CSC 370

Data Modelling

Major topics

- **Entity-Relationship diagrams**
- **Weak entity sets**
- **Converting E/R diagrams to relations**
- **UML-related diagramming**
- **Object Description Language (ODL)**

Entity-Relationship Model: Introduction

- E/R diagramming model allows us to visually represent database-schema designs
  - We show structure, not dynamic behavior
  - Can denote some constraints
  - For those without background in databases, somewhat easier to understand proposed designs
- Resulting designs are called **entity-relationship diagrams** (or E/R diagrams)
- We must also later convert E/R diagrams into relations

Some motivation

- Design can be difficult...
  - ... and consequences of getting the design wrong can be serious.
- The client may assert that they know what they want in a database...
  - ... but they don't know precisely what should be in it.
  - ... nor do they exactly understand the nuances of how the data is related
- Sketching out key components is one effective way to design and develop an actual database

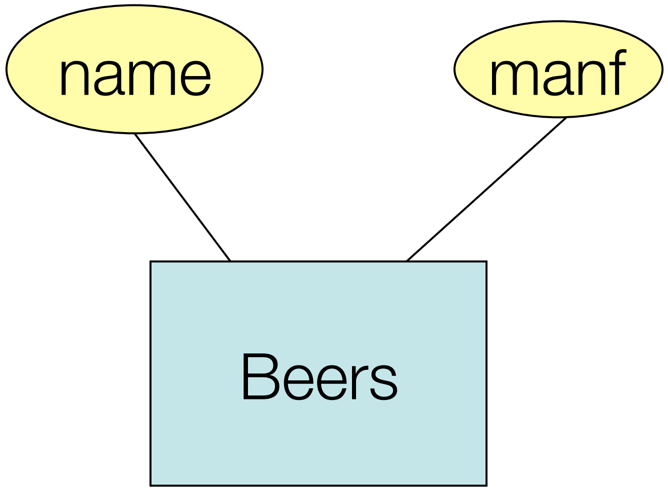
Vocabulary

- **Entity**
  - a thing or object
  - often is something which has an identity
- **Entity Set**
  - collection of similar entities
  - quite similar to the notion of "instances of a class" in Java
- **Attribute**
  - property of an entity
  - all entities belong to the same entity set will have the same number and type of attributes
  - simple values (i.e., not structs, lists, sets, etc.)

E/R diagrams

- There are a few "dialects" of E/R diagrams
- We will use the one where:
  - entity set = rectangle
  - attribute = oval
  - relationships = diamond
- Even if the actual symbols change, reasoning about E/R diagrams is the same from dialect to dialect

Example

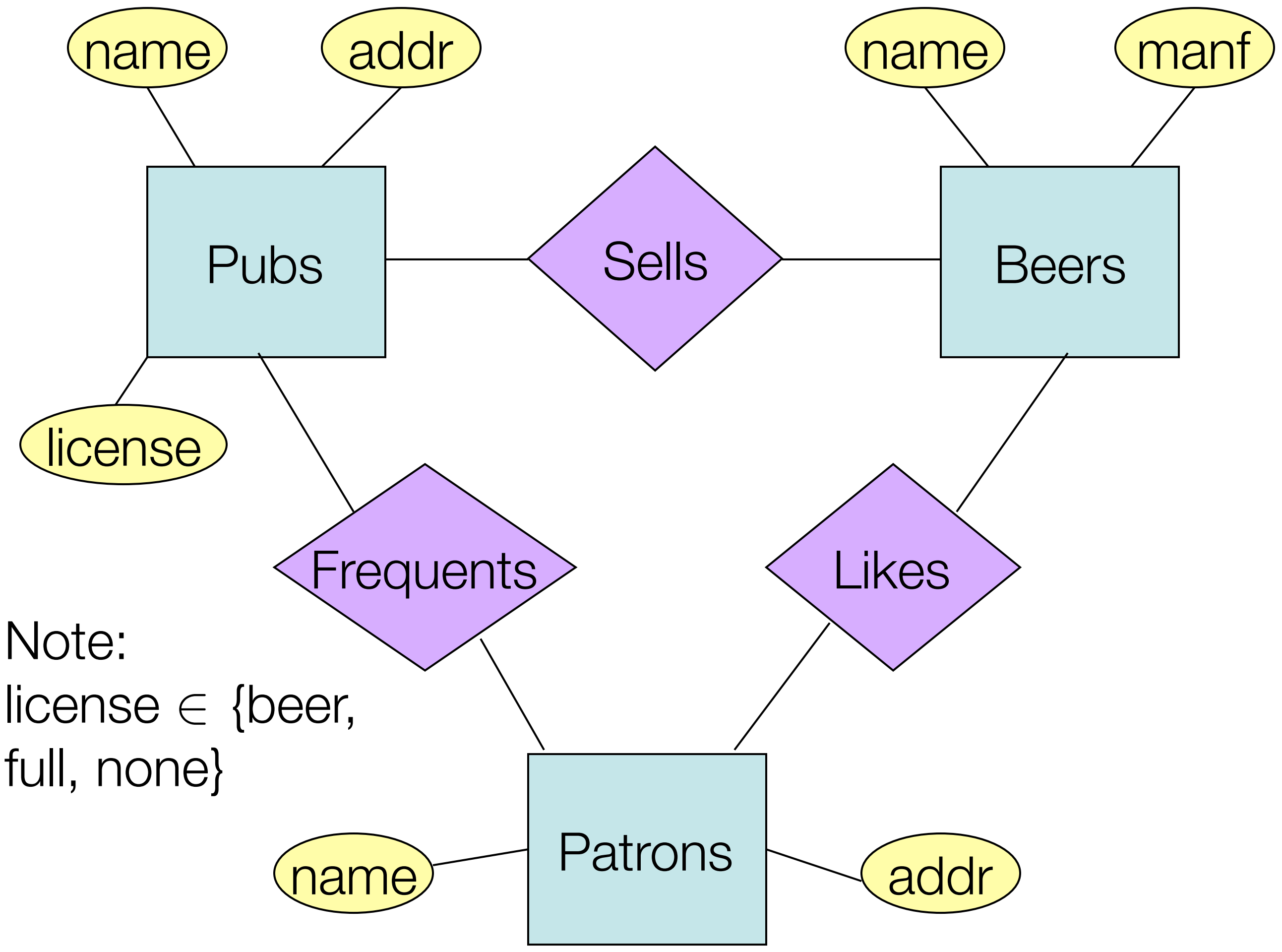


- Entity set **Beers** has two attributes, **name** and **manf** (manufacturer).
- Each Beers entity has values for these two attributes, e.g. ("Blue", "Labatt's")
  - values are not shown in the diagram

Relationships

- We connect two or more entity sets via a relationship
- Relationship:
  - Diamond
  - Lines connect the diamond to each of the entity sets involved in the relationship
  - No strict rule regarding the placement of entities in the relationship.
  - Convention: Lay out entities from left to right, top to bottom (i.e., to help with "reading" of the diagram)...
  - ... but this is not always possible.

Example: Relationships



Note:
license $\in$ {beer, full, none}

Pub(s) sell some beers.

Patron(s) like some beers.

Patron(s) frequent some pubs.

Relationship Set

- The **value** of a entity set at any one time is the **set of entities that belong to it**
  - In our example: value of Pub is the set of all pubs in our database
- The **value** of a relationship is a **relationship set**
  - This is a set of tuples with one component for each related entity set.

Example: relationship set

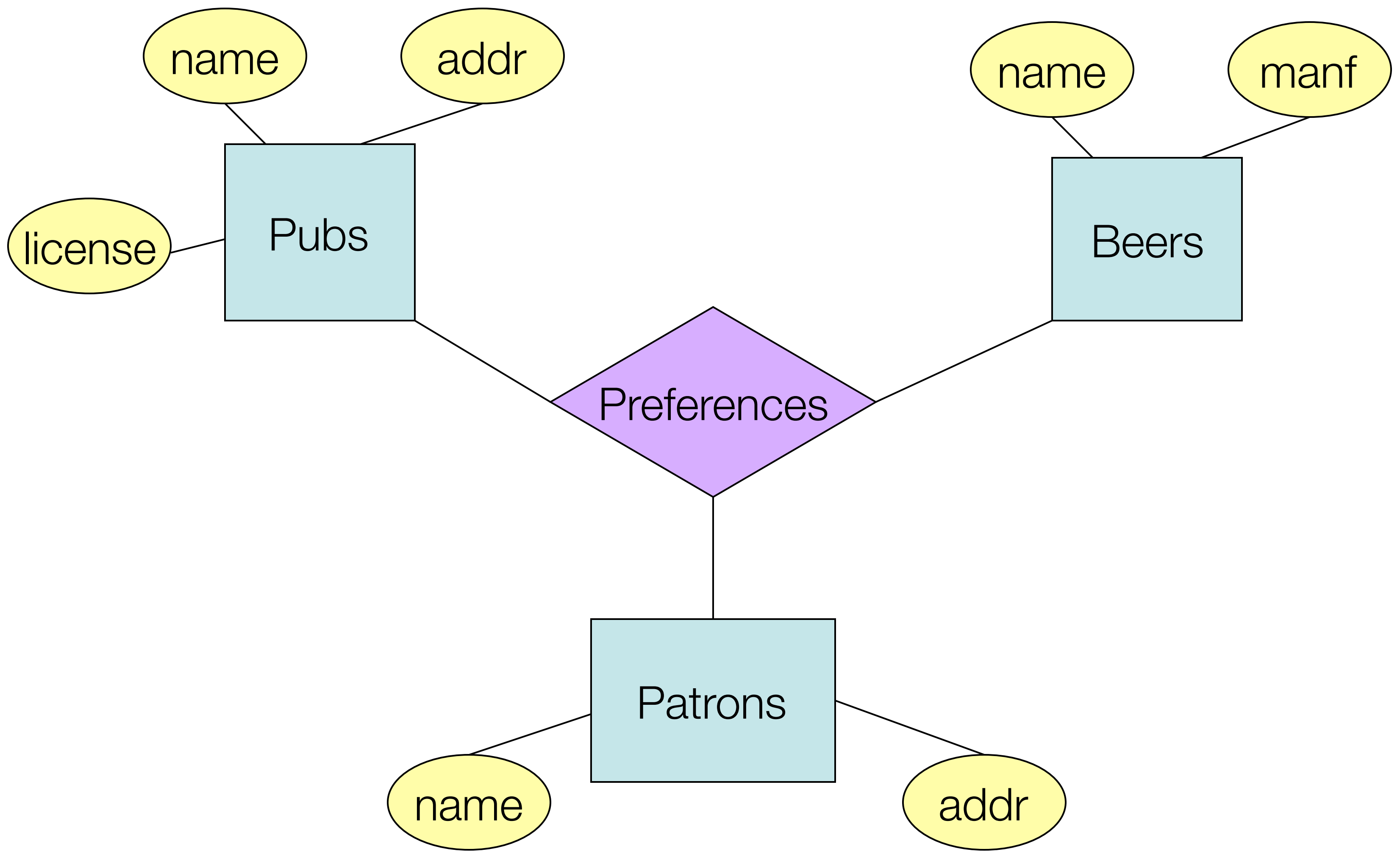
- For the relationship named Sells, we might have a relationship set as shown here.
- Note:
  - We're making some big assumptions here about how we identify individual entities!

pub	**beer**
Cheers	Blue
Cheers	Blue Buck
Cheers	Stella Artois
Parched Programmer	Blue
Parched Programmer	Amnesiac
Parched Programmer	Chimay Blue

Multiway Relationships

- We do to have a relationship set that connects more than two entity sets
- Example: Suppose that patrons will only imbibe (i.e., drink) certain beers at certain pubs.
  - Our three existing relationships Like, Sells, and Frequents – all of them binary – cannot allow us to model this.
  - However, a three-way relationship named Preferences could begin to let us do this.

Example: Three-way Relationship



Example: relationship set for Preferences

pub	patron	beer
Cheers	Cliff	Blue
Parched Programmer	Cliff	Blue Buck
Parched Programmer	Cliff	Stella Artois
Cheers	Norm	Blue
Cheers	Norm	Amnesiac
Cheers	Bill G	Coors Light
Parched Programmer	Bill G	Bud Light

Note: Although we can now express this three-way relationship, we have not constrained data as originally suggested (i.e., right now "Bill G" could still like "Bud Light" at "Cheers"). We'll bolt on this constraint shortly.