Review: Participation Constraints

- Does every department have a manager?
  - If so, this is a participation constraint: the participation of Departments in Manages is said to be total (vs. partial).

- Every did value in Departments table must appear in a row of the Manages table (with a non-null ssn value!)
Participation Constraints in SQL

- We can capture participation constraints involving one entity set in a binary relationship, but little else (without resorting to CHECK constraints).

```sql
CREATE TABLE Dept_Mgr(
  did  INTEGER,
  dname CHAR(20),
  budget REAL,
  ssn  CHAR(11) NOT NULL,
  since DATE,
  PRIMARY KEY (did),
  FOREIGN KEY (ssn) REFERENCES Employees,
  ON DELETE NO ACTION);
```
A weak entity can be identified uniquely only by considering the primary key of another (owner) entity.

- Owner entity set and weak entity set must participate in a one-to-many relationship set (1 owner, many weak entities).
- Weak entity set must have total participation in this identifying relationship set.
Translating Weak Entity Sets

• Weak entity set and identifying relationship set are translated into a single table.
  – When the owner entity is deleted, all owned weak entities must also be deleted.

```sql
CREATE TABLE Dep_Policy (
    pname CHAR(20),
    age INTEGER,
    cost REAL,
    ssn CHAR(11) NOT NULL,
    PRIMARY KEY (pname, ssn),
    FOREIGN KEY (ssn) REFERENCES Employees,
    ON DELETE CASCADE)
```
Review: Binary vs. Ternary Relationships

Bad design

Better design

• What are the additional constraints in the 2nd diagram?
Binary vs. Ternary Relationships (Cont.)

- The key constraints allow us to combine Purchaser with Policies and Beneficiary with Dependents.

```sql
CREATE TABLE Policies (  
policyid INTEGER,  
cost REAL,  
ssn CHAR(11) NOT NULL,  
PRIMARY KEY (policyid),  
FOREIGN KEY (ssn) REFERENCES Employees,  
ON DELETE CASCADE);
```

- Participation constraints lead to NOT NULL constraints.

```sql
CREATE TABLE Dependents (  
pname CHAR(20),  
age INTEGER,  
policyid INTEGER,  
PRIMARY KEY (pname, policyid),  
FOREIGN KEY (policyid) REFERENCES Policies,  
ON DELETE CASCADE);
```
ISA (‘is a’) Hierarchies

- As in C++, or other PLs, attributes are inherited.
- If we declare A ISA B, every A entity is also considered to be a B entity.

Overlap constraints: Can Joe be an Hourly_Emps as well as a Contract_Emps entity? (Allowed/disallowed)

Covering constraints: Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? (Yes/no)

Reasons for using ISA:
- To add descriptive attributes specific to a subclass.
- To identify entities that participate in a relationship.
Translating ISA Hierarchies to Relations

• General approach:
  - 3 relations: Employees, Hourly_Emps and Contract_Emps.
    • *Hourly_Emps*: Every employee is recorded in Employees. For hourly emps, extra info recorded in Hourly_Emps (*hourly_wages, hours_worked, ssn*); must delete Hourly_Emps tuple if referenced Employees tuple is deleted.
    • *Contract_Emps*: Every employee is recorded in Employees. Extra info recorded in Contract_Emps (contract_id); must delete Contract_Emps tuple if referenced Employees tuple is deleted.
    • Queries involving all employees easy, those involving just Hourly_Emps require a join to get some attributes.

• Alternative: Just Hourly_Emps and Contract_Emps.
  - *Hourly_Emps*: *ssn, name, lot, hourly_wages, hours_worked.*
  - Similar for Contract_Emps
  - Each employee must be in one of these two subclasses.
Views

• A \textit{view} is just a relation, but we store a \textit{definition}, rather than a set of tuples.

\begin{verbatim}
CREATE VIEW YoungStudents (name, grade)
AS SELECT S.name, E.grade
FROM Students S, Enrolled E
WHERE S.sid = E.sid and S.age<21;
\end{verbatim}

• Views can be dropped using the \textbf{DROP VIEW} command.
Views and Security

- Views can be used to present necessary information (or a summary), while hiding details in underlying relation(s).
  - Given YoungStudents, but not Students or Enrolled, we can find young students who are enrolled, but not the *cids* of the courses they are enrolled in.
Relational Model: Summary

• A tabular representation of data.
• Simple and intuitive, currently the most widely used.
• Integrity constraints can be specified by the DBA, based on application semantics. DBMS checks for violations.
  – Two important ICs: primary and foreign keys
  – In addition, we always have domain constraints.
• Powerful and natural query languages exist.
• Rules to translate ER to relational model