Translation to Relational Schema

CIS 205

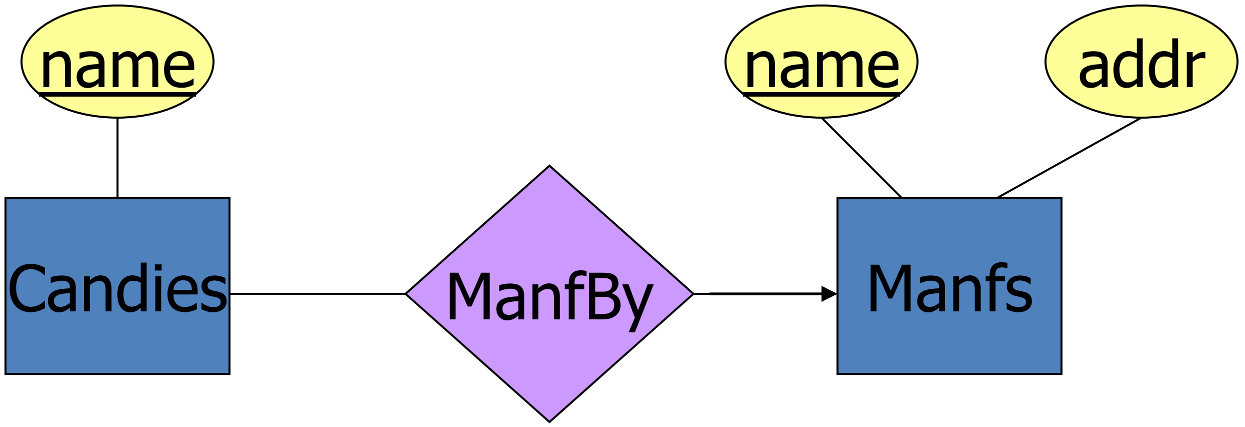
Agenda

- Translation of ERD into Relational Schema
- Stable Translation (We will concentrate on the stable translations)
- Mapped Translation (Will show example)
- Foreign Keys
- Referential Integrity
- Translation Exercises

Relationships Review

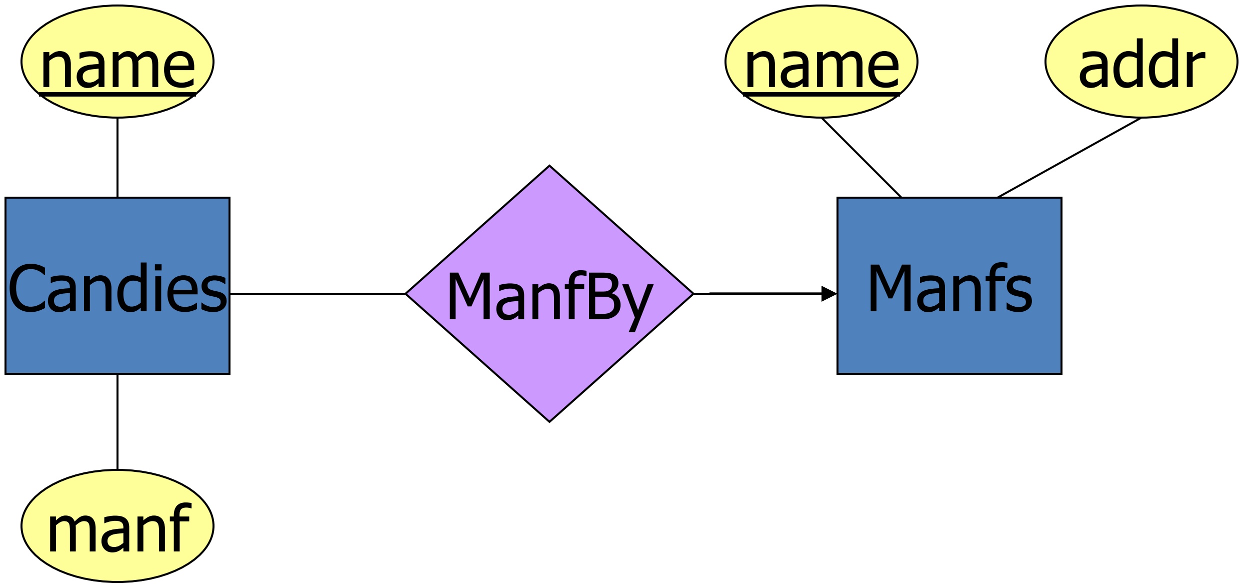
- A relationship connects two or more entity sets.
- It is represented by a diamond, with lines to each of the entity sets involved.

Example: Good



This design gives the address of each manufacturer exactly once.

Example: Bad



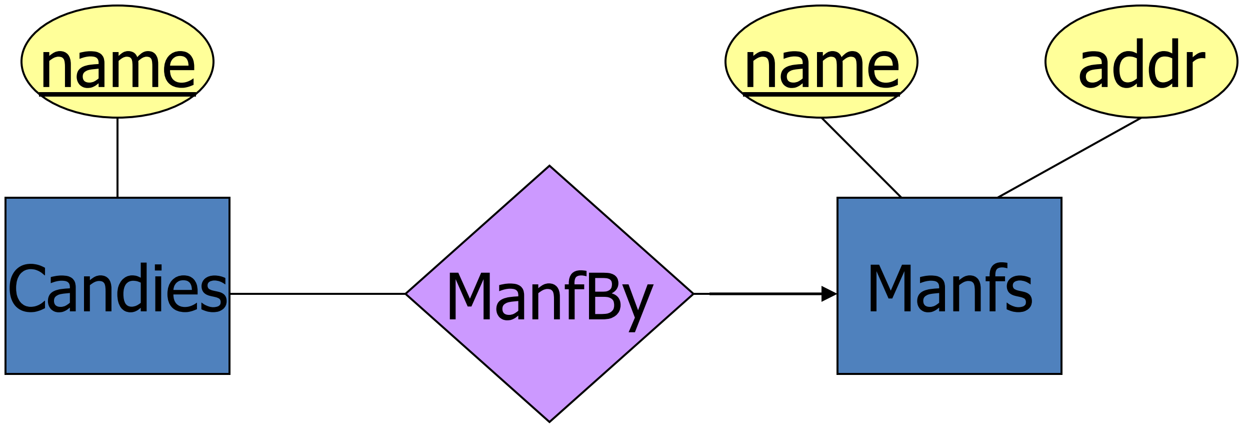
This design states the manufacturer of a candy twice: as an attribute and as a related entity.

Entity Sets Versus Attributes

- An entity set should satisfy at least one of the following conditions:
  - It is more than the name of something; it has at least one nonkey attribute.

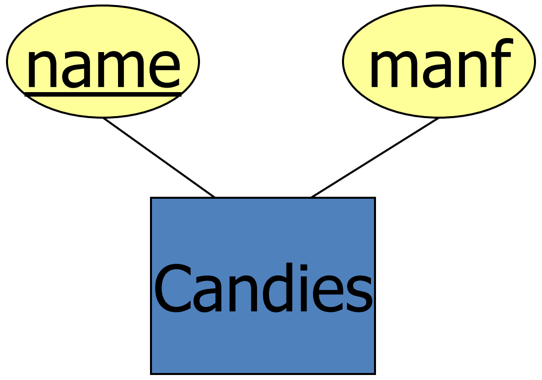
    or
  - It is the “many” in a many-one or many-many relationship.

Example: Good



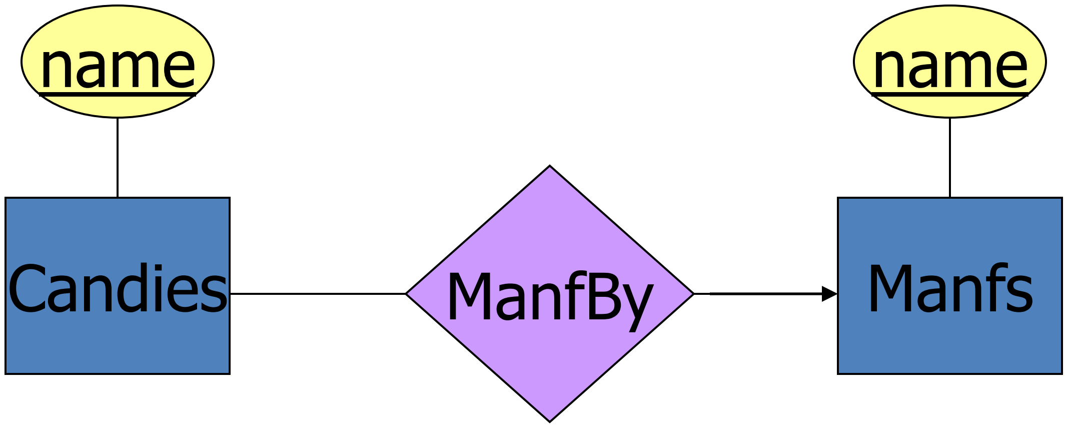
- Manfs deserves to be an entity set because of the nonkey attribute addr.
- Candies deserves to be an entity set because it is the “many” of the many-one relationship ManfBy.

Example: Good



There is no need to make the manufacturer an entity set, because we record nothing about manufacturers besides their name.

Example: Bad



Since the manufacturer is nothing but a name, and is not at the "many" end of any relationship, it should not be an entity set.

Translation of ERD into Relational Schema

- Considerations for Translating
  - Minimize the number of relations to reduce query-processing time
  - Do not allow null values if possible
  - Provide a design that accommodates potential changes of the schema

Translation of ERD into Relational Schema

- Reality
  - In general, decreasing the number of relations means…
    - Increased efficiency of query processing
  - But also means…
    - More null values
    - Less semantic clarity
    - Less flexibility

Translation of ERD into Relational Schema

- Two Translation Techniques that we will examine
  - Stable Translation
  - Mapped Translation
- General Rules – Regardless of technique
  - A weak (or identifying) relationship is always combined with the weak entity (will explain this later)
  - Each two entity relationship becomes a separate table
  - Each ternary relationship becomes a separate table (will explain this later)

- In a relational database, a **Weak Entity** is an entity that cannot be uniquely identified by its attributes alone; therefore, it must use a foreign key in conjunction with its attributes to create a primary key. The foreign key is typically a primary key of an entity it is related to.

Stable Translation

- Every entity becomes a table
  - All the attributes of the entity become the attributes of the table
- Every relationship becomes a table
  - Add Primary Key to the relationship table
  - Add any non-key attribute of the relationship to the relationship table
  - A weak relationship is always combined with the weak entity

Stable Translation

- Stable translation is called “stable” because a change in the cardinality constraints does not change the table structure
- Advantage
  - Provides stable schema against constraint changes
- Disadvantage
  - Generates a larger number of tables than other methods (this will be acceptable while we learn Database Management}

EAR Translation

- Every entity becomes a table
  - All the attributes of the entity become the attributes of the table
- 1:N relationships are mapped into the N-side entity type. For each N-side entity type, add to the N-side entity type
  - The PK of the 1-side entity type (this is called the foreign key attribute)
  - Any non-key attributes of the relationship
  - (We will go over an example of this in class)

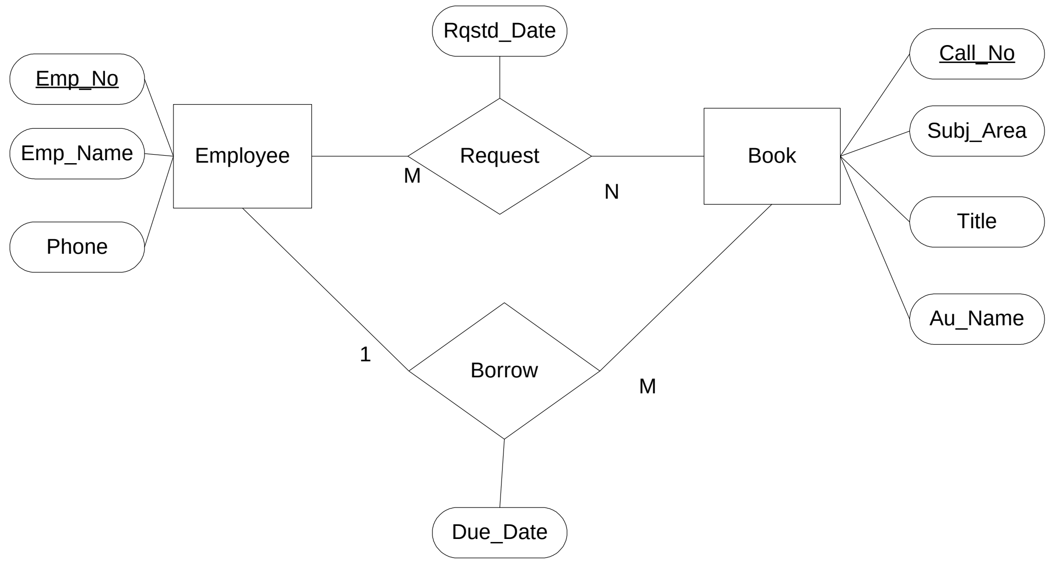
EAR Translation

- Create a separate table for each M:N relationship and ternary relationship
  - Add Primary Key to the relationship table
  - Add any non-key attribute of the relationship to the relationship table
- 1:1 relationship can be combined with either side of the entity (recommendations for this are provided later in this presentation)

Mapped Translation

- Advantage
  - Minimize the number of tables
- Disadvantages
  - The schema can be changed if cardinality changes
  - Foreign keys may have null values (not a very good practice in Database Design)
- The bottom line is that you sacrifice stability for increase speed and efficiency

Translation Example
Will Explain During Lecture



STABLE TRANSLATION METHOD

Employee(<u>Emp_No</u>, Emp_Name, Phone)

Request(<u>Emp_No</u>, <u>Call_No</u>, Rqstd_Date)

Borrow(<u>Call_No</u>, Emp_No, Due_Date)

Book(<u>Call_No</u>, Subj_Area, Title, Au_Name)

MAPPED TRANSLATION METHOD

Employee(<u>Emp_No</u>, Emp_Name, Phone)

Request(<u>Emp_No</u>, <u>Call_No</u>, Rqstd_Date)

Book(<u>Call_No</u>, Subj_Area, Title, Au_Name,
Emp_No, Due_Date)

Foreign Keys

- Foreign Key (FK)
  - An attribute which is a PK in another relation
  - Properties of a FK
    - Represents 1:N or 1:1 relationship in the relational model
    - Can have a null value (not a good thing)
    - Example
      - BOOK(<u>Call_No</u>, Title, Author, Emp_No, Due_Date)
      - The FK Emp_No is null for the books not borrowed

Foreign Keys and Referential Integrity

One should always try to maintain Referential Integrity when possible

- The value of a FK, if not null, must exist in its original table which the FK references, it is the PK of the original table
- Referential Integrity constraint
  - PK is at the end of the line with the arrow head
  - FK is at the other end of the line
- Referential Integrity enforces the consistency between two tables
  - Whenever a value of a PK is changed, the associated FK values must be correspondingly updated(it is good not to change PK)

Foreign Keys and Referential Integrity

- Remember, to truly represent a relationship you need to make sure that the appropriate foreign keys are in place in the schema to allow joins between tables
- All relationships turn into foreign keys within the relationship tables in the stable translation. They may also turn into “not null” constraints (will explain later).

Foreign Keys and Referential Integrity

- To represent a **1:M relationship**, take the primary key of the table on the “1” side and insert it as a foreign key into the table on the “M” side. This is the most basic use of a foreign key. It may make sense to rename the foreign key to reflect its relationship to the table you are inserting it into.

Foreign Keys and Referential Integrity

- To represent a **1:1 relationship** you have a choice. Ask yourself whether it makes more sense to leave this as two separate tables, or to join them together to make one big table. This will depend on whether records will usually exist in both tables, how data will be accesses, if joins will be made constantly or only occasionally when data is queried. If the relationship is mandatory on both sides, there is probably little point in representing it as two tables (discuss in class)

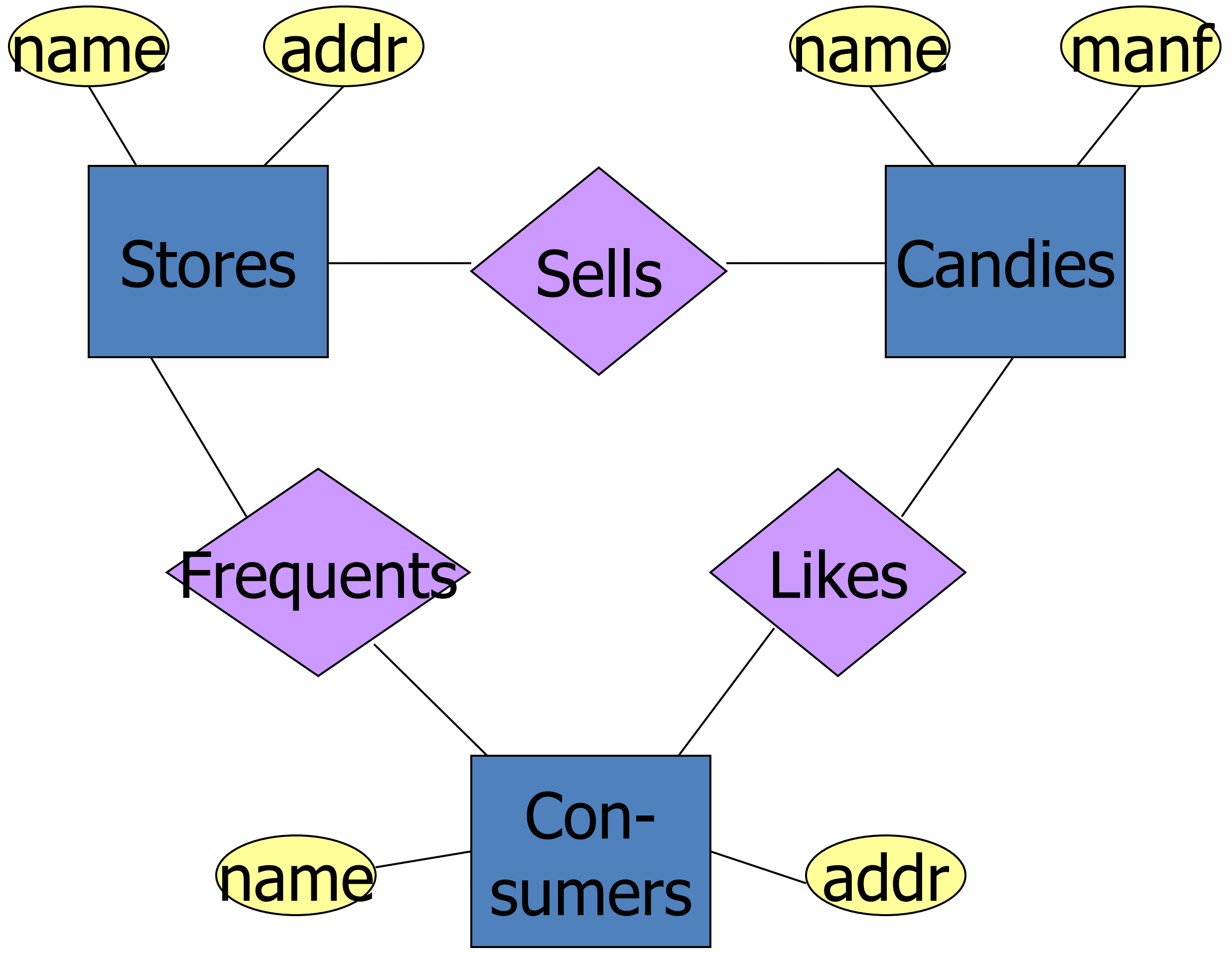
Foreign Keys and Referential Integrity

- Assuming you chose to retain two tables for a **1:1 relationship**, you must chose which table will receive the primary key of the other as a foreign key.
- Put the foreign key in the table that will be accessed less frequently – this minimizes the number of joins required when querying

Foreign Keys and Referential Integrity

- avoid **n-ary relationships** in the first place (a single relationship including three or more entities). They can usually be better represented by using an additional entity and a set of binary relationships (will provide illustration in class)

Example



Stores sell some candies.

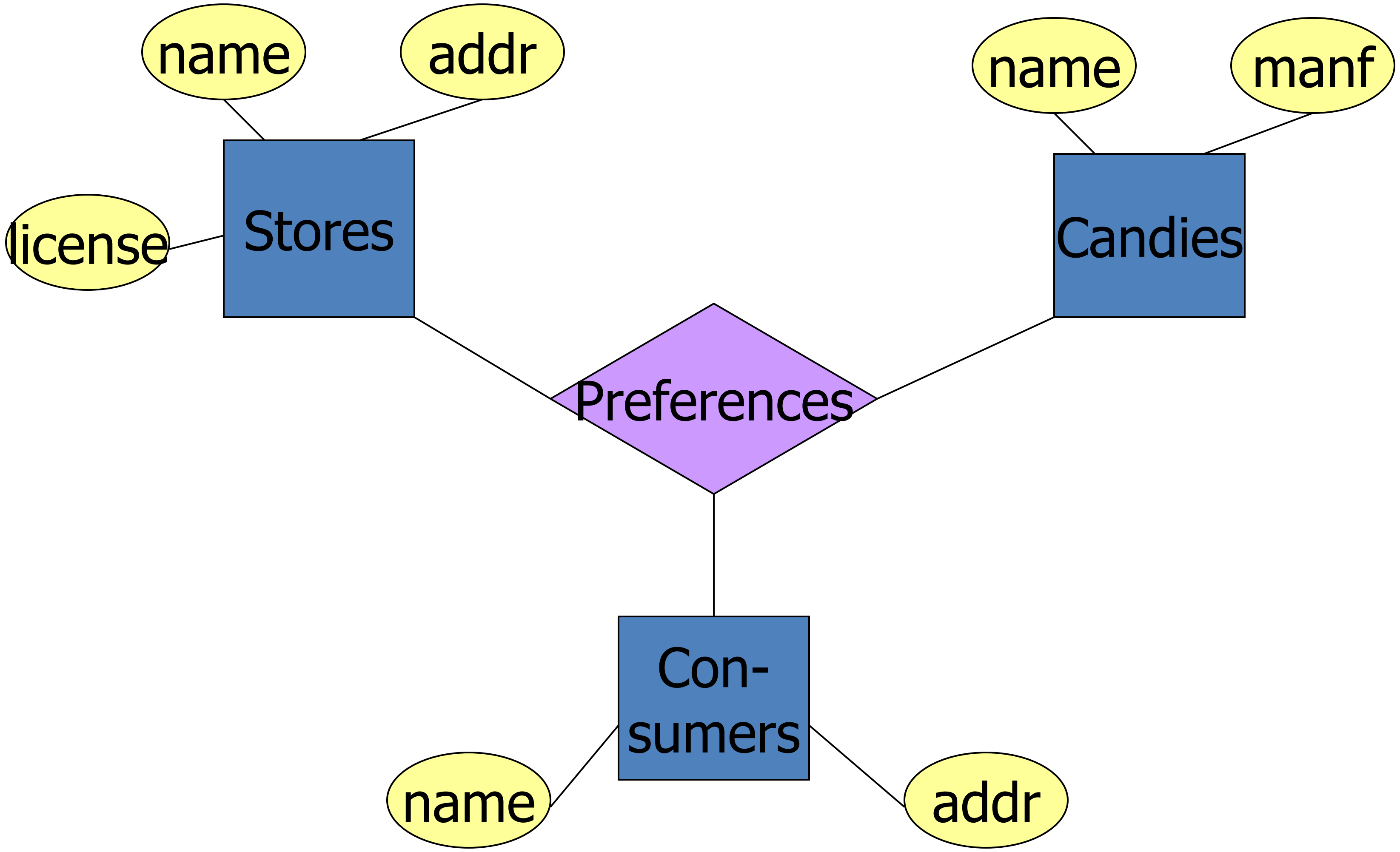
Consumers like some candies.

Consumers frequent some stores.

Multiway Relationships

- Sometimes, we need a relationship that connects more than two entity sets.
- Suppose that consumers will only buy certain candies at certain stores.
  - Our three binary relationships Likes, Sells, and Frequents do not allow us to make this distinction.
  - But a 3-way relationship would.

Example


name
addr
license
Stores
name
manf
Candies
Preferences
Con-
sumers
name
addr

A Typical Relationship Set

Store	Consumer	Candy
7-11	Ann	Kitkat
Kroger	Ann	Twizzler
Kroger	Ann	Snickers
7-11	Bob	Twizzler
7-11	Bob	Kitkat
7-11	Cal	Kitkat
Kroger	Cal	Reeses

One-One Relationships

- In a *one-one* relationship, each entity of either entity set is related to at most one entity of the other set.
- Example: Relationship Best-seller between entity sets Manfs (manufacturer) and Candies.
  - A candy cannot be made by more than one manufacturer, and no manufacturer can have more than one best-seller (assume no ties).

Weak Entity-Set Rules

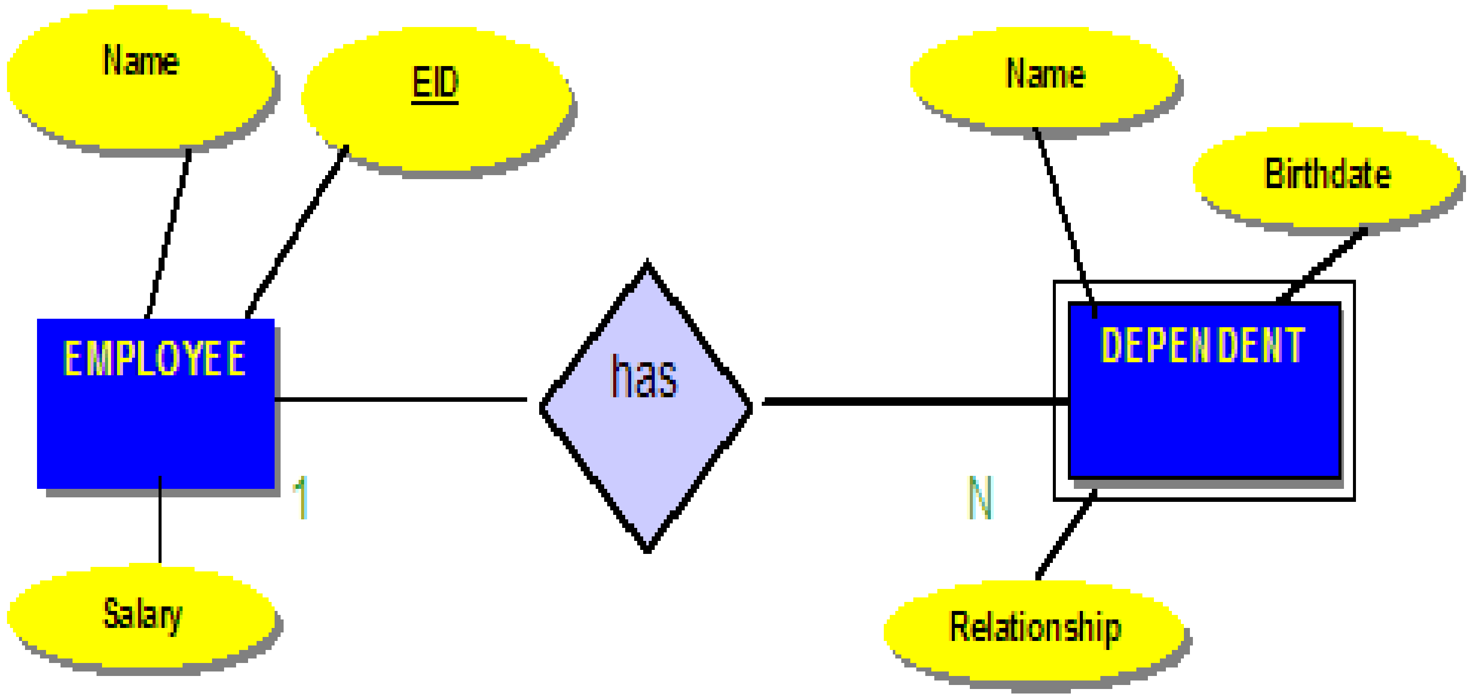
- A weak entity set has one or more many-one relationships to other (supporting) entity sets.
  - Not every many-one relationship from a weak entity set need be supporting.
- The key for a weak entity set is its own underlined attributes and the keys for the supporting entity sets.
  - E.g., (player) number and (team) name is a key for Players in the previous example.

Weak Entity Example

- name is almost a key for football players, but there might be two with the same name.
- number is certainly not a key, since players on two teams could have the same number.
- But number, together with the team name related to the player by Plays-on should be unique.

Weak Entity Example

The Employee entity would be the Strong Entity while the optional Dependent Entity would be the Weak Entity



When Do We Need Weak Entity Sets?

- The usual reason is that there is no global authority capable of creating unique ID’s.
- Example: it is unlikely that there could be an agreement to assign unique player numbers across all football teams in the world.

Design Techniques

1. Avoid redundancy.
2. Limit the use of weak entity sets.
3. Don’t use an entity set when an attribute will do.

Avoiding Redundancy

- *Redundancy* occurs when we say the same thing in two or more different ways.
- Redundancy wastes space and (more importantly) encourages inconsistency.
  - The two instances of the same fact may become inconsistent if we change one and forget to change the other.