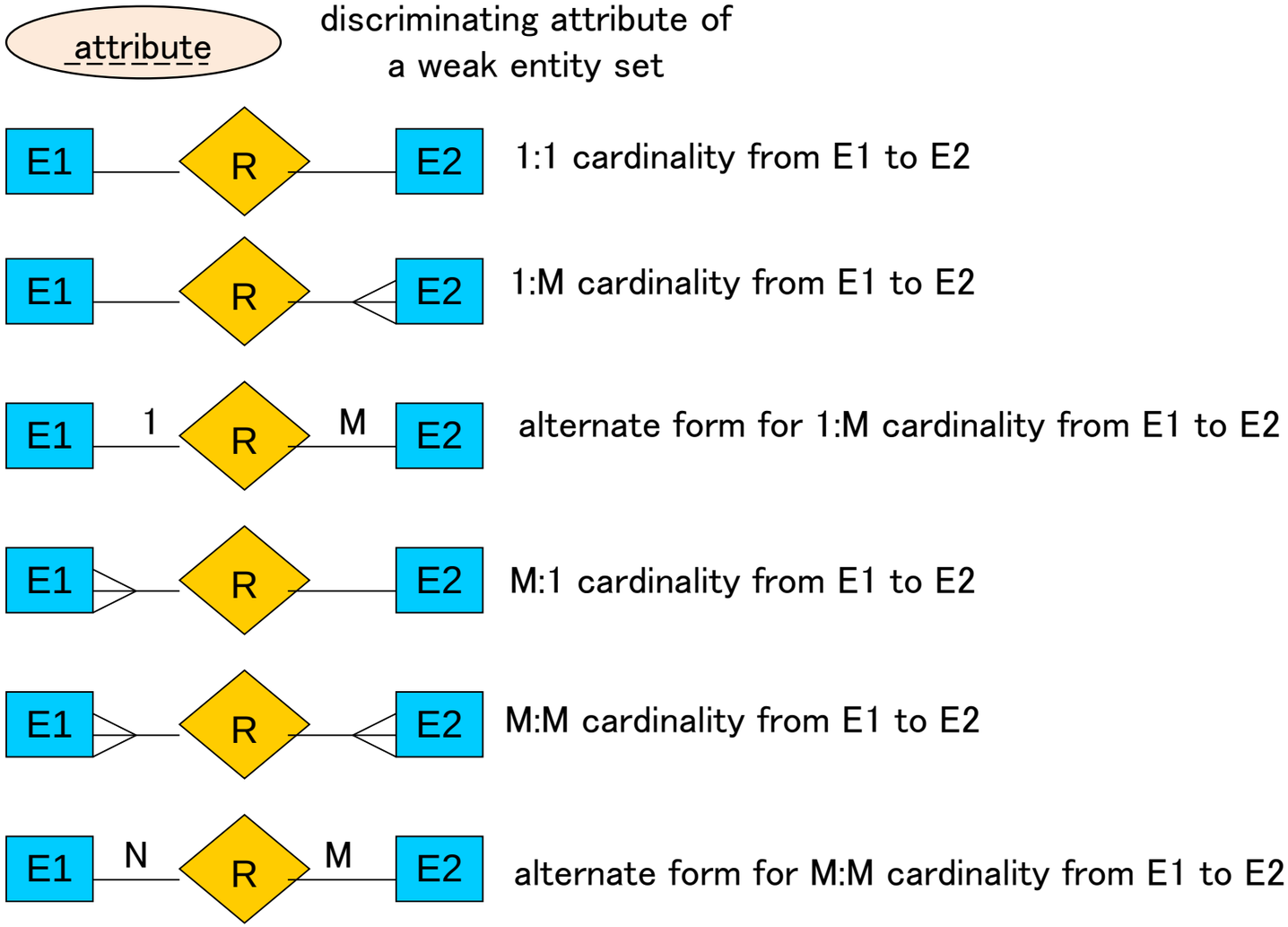


total participation of
entity set in relationship

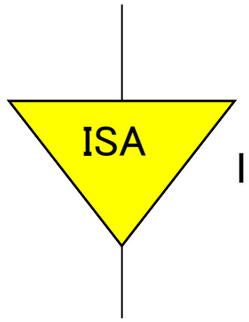


partial participation of
entity set in relationship

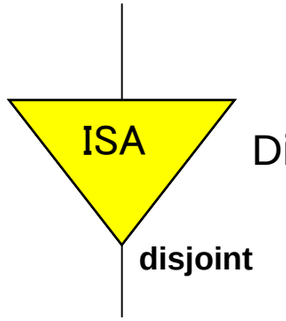
ERD Notations



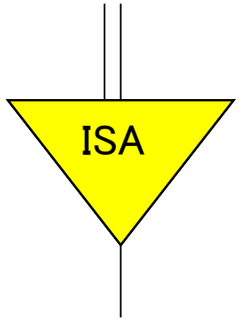
ERD Notations



ISA (specialization or generalization)(partial participation)

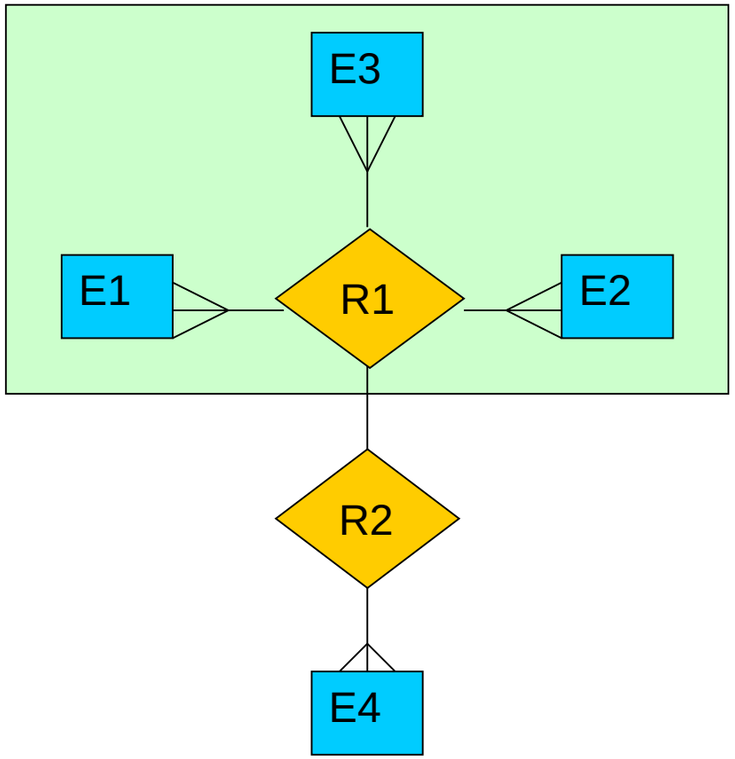


Disjoint ISA (specialization or generalization)

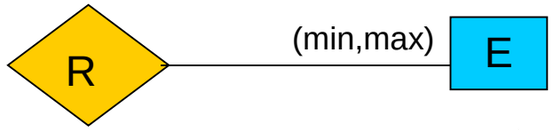


Total generalization

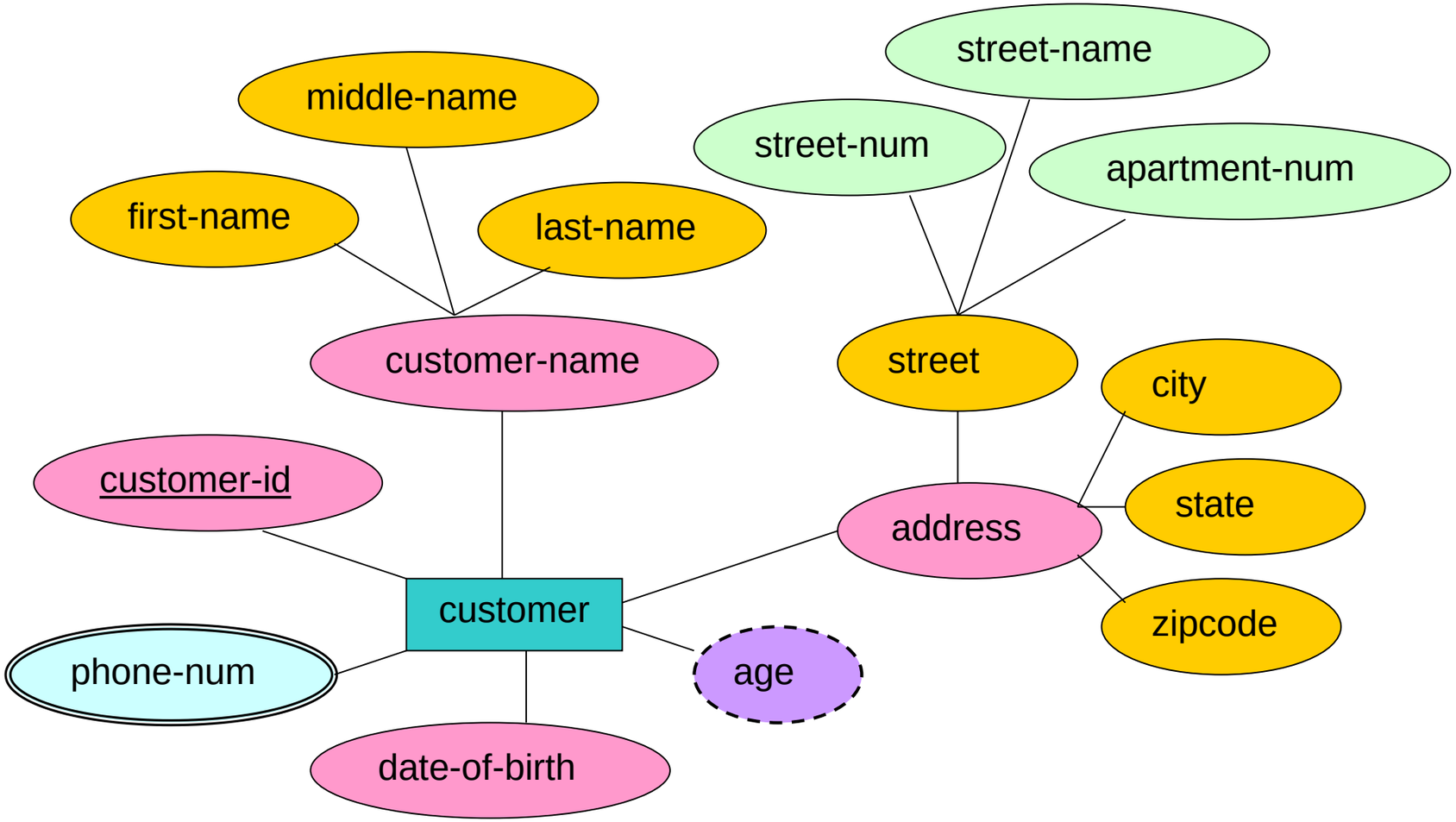
ERD Notations

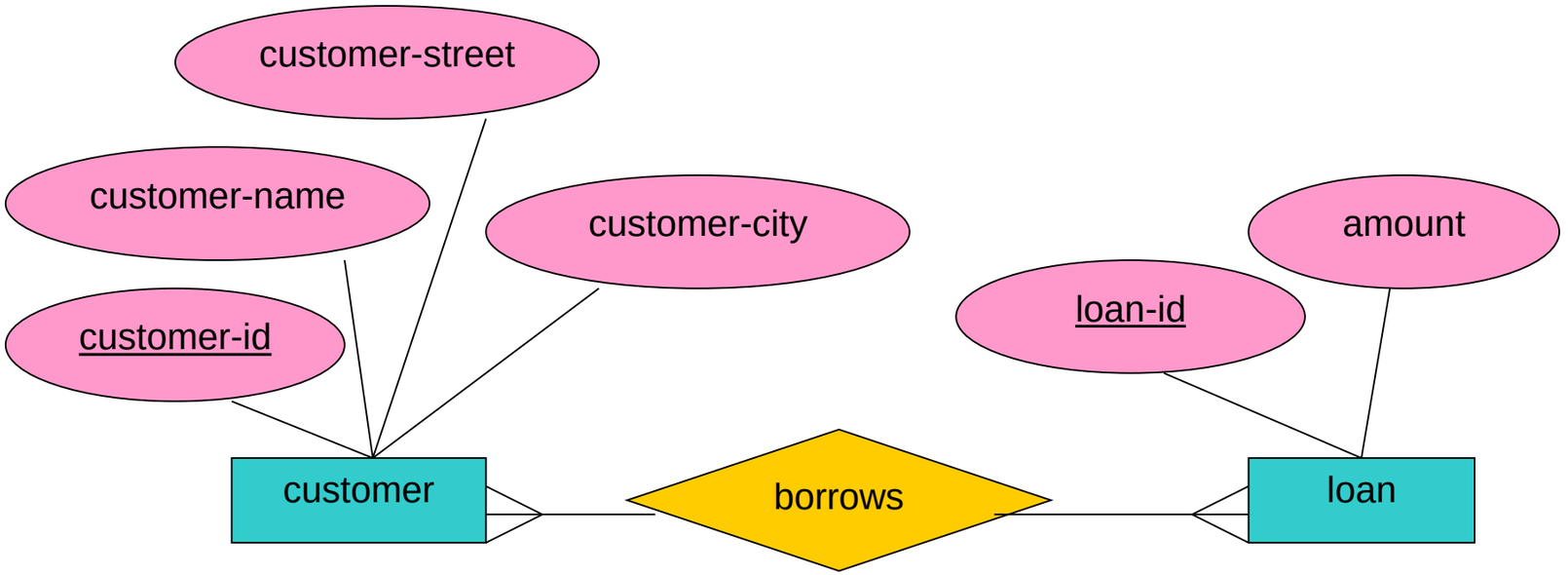


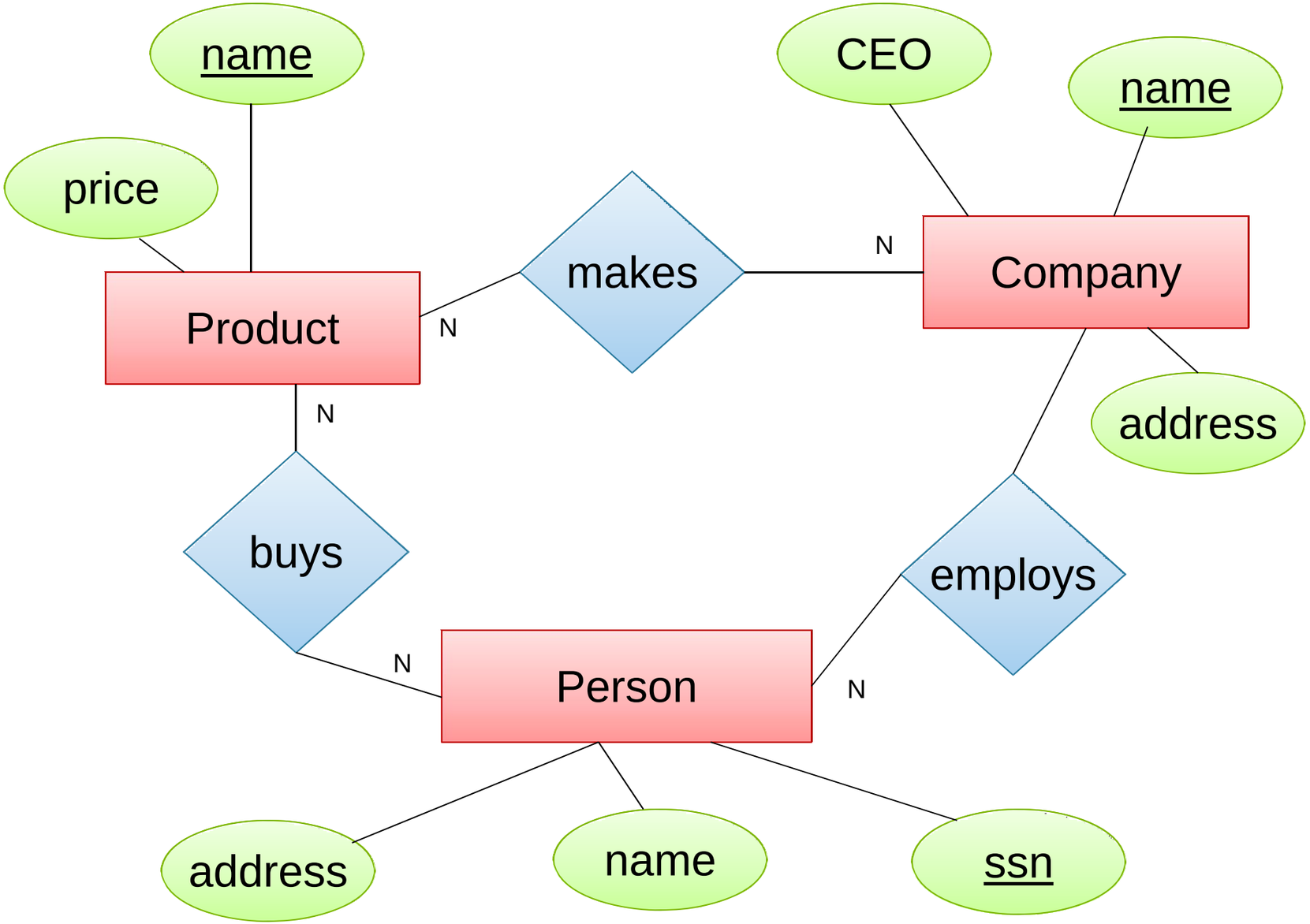
Aggregation: box drawn around relationship which is treated as an entity



Structural constraint: (min,max) on the participation of an entity in a relationship

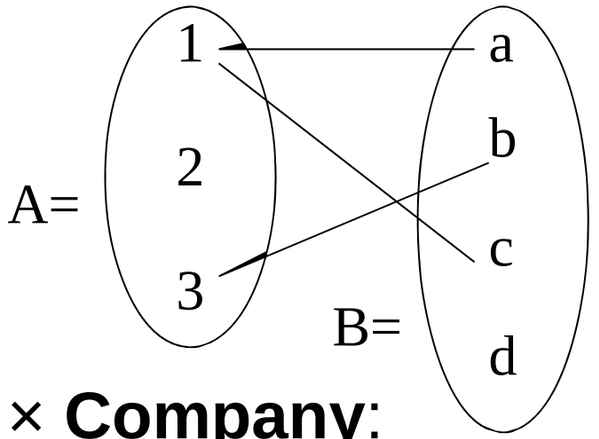

street-name
middle-name
street-num
apartment-num
first-name
last-name
customer-name
street
city
customer-id
state
address
customer
zipcode
phone-num
age
date-of-birth


customer-street
customer-name
customer-city
customer-id
loan-id
amount
customer
borrows
loan

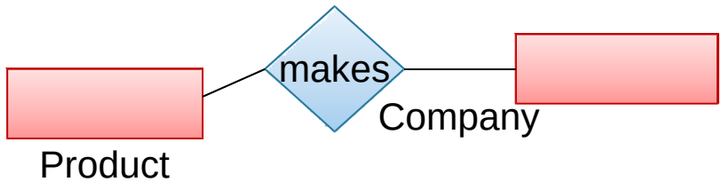

name
price
Product
makes
N
N
CEO
name
Company
address
N
buys
employs
N
N
Person
address
name
ssn

What is a Relation(ship)?

- A *relationship* is an association among several entities.
- A mathematical definition:
  - if A, B are sets, then a relation R is a subset of A × B
- A = {1, 2, 3}, B = {a, b, c, d},
  A × B = {(1,a), (1,b), . . ., (3,d)}
  R = {(1,a), (1,c), (3,b)}



- **makes** is a subset of **Product** × **Company**:



What is a Relationship?

- The association between entity sets is referred to as participation; that is, the entity sets E_1, E_2, …, E_n participate in the relationship *R*.
- A *relationship instance* in an E-R schema represents an association between named entities in the real world enterprise which is being modeled.
- A relationship may also have attributes which are called *descriptive attributes*.

Constrains in a Relationship

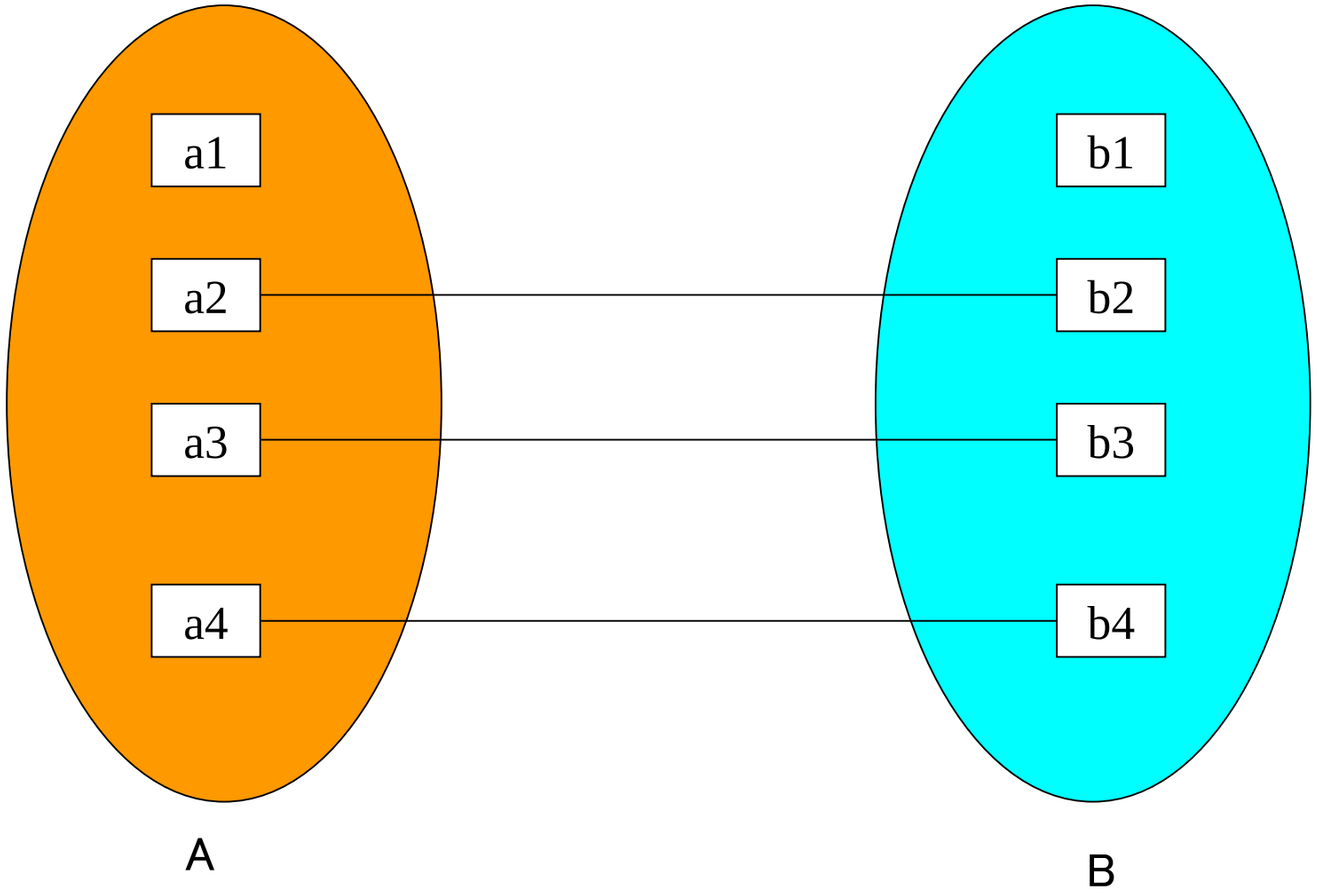
Mapping cardinalities (also called *cardinality ratios*), express the number of entities to which another entity can be associated via a relationship set.

Mapping cardinalities are most useful in describing binary relationships, although they can be helpful in describing relationship sets that involve more than two entity sets. We will focus only on binary relationships for now.

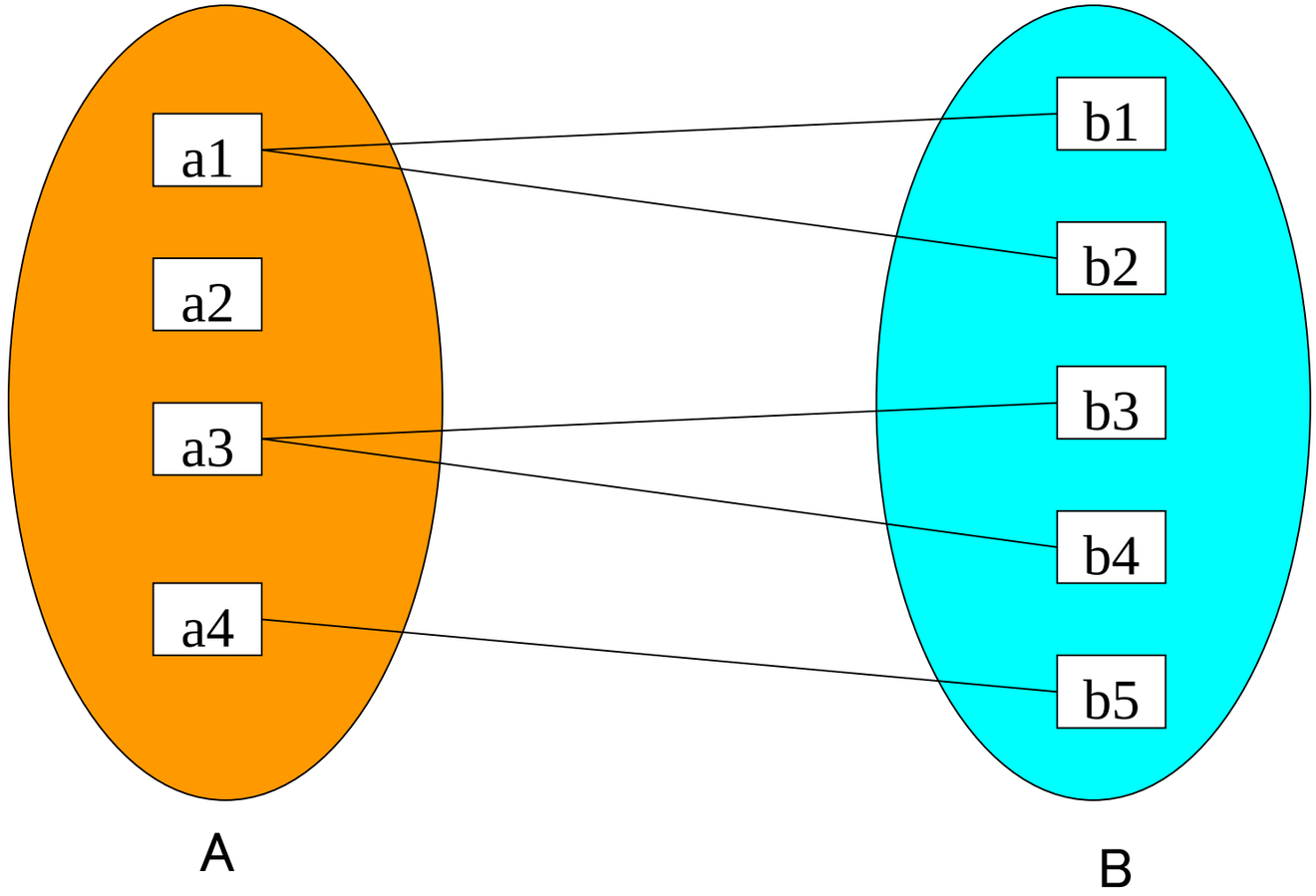
For a binary relationship set *R* between entity sets *A* and *B*, the mapping cardinality must be one of the following:

(1:1) *one to one from A to B*
(1:M) *one to many from A to B*
(M:1) *many to 1 from A to B*
(M:M) *many to many from A to B*

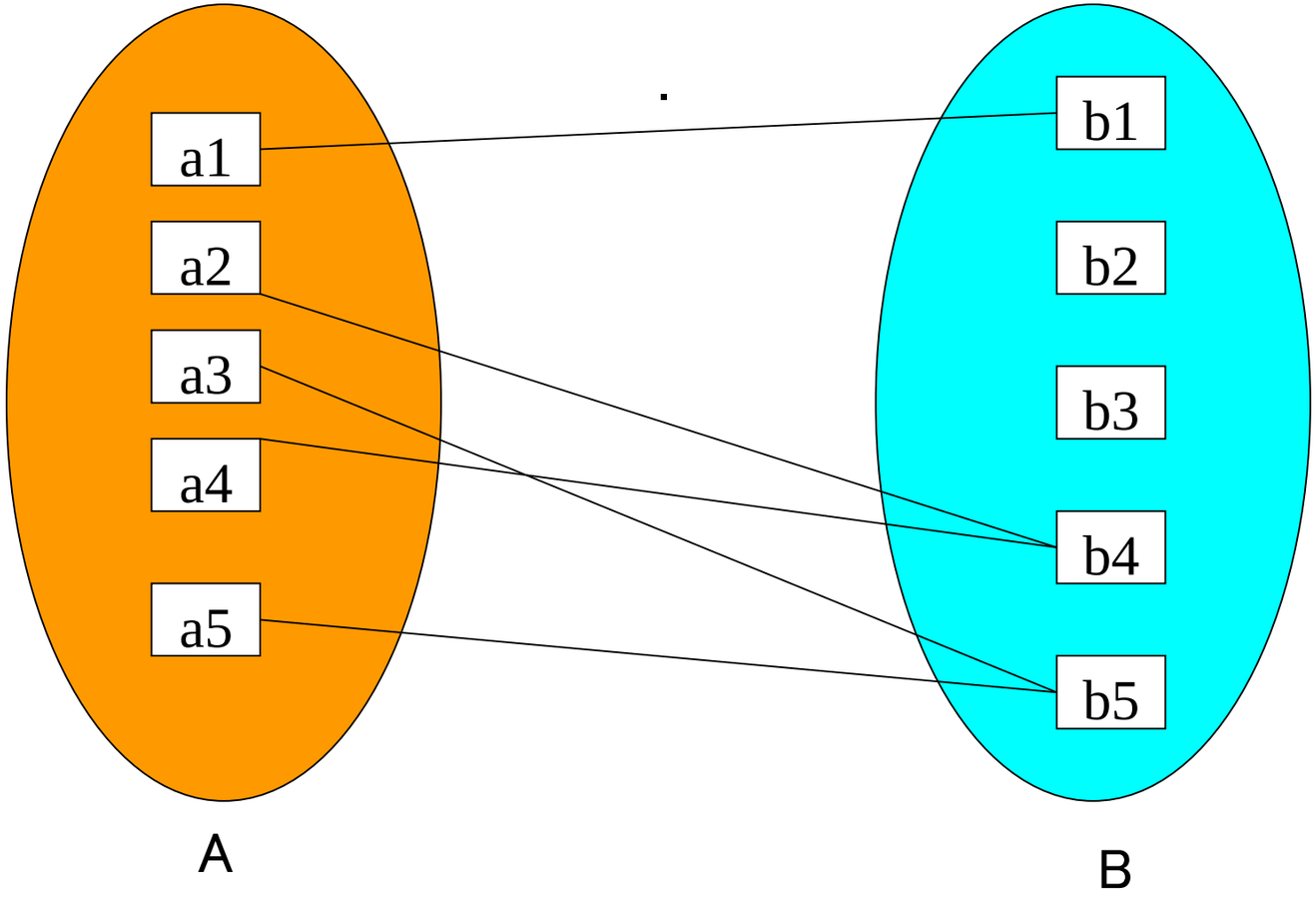
Mapping Cardinality: 1:1 from A to B



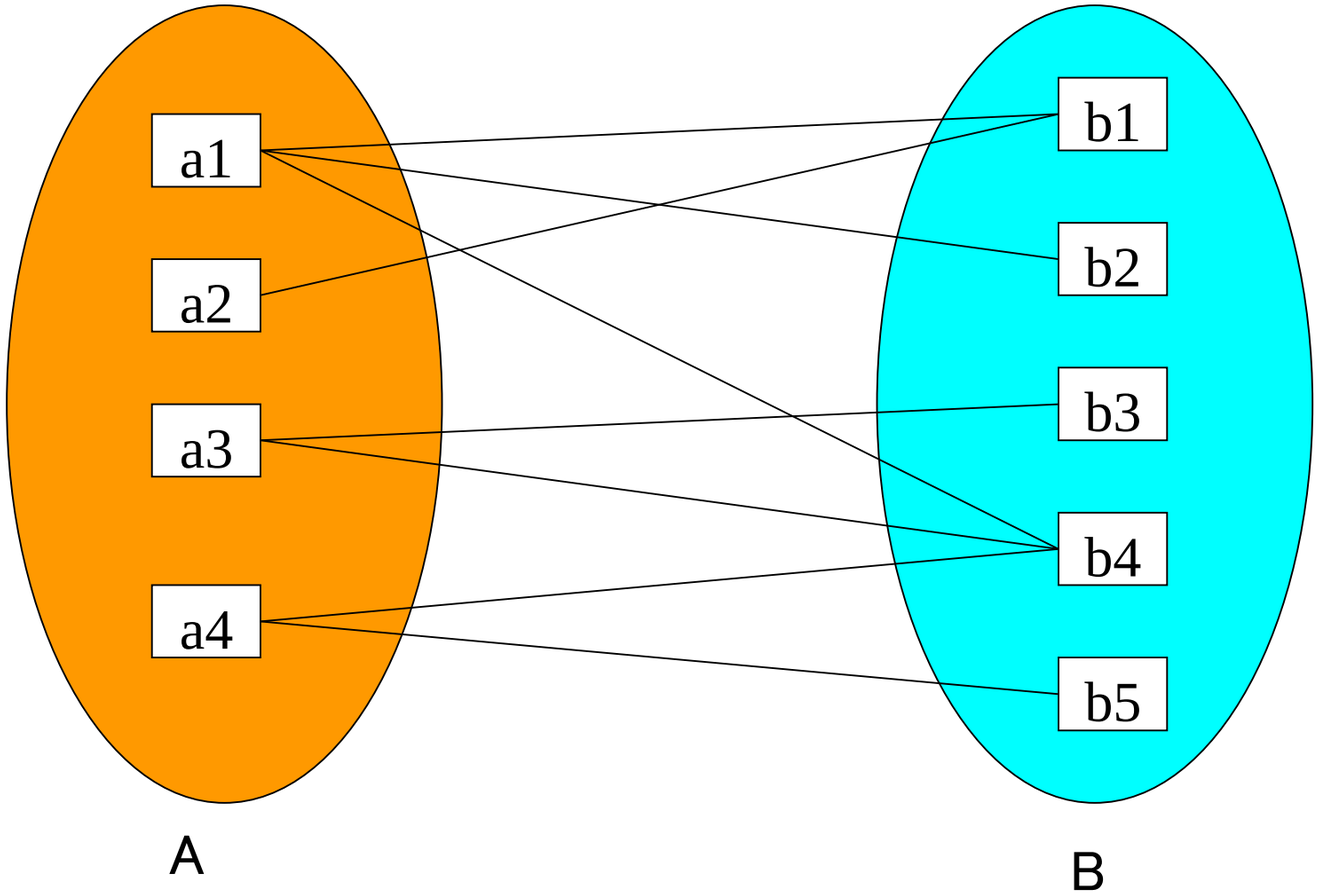
Mapping Cardinality: 1:M from A to B



Mapping Cardinality: M:1 from A to B



Mapping Cardinality: M:M from A to B



Participation constraints

The participation of an entity set *E* in a relationship set *R* is said to be *total* if every entity in *E* participates in at least one relationship in *R.*

If only some of the entities in *E* participate in a relationship in *R*, the participation of entity set *E* is relationship *R* is said to be *partial*.

Attributes in ERD

Simple attribute: A simple attribute contains no subparts

Composite attribute: a composite attribute will contains subparts.

Single-valued attribute: A single-valued attribute may have at most one value at any particular time instance.

Multi-valued attribute: A multiple-valued attribute may have several different values at any particular time instance.

Derived attribute: This is an attribute whose value is derived (computed) from the values of other related attributes or entities.

Null attribute: An attribute takes a null status when an entity does not have a value for it. Null values are usually special cases that can be handled in a number of different ways depending on the situation.

Keys of an Entity set

We need some mechanism to distinguish entities within a given entity set.

Conceptually, individual entities are distinct;

The differences among entities must be expressed in terms of their attributes.

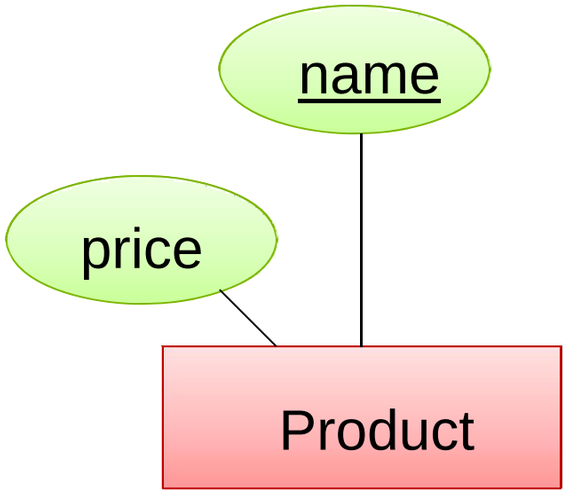
The values of the attribute values of an entity must be such that they can *uniquely identify* the entity.

A ***key*** allows us to identify a set of attributes that suffice to distinguish entities from each other.

Keys also help uniquely identify relationships, and thus distinguish relationships from one another.

Keys in E/R Diagrams

- Every entity set **must** have a key

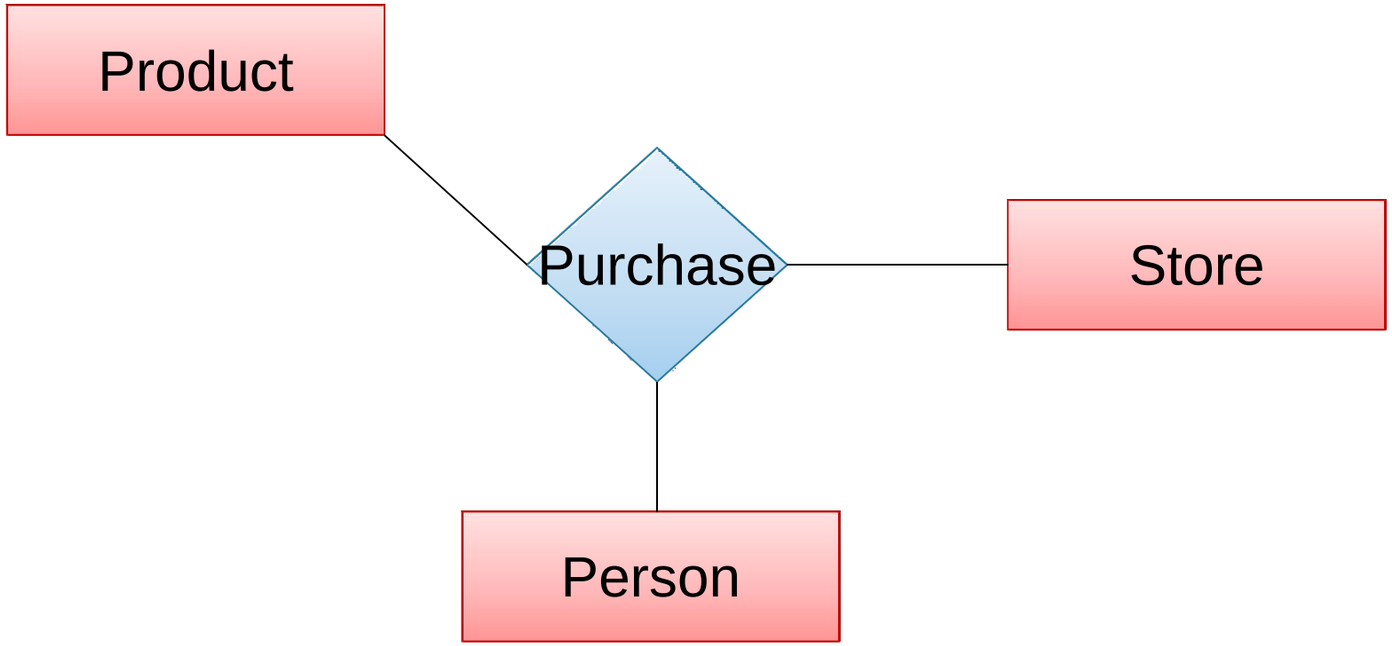


Different types of Keys in ERD

- A *superkey* is a set of one or more attributes that, taken collectively, allow us to identify uniquely an entity in the entity set.
- If the set *K* is a superkey of entity set *E*, then so too is any superset of *K*. We are interested only in superkeys for which no proper subset of *K* is a superkey. Such a minimal superkey is called a *candidate key*.
- Either there is only a single such set of attributes or there are several distinct sets from which only one is selected by the database designer and this set of attributes defines the *primary key* which is typically referred to simply as the *key* of the entity set.
- A key (primary, candidate, and super) is a property of the entity set, rather than of the individual entities. Any two individual entities in the set are prohibited from having the same value on all attributes which comprise the key attributes at the same time. This constraint on the allowed values of an entity within the set is a *key constraint*.

Multi-way Relationships

How do we model a purchase relationship between buyers, products and stores?

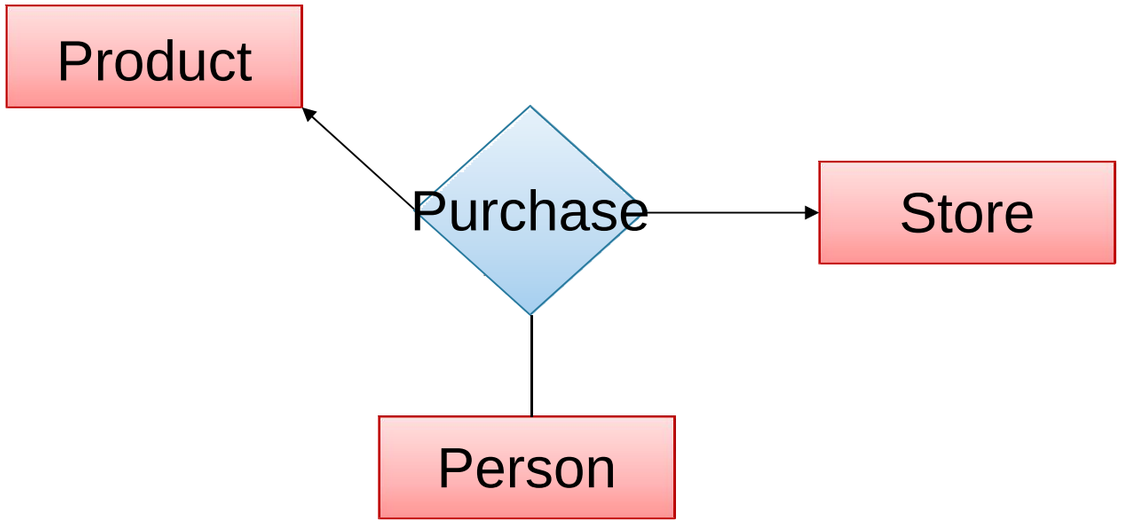


Can still model as a mathematical set (Q. how ?)

A. As a set of triples$\subseteq$ Person $\times$ Product $\times$ Store

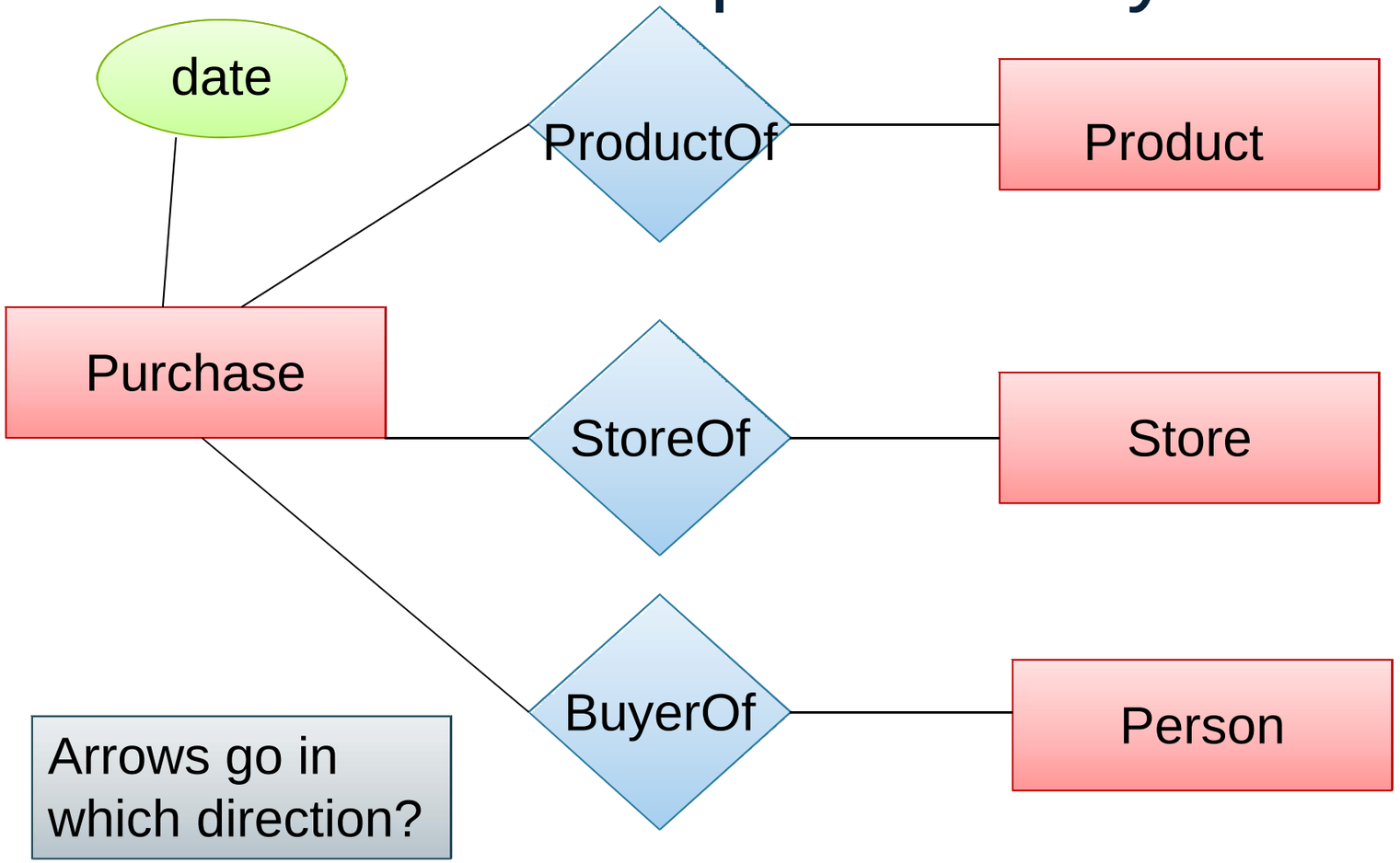
Arrows in Multiway Relationships

Q: What does the arrow mean ?

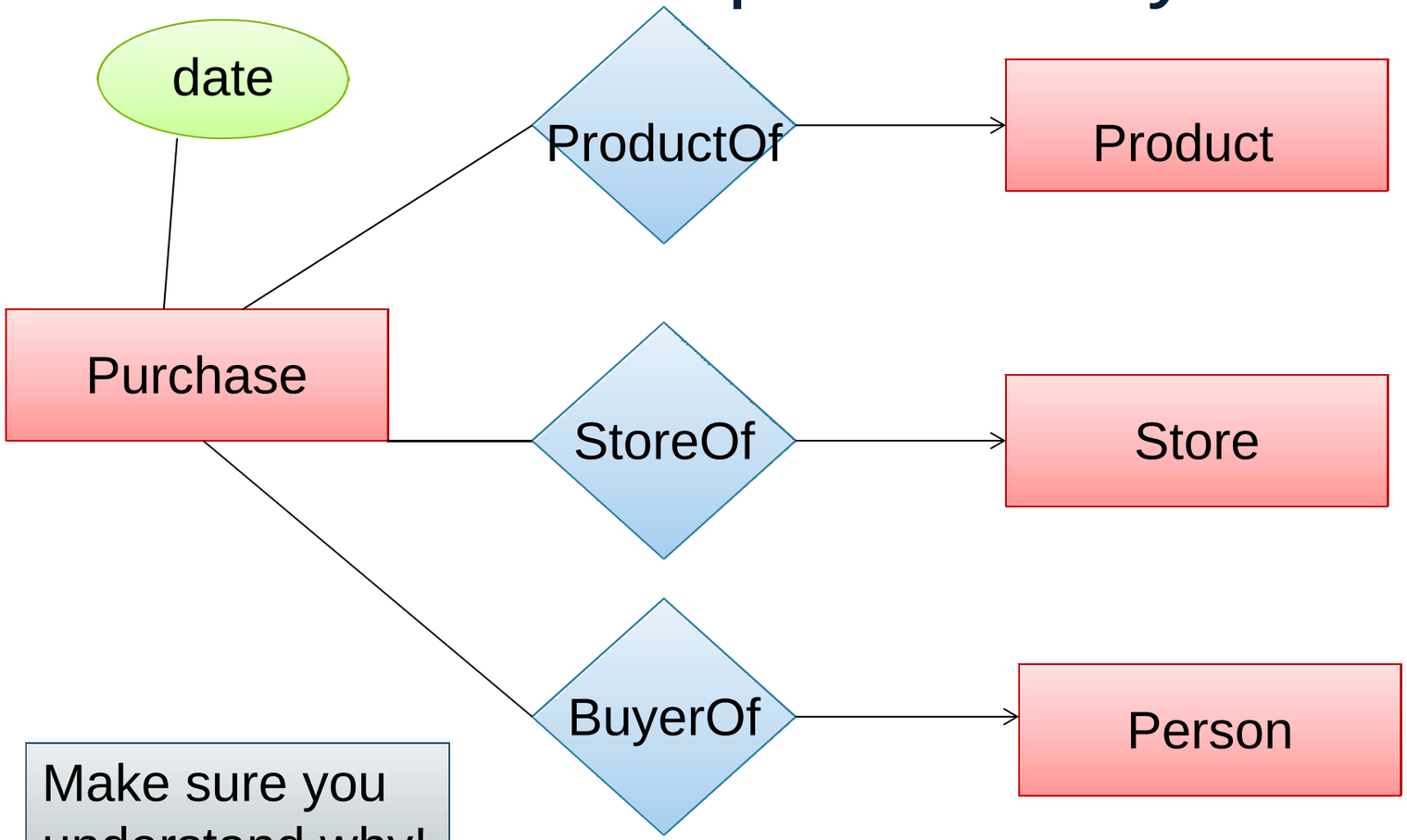


A: A given person buys a given product from at most one store AND every store sells to every person at most one product

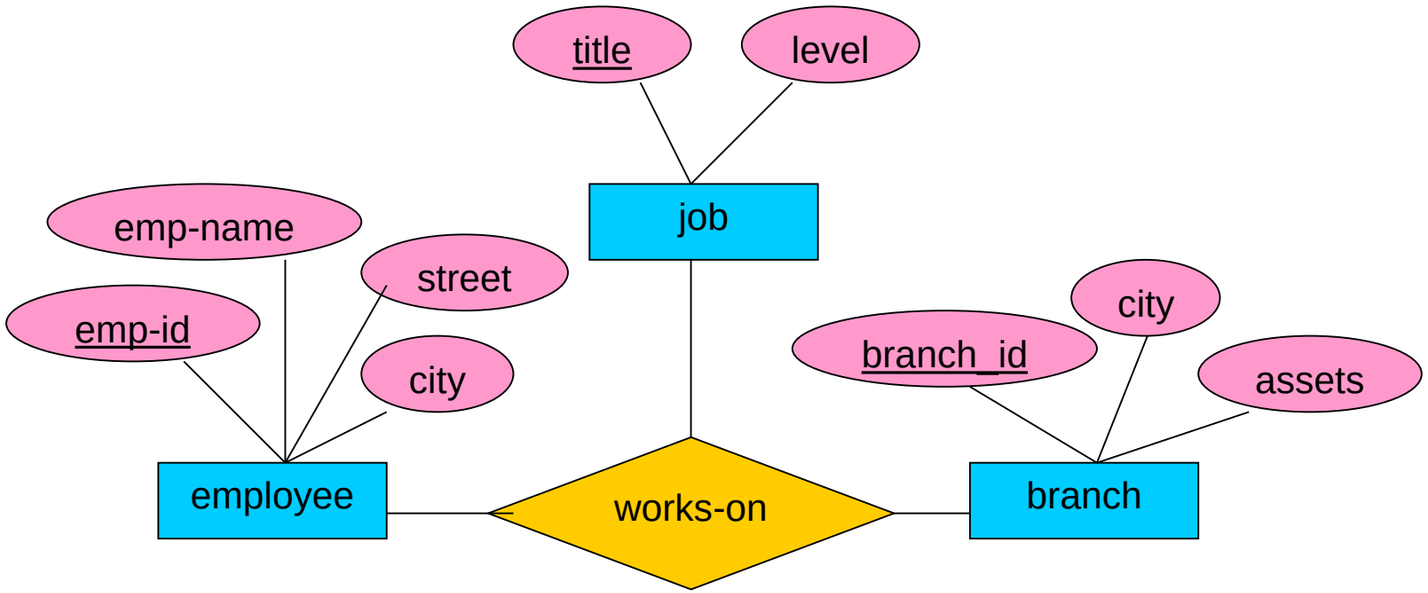
Converting Multi-way Relationships to Binary

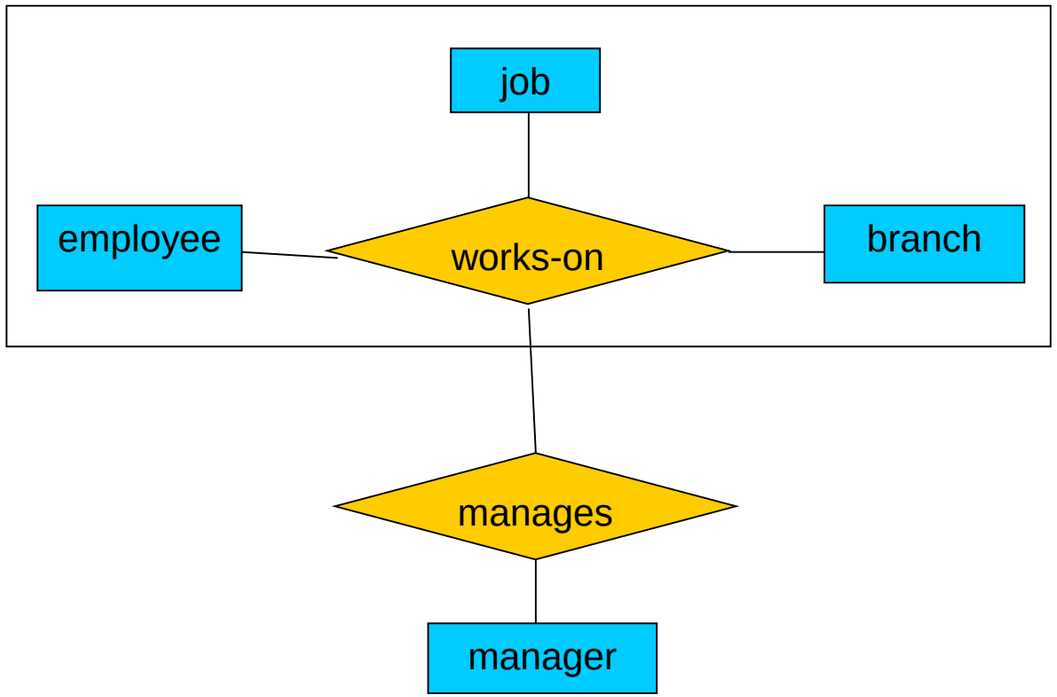


Converting Multi-way Relationships to Binary



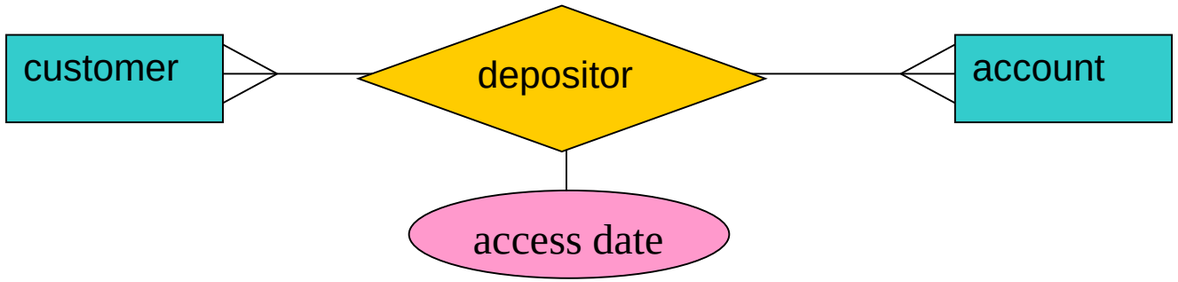
Aggregation





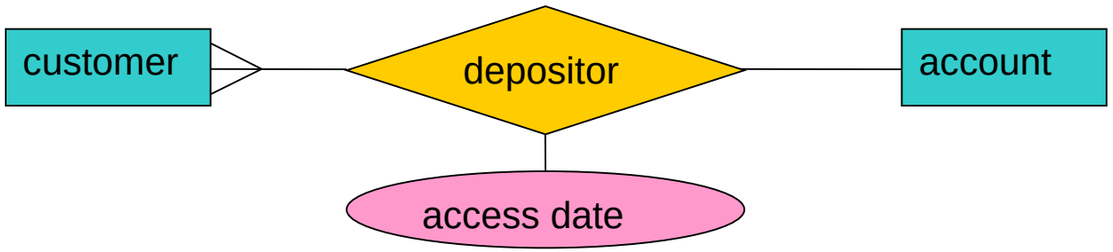
Effect of Cardinality Constraints on Keys

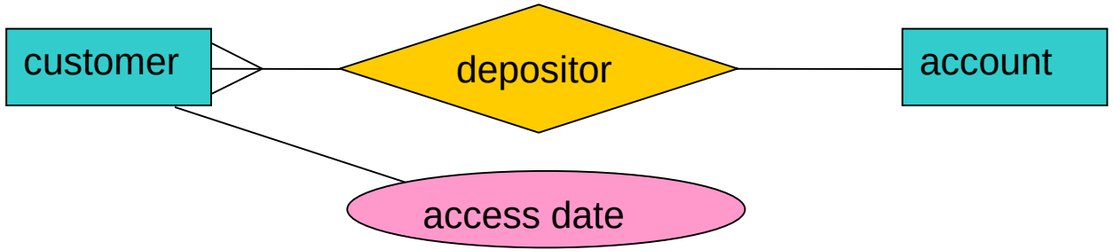
This E-R diagram represents a many-to-many cardinality for the relationship *depositor* with an attribute of *access date* associated with the relationship set with two entities *customer* and *account* participating in the relationship. The primary key of the relationship *depositor* will consist of the union of the primary keys of *customer* and *account*



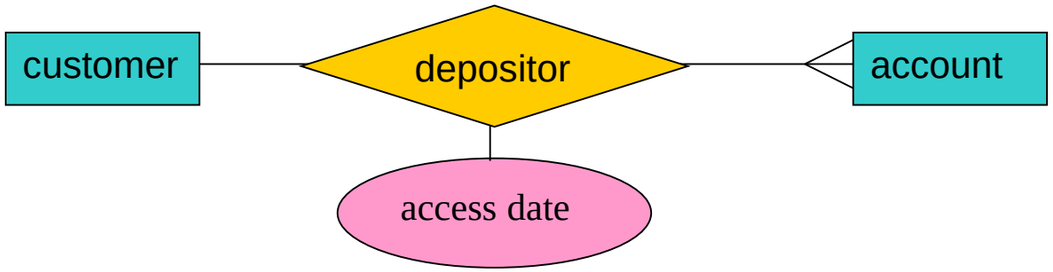
Effect of Cardinality Constraints on Keys

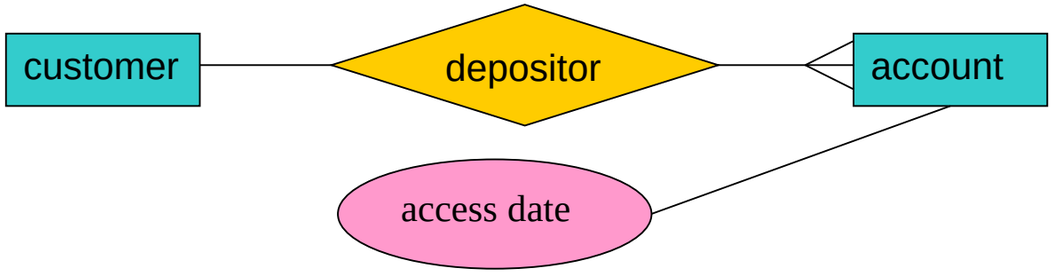
the *depositor* relationship is many-to-one from *customer* to *account*
In this case the primary key of the *depositor* relationship is simply the primary key of the *customer* entity set.





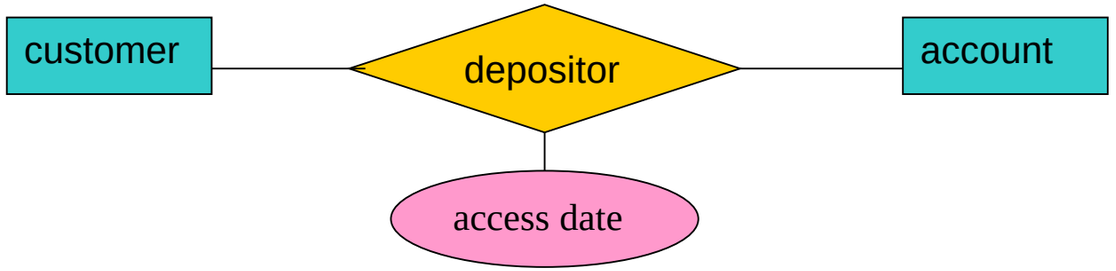
Same for the one-to-many relationship

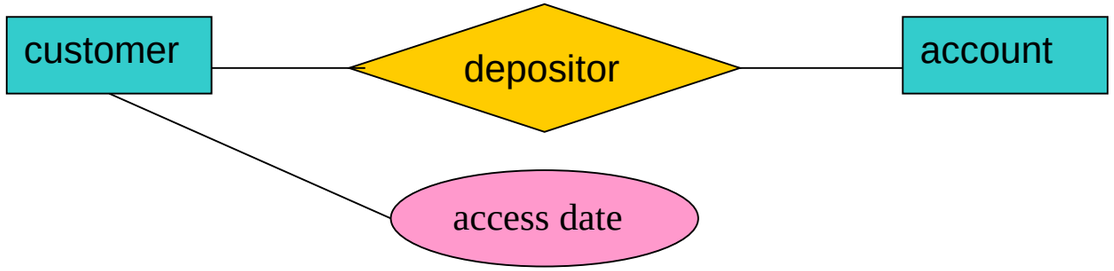


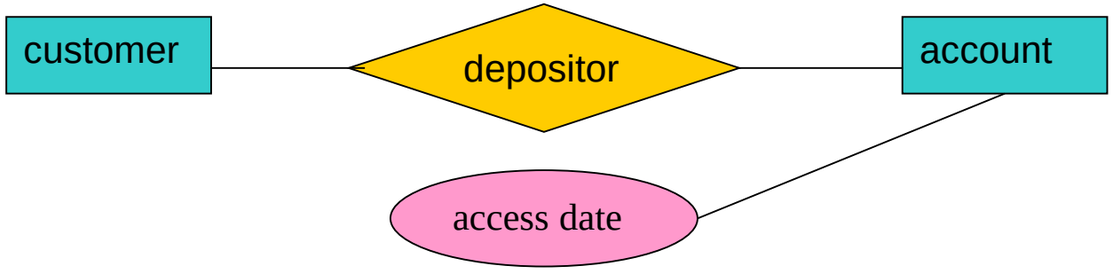


Effect of Cardinality Constraints on Keys

In this one-to-one relationship from *customer* to *account*, the attribute *access-date* could be associated with either the *customer* set or the *account* set without loss of information







Further design issues

Designer should define the given information is:

- Entity sets or Attributes?
- Entity sets or Relationship sets?
- Strong entity sets or Weak entity sets?

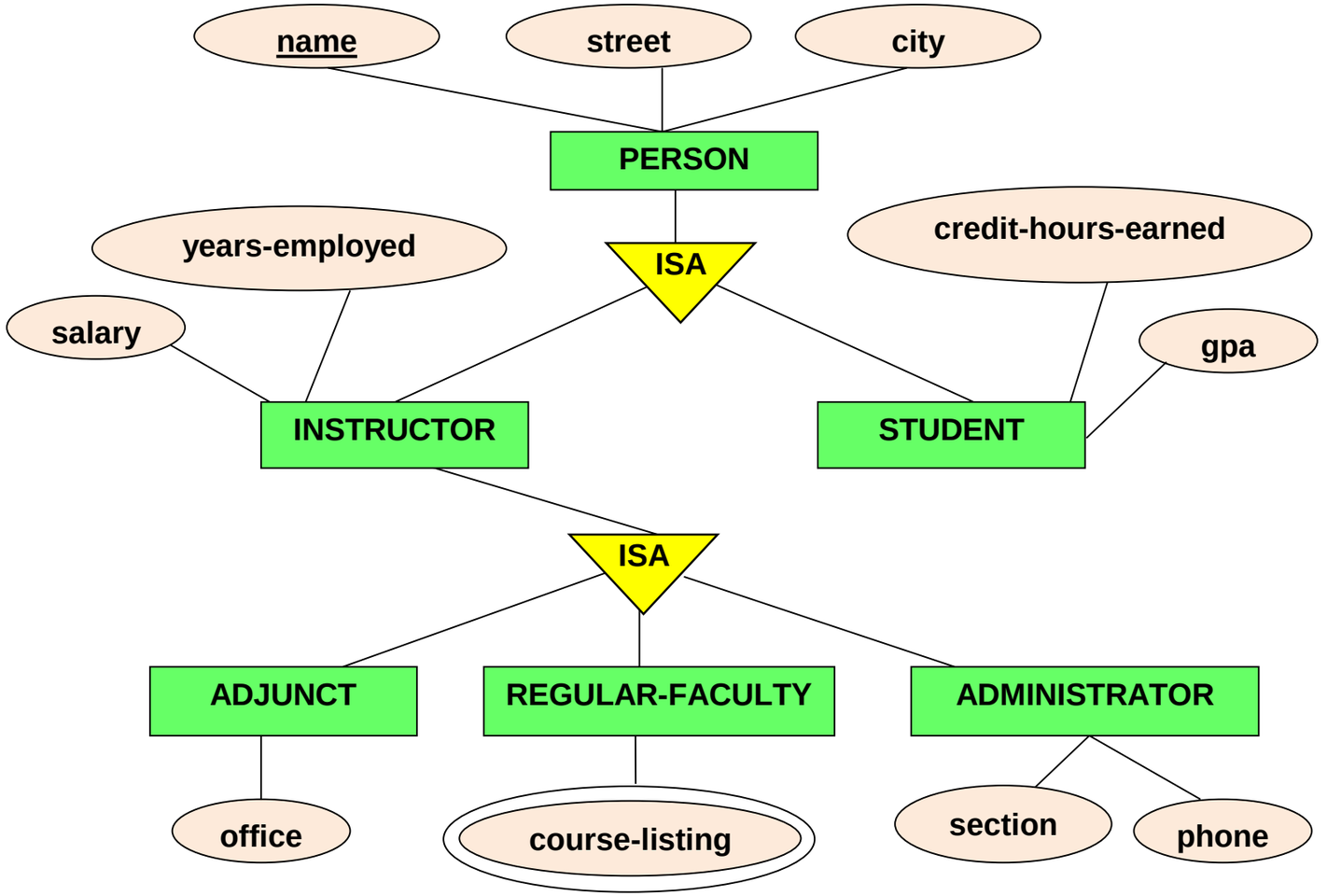
Specialization

- An entity set may include sub-groupings of entities that are distinct in some way from other entities in the set.
- The process of designating sub-groupings within an entity set is called *specialization*.
- Specialization can be repeatedly applied so that there may be specializations within specializations.
- In terms of an E-R diagram, specialization is depicted by a triangle shaped component which is labeled ISA, which is a shorthand form of the “is-a” superclass-subclass relationship.

Generalization

- In this process, multiple entity sets are synthesized into a higher-level entity on the basis of common attributes.
- This commonality of attributes is expressed by *generalization,* which is a containment relationship that exists between a higher-level entity set and one or more lower level entity sets.
- The higher-level entity set represents the *superclass* and the lower-level entity represents the *subclass.*
- For all practical purposes, generalization is just the inverse of specialization and both processes can be applied (almost interchangeably) in designing the schema for some real-world scenario.

Specialization and Generalization



Specialization and Generalization

- Differences in the two approaches are normally characterized by their starting points and overall goal.
- Specialization arises from a single entity set; it emphasizes differences among the entities within the set by creating distinct lower-level entity sets. These lower-level entity sets may have attributes or participate in relationships, that do not apply to all the entities in the higher-level entity set.
  - The reason that a designer may need to use specialization is to represent such distinctive features of the real world scenario.
- Generalization arises from the recognition that a number of entity sets share some common characteristics.
  - On the basis of these commonalities, generalization synthesizes these entity sets into a single, higher-level entity set.
  - Generalization is used to emphasize the similarities among lower-level entity sets and to hide the differences. It also permits an economy of representation in that the shared attributes are not replicated.

Attribute Inheritance

- A crucial property of the higher and lower level entities that are created by specialization and generalization is *attribute inheritance*.
- The attributes of the higher-level entity sets are said to be *inherited* by the lower-level entity sets.
- A lower-level entity (subclass) inherits all attributes and relationships (including the participation in those relationship) which belong to the higher-level entity set (superclass) which defines it.
- Higher-level entity sets do not inherit any attribute or relationship which is defined within the lower-level entity set.
- ERD is typically developed will be a *hierarchy* of entity sets, in which the highest-level entity appears at the top of the hierarchy.
  - If, in such a hierarchy, a given entity set may be involved as a lower-level entity set in only one ISA relationship, then the inheritance is said to be *single-inheritance*.
  - If, on the other hand, a given entity set is involved as a lower-level entity set in more than one ISA relationship, then the inheritance is said to be *multiple-inheritance* (then the resulting structure is called a *lattice*).

Constraints on Generalization

- *Predicate-defined:* In predicate-defined lower-level entity sets, membership is evaluated on the basis of whether or not an entity satisfies an explicit predicate (a condition).
- *User-defined:* User-defined lower-level entity sets are not constrained by a membership condition; rather, the database user assigns entities to a given entity set.
- *Disjoint*: A disjointness constraint requires that an entity belong to no more than one lower-level entity set.
- *Overlapping:* In *overlapping generalizations*, the same entity may belong to more than one lower-level entity set within a single generalization.
- *Total generalization/specialization:* Each higher-level entity must belong to a lower-level entity.
- *Partial generalization/specialization:* Some higher-level entities may not belong to any lower-level entity set.

ERD with Role indicator

- Roles in an E-R diagram are indicated by labeling the lines that connect entity sets to relationship sets.
- Roles can be identified for unary (recursive), binary, and nonbinary relationships.