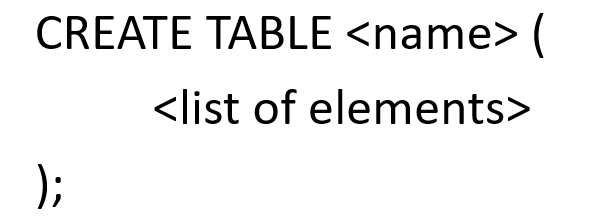
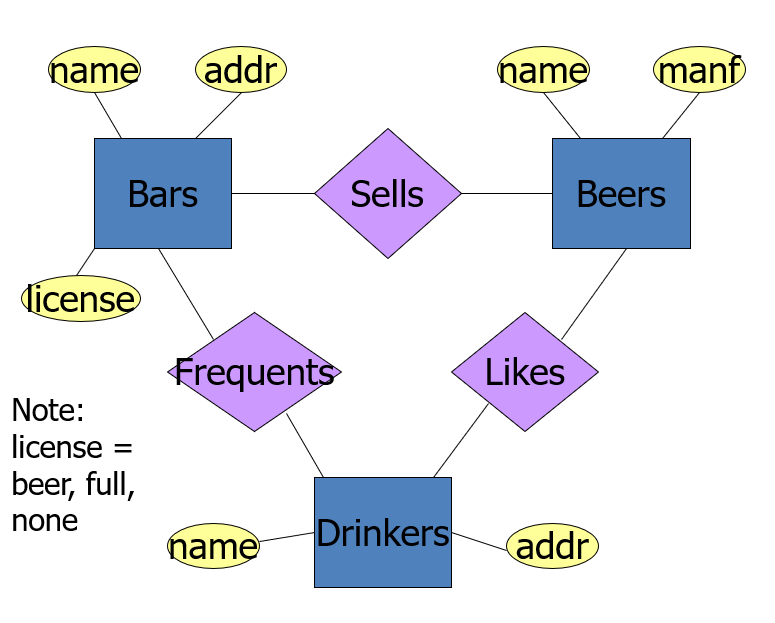
**Database Management Systems** - Collection of interrelated data and set of programs to access the data  
**Advantage of DBMS vs. file system** -Convenient and efficient processing of data  
**Describe physical level / logical level / view level abstraction**.  
View level: application programs hide details of data types  
Logical level: What is the data? Data types and such  
Physical level: How is the data stored?  
**Data Model** - A collection of tools for describing data, relationships among data items, semantics. Can be represented as ER models  
**Data Definition Language (DDL)** - Defines the database schema and constraints  
**Data Manipulation Language (DML)** - Language to access and manipulate the data. A query language  
**Procedural Query Language**: user specifies what data is required and how to get that data  
**Non procedural Query Language**: user specifies what data is required without how to get  
**Transaction** - Unit of work to be executed atomically and in isolation  
**Concurrency control is important to ensure consistency**  
***A*tomicity**: all-or-nothing of the transaction’s effect will take place  
***C*onsistency**: each transaction leaves the system in a consistent state  
***I*solation**: each transaction must appear to be executed as if no others are executed at the same time  
***D*urabilit**y: effect of a transaction must never be lost after the transaction is completed  
**Relation** – a table with columns and tuples  
**Database** – collection of relations  
**Relation schema**: relation name and attribute list  
**database schema** - set of all relation schemas in the database  
SQL is a query language with a data definition component; it is both a DML and a DDL, and it is non-procedural language  
A **key** is a single (or list of) attributes that can differentiate two nearly identical tuples. If a relation has a key, then no two tuples may match in all attributes. There can only be one **PRIMARY KEY** per relation, but multiple **UNIQUE** keys are allowed.  
**Create a Relation**:  **Delete a Relation**: 

**Attribute** – basic element in a relation (column).  
**Data type**s include: INT, FLOAT, CHAR(n) VARCHAR(n), DATE, TIME  
**Relational Algebra** – Mathematical system including operands and operators  
**Operands** – variables or values  
**Operators** – symbols denoting procedures  
**Union, Intersection and Difference** work exactly as you would think, but both operands MUST have the same relation schema  
R1 := **σ*C***(R2**) selection** ( all the tuples of R2 that satisfy condition c )  
R1 := **π*L***(R2**) projection** ( all the tuples of R2 with only the attributes on the list L, removing duplicates )  
R3 := R1 **Χ** R2 **cartesian product** ( as per usual, but it is R1attributes+R2attributes in that order)  
R3 := R1 **⋈*C*** **R2 theta join** (cartesian product then selection)  
R3 := R1 **⋈** R2 **natural join** ( theta join but w/o selection. Will equate attributes of same name as well )  
R1 := **ρ**R1(A1,…,A*n*)(R2) **rename** ( use to rename an attribute )  
**Bag** – a set, but an element can appear more than once  
**SQL Query** consists of: **SELECT FROM WHERE**  
**Rename in SQL** : SELECT ‘name’ AS ‘newname’  
**Conditionals Allowed in the WHERE clause: AND OR NOT = <> < > <= >=**

Write the create table SQL statement for the **Outcomes** relation, that includes the foreign key constraint for the attributes ship (corresponding to the name attribute in the Ships relation) and battle (corresponding to the name attribute in the Battles relation).

CREATE TABLE **Outcomes** (

ship CHAR(20) REFERECES Ships(name),

battle CHAR(20) REFERECES Battles(name),

results Date) ;

Write a relational algebra constraint that no ship could have participated in a battle after it was sunk.

All ships and dates that were sunk:

R1:= π ship, date ϭ Battles.name-Outcomes.battle AND result=’sunk’ (Outcomes X Battles)

ϭ R1.ship=Outcome.ship AND R1.date < Battles.date AND Battles.name-Outcomes.battle  (Outcomes X Battles X R1) = ∅

Consider the following two DDL statements:

1. CREATE TABLE **Ships** (

name CHAR(30),

class CHAR(30) CHECK ( class IN (SELECT class FROM Classes)),

launched Date );

1. CREATE TABLE **Ships** (

name CHAR(30),

class CHAR(30) References Classes.class,

launched Date );

Are these two statements equivalent?

Not. Statement 2 enforces referential integrity constraint that a ship may only belong to a class that is represented in the Classes relation. This constraint is enforced when 1) a new record is inserted to the Ships relations, 2) a record in the Ships relation is modified, OR 3) a record in the Classes relation is modified or deleted.

Statement 1 enforces the constraint that a ship may only belong to a class that is represented in the Classes relation. This constraint is enforced only when the attribute class of the Ships relation is updated or inserted. It is NOT checked if the Classes relation is modified.