Today

- Integrity Constraints
- ER -> Relational
- Class Participation Exercise

Integrity Constraints

- ***Integrity Constraint***: a condition specified on the database schema and restricts the data that can be stored.
  - Created when schema is defined
  - Checked when relations are modified
- If a database instance satisfies all the integrity constraints on the schema, then it is a ***legal*** *instance.*
- Changes to the database are not allowed to violate any integrity constraint
- Helps to prevent a whole class of errors and keeps database in a valid state

Keys

- (From ER) **Key**: minimal set of attributes that uniquely identifies an entity in an entity set
- Similarly, in relational model, a **key constraint** states that a certain minimal subset of fields of a relation is a unique identifier for that tuple
- Any set of fields that uniquely identifiers a tuple according to a key constraint is called a **candidate key** for the relation (“key”)
- Out of all candidate keys, can specify a single **primary key** for a relation. Can be used to refer to this tuple elsewhere in the database

Keys

```
CREATE TABLE Students ( sid      CHAR(20) ,
                        name     CHAR(30) ,
                        login    CHAR(20) ,
                        age      INTEGER,
                        gpa      REAL,
                        UNIQUE (name, age),
                        CONSTRAINT StudentsKey PRIMARY KEY (sid) )
```

- or "PRIMARY KEY (sid)"
- or "sid CHAR(20) PRIMARY KEY"

NULL Constraints

- By default, most columns (except columns specified as primary keys) can have NULL values
- Can add a constraint that a field can never be NULL
- Could be useful for participation constraints

```
CREATE TABLE Student (
     sid INT PRIMARY KEY,
     name VARCHAR(20) NOT NULL)
```

Foreign Key Constraints

- ***Foreign Key***: Set of fields in one relation that is used to refer to a tuple in another relation.
  - Corresponds to the primary key in the second relation
  - “Logical pointer”

```
CREATE TABLE EnrolledIn (
        sid INT references Students
        cid INT references Courses
        grade CHAR (2)
        primary key(sid,cid) )
```

Referential Integrity

- ***Referential Integrity*** requires that foreign keys refer to a valid, existing primary key in another relation
- On INSERT, reject action if foreign key refers to a tuple that doesn't exist
- On DELETE or UPDATE, either
  - Disallow action (default)
  - Cascade changes
  - Set foreign key to default value or NULL

Referential Integrity

```
CREATE TABLE Enrolled (
          sid INT
          cid INT
          grade CHAR(2)
          PRIMARY KEY (sid,cid)
          FOREIGN KEY (sid)
          REFERENCES Students
               ON DELETE CASCADE
               ON UPDATE CASCADE)
```

Referential Integrity

```
CREATE TABLE Enrolled (
          sid INT
          cid INT
          grade CHAR(2)
          PRIMARY KEY (sid,cid)
          FOREIGN KEY (sid)
          REFERENCES Students
               ON DELETE CASCADE
               ON UPDATE SET DEFAULT)
```

ER -> Relational

- ER model is convenient for representing an initial, high-level database design
- Given ER model, use a standard approach to generate a relational database schema that approximates the ER design
- Approximate because we may not capture all the constraints in the ER design (may be possible in SQL but possibly very expensive)

Mapping Entity Sets

- Create table for each entity set
- Map attributes to fields
- Declare primary key

Mapping Relationship Set (no key constraints)

- Create table for relationship set
- Add primary keys of entity sets participating in the relationship as primary keys of the relation
- Map attributes of the relationship to fields

Mapping Relationship Set (with key constraints)

- Option 1:
  - Create table for relationship set
  - Add primary keys of participating entity sets as fields
  - Map attributes of the relationship to fields
  - Declare primary key using key fields from source entity set (where the arrow is coming from)

Mapping Relationship Set (with key constraints)

- Option 1:


```
CREATE TABLE manages(
      employee_id          integer,
      department_id        integer,
      PRIMARY KEY (department_id),
      FOREIGN KEY (employee_id) REFERENCES Employees,
      FOREIGN KEY (department_id) REFERENCES Departments);
```

Mapping Relationship Set (with key constraints)

- Option 2
  - Add primary key of target entity as a field in the source

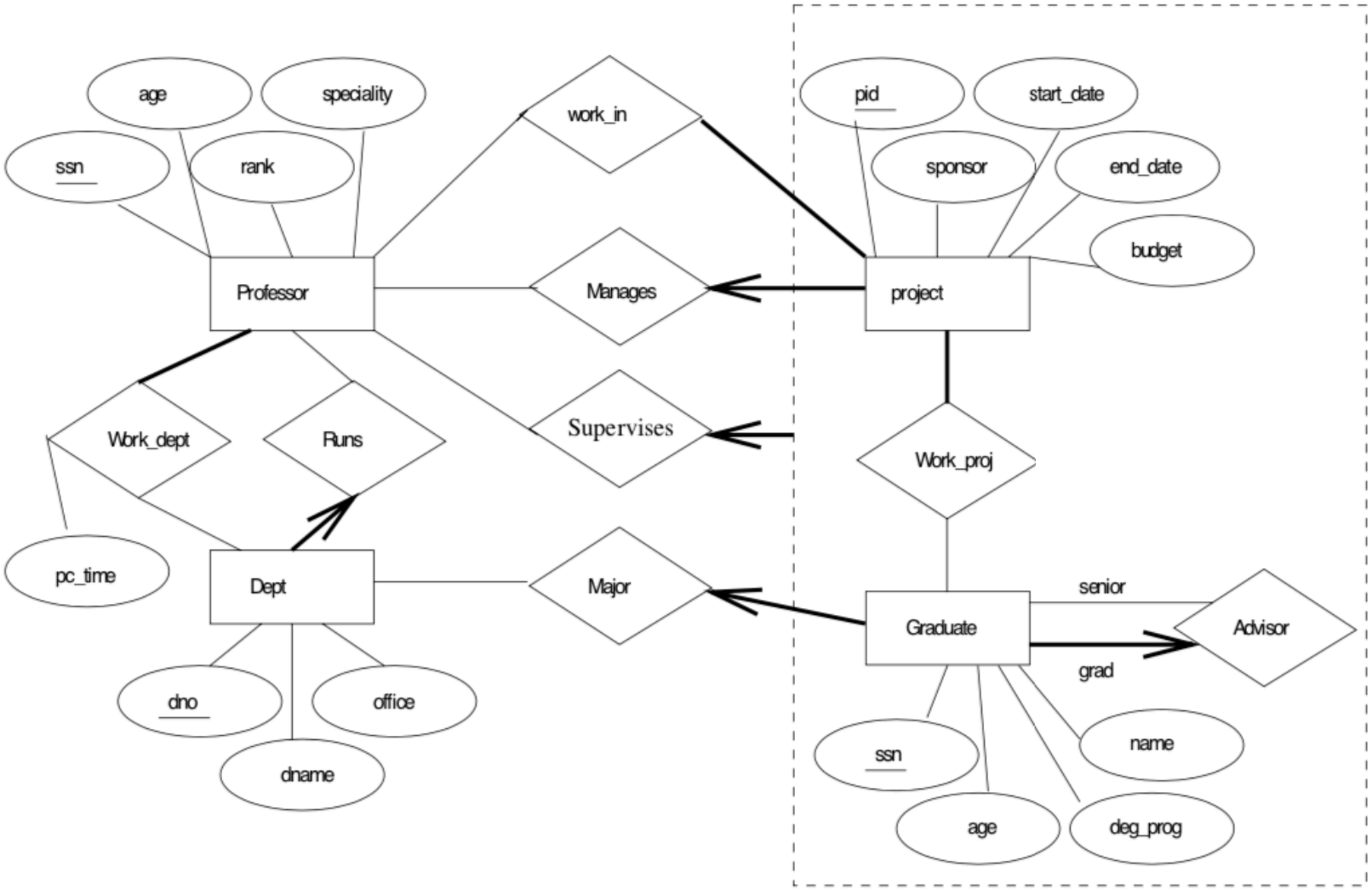
Mapping Relationship Set (with key constraints)

- Option 2:


```
CREATE TABLE department(
        department_id           integer,
        department_name         varchar(20),
        employee_id             integer,
        PRIMARY KEY (department_id),
        FOREIGN KEY (employee_id) REFERENCES Employees);
```

- Note: if we declare employee_id as NOT_NULL, can enforce participation constraint. Cannot easily do this with Option 1.

- Professors have an SSN, a name, an age, a rank, and a research specialty.
- Projects have a project number, a sponsor name (e.g., NSF), a starting date, an ending date, and a budget.
- Graduate students have an SSN, a name, an age, and a degree program
- Each project is managed by one professor
- Each project is worked on by one or more professors
- Professors can manage and/or work on multiple projects.
- Each project is worked on by one or more graduate students
- When graduate students work on a project, a professor must supervise their work on the project.
- Graduate students can work on multiple projects, in which case they will have a (potentially different) supervisor for each one.
- Departments have a department number, a department name, and a main office.
- Departments have a professor (known as the chairman) who runs the department.
- Professors work in one or more departments, and for each department that they work in, a time percentage is associated with their job.
- Graduate students have one major department in which they are working on their degree.
- Each graduate student has another, more senior graduate student (known as a student advisor) who advises him or her on what courses to take.




```
CREATE TABLE Supervises(
        prof_ssn  CHAR(10),
        grad_ssn  CHAR(10),
        pid       INTEGER,
        PRIMARY KEY (prof_ssn, grad_ssn, pid),
        PRIMARY KEY (grad_ssn, pid),
        FOREIGN KEY (prof_ssn) REFERENCES Professors,
        FOREIGN KEY (grad_ssn) REFERENCES Graduates,
        FOREIGN KEY (pid) REFERENCES Projects);
```

NOTE: because of the arrow from the aggregation, each work_proj is only supervised by one professor. Therefore, The primary key on Supervises should only include grad_ssn and pid, but NOT prof_ssn. This is because for each unique work_proj (defined by a pid and grad_ssn combination), there should only be one entry in the Supervises table.