Relational Model to SQL

Data Model

Conceptual Design: ER to Relational to SQL

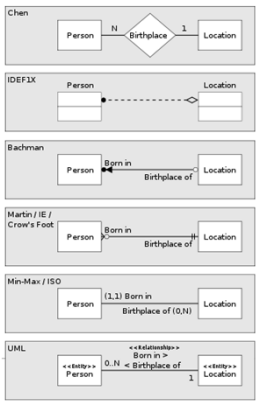
- How to represent
  - Entity sets,
  - Relationship sets,
  - Attributes,
  - Key and participation constraints,
  - Subclasses,
  - Weak entity sets

  . . . ?

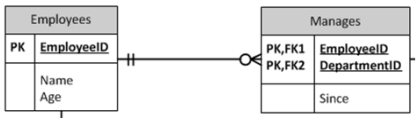
Problem Solving Steps

- Understand the business rules/requirements
- Draw the ER diagram
- Draw the Relational Model
- Write the SQL and create the database

NOTATIONS



CROW'S FEET

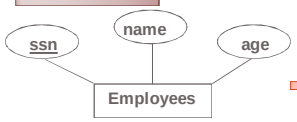


- Entities
- Relationships
  - 1-N
  - 1-1
  - N-N

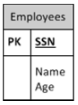
ENTITY SETS

- Entity sets are translated to tables.

ER Diagram



Relational

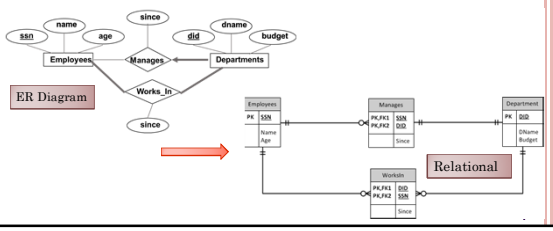


SQL

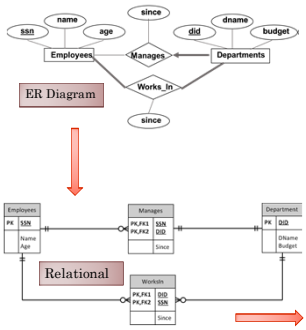
```
CREATE TABLE Employees
    (ssn CHAR(11),
    name CHAR(20),
    lot  INTEGER,
    PRIMARY KEY  (ssn));
```

Relationship Sets

- Relationship sets are also translated to tables.
  - Keys for each participating entity set (as foreign keys).
  - The combination of these keys forms a *superkey* for the table.
  - All descriptive attributes of the relationship set.



Relationship Sets

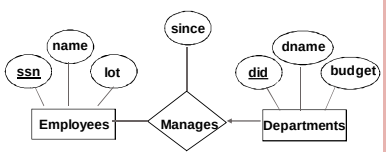


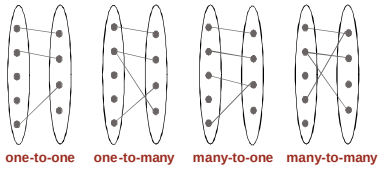
SQL

```
CREATE TABLE Works_In(
  ssn  CHAR(11),
  did  INTEGER,
  since  DATE,
  PRIMARY KEY (ssn, did),
  FOREIGN KEY (ssn)
     REFERENCES Employees,
  FOREIGN KEY (did)
     REFERENCES Departments);
```


Key Constraints

- Each dept has at most one manager, according to the key constraint on Manages.

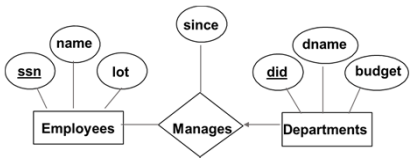




one-to-one one-to-many many-to-one many-to-many

Translation to relational model?

KEY CONSTRAINTS



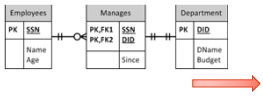
- 2 choices
  - Map relationship set to a table
    - Separate tables for Employees and Departments.
    - Note that *did* is the key now!
  - Since each department has a unique manager, we could instead combine Manages and Departments.

KEY CONSTRAINTS

- Choice 1
  - Map relationship set to a table
    - Separate tables for Employees and Departments.
    - Note that *did* is the key now!

ER Diagram

Relational



SQL

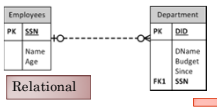
```
CREATE TABLE Manages(
  ssn CHAR(11),
  did INTEGER,
  since DATE,
  PRIMARY KEY (did),
  FOREIGN KEY (ssn) REFERENCES Employees,
  FOREIGN KEY (did) REFERENCES Departments)
```


KEY CONSTRAINTS

- Choice 2
  - Since each department has a unique manager
  - Combine Manages and Departments!!

ER Diagram

Relational



SQL

```
CREATE TABLE Dept_Mgr(
  did INTEGER,
  dname CHAR(20),
  budget REAL,
  ssn CHAR(11),
  since DATE,
  PRIMARY KEY (did),
  FOREIGN KEY (ssn) REFERENCES Employees)
```

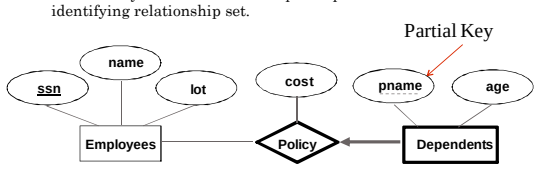
Participation Constraints

- We can capture participation constraints involving one entity set in a binary relationship, using NOT NULL.
- In other cases, we need CHECK constraints.

```
CREATE TABLE Dept_Mgr(
  did INTEGER,
  dname CHAR(20),
  budget REAL,
  manager CHAR(11) NOT NULL,
  since DATE,
  PRIMARY KEY (did),
  FOREIGN KEY (manager) REFERENCES Employees,
    ON DELETE NO ACTION)
```


Weak Entity Sets

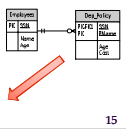
- A weak entity set can be identified uniquely only by considering the primary key of another (owner) entity set.
  - Owner entity set and weak entity set must participate in a one-to-many relationship set (one owner, many weak entities).
    - Weak entity has partial key. It's primary key is made of
      - Its own partial key
      - Primary key of Strong Entity
  - Weak entity set must have total participation in this identifying relationship set.



Weak Entity Sets

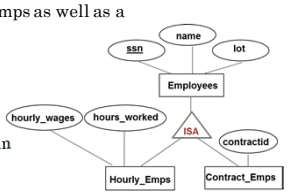
- Weak entity set and identifying relationship set are translated into a single table.
  - When the owner entity is deleted, all owned weak entities must also be deleted.


```
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)
```



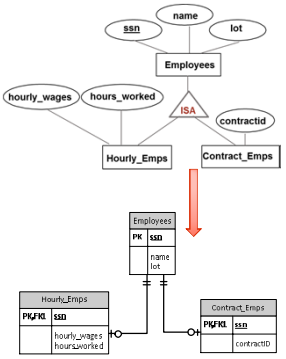
SUBCLASSES

- declare A ISA B
  - every A entity is also considered to be a B entity
  - A is a specialization of B
- Attributes of B are inherited to A.
- Overlap constraints
  - Can Joe be an Hourly_Emps as well as a Contract_Emps entity?
    - depends
- Covering constraints
  - Does every Employees entity either have to be an Hourly_Emps or a Contract_Emps entity?
    - depends

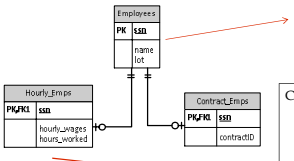


SUBCLASSES

- One table for each of the entity sets (superclass and subclasses).
- ISA relationship does not require additional table.
- All tables have the same key, i.e. the key of the superclass.
- E.g.: One table each for Employees, Hourly_Emps and Contract_Emps.
  - General employee attributes are recorded in Employees
  - For hourly emps and contract emps, extra info recorded in the respective relations



SUBCLASSES



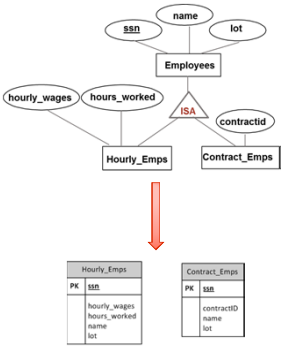

```
CREATE TABLE Employees(
  ssn  CHAR(11),
  name  CHAR(20),
  lot  INTEGER,
  PRIMARY KEY  (ssn))
```

```
CREATE TABLE Hourly_Emps(
  ssn  CHAR(11),
  hourly_wages  REAL,
  hours_worked INTEGER,
  PRIMARY KEY  (ssn),
  FOREIGN KEY  (ssn)
    REFERENCES Employees,
    ON DELETE CASCADE)
```

- Queries involving all employees easy, those involving just Hourly_Emps require a join to get their special attributes.

SUBCLASSES

- Alternative translation
  - Create tables for the subclasses only. These tables have all attributes of the superclass(es) and the subclass.
  - *This approach is applicable only if the subclasses cover the superclass.*
- Queries involving all employees difficult, those on Hourly_Emps and Contract_Emps alone are easy.
- Only applicable, if Hourly_Emps AND Contract_Emps COVER Employees



BINARY VS. TERNARY RELATIONSHIPS

- The key constraints allow us to combine Purchaser with Policies and Beneficiary with Dependents.
- Participation constraints lead to NOT NULL constraints.

```
CREATE TABLE Dependents (
  pname  CHAR(20),
  age  INTEGER,
  policyid  INTEGER NOT NULL,
  PRIMARY KEY (pname, policyid).
  FOREIGN KEY (policyid) REFERENCES Policies,
    ON DELETE CASCADE)
```

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


SUMMARY

- High-level design follows requirements analysis and yields a high-level description of data to be stored.
- ER model popular for high-level design.
  - Constructs are expressive, close to the way people think about their applications.
- Basic constructs: entities, relationships, and attributes (of entities and relationships).
- Some additional constructs: weak entities, subclasses, and constraints.
- ER design is subjective. There are often many ways to model a given scenario! Analyzing alternatives can be tricky, especially for a large enterprise.

Summary

- There are guidelines to translate ER diagrams to a relational database schema.
- However, there are often alternatives that need to be carefully considered.
- Entity sets and relationship sets are all represented by relations.
- Some constructs of the ER model cannot be easily translated, e.g. multiple participation constraints.

Walkthrough

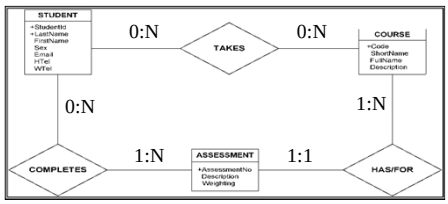
- Business Rules
  - A Student can take many Courses
  - A Course can be taken by many Students
  - A Student can complete many Assessments
  - An Assessment must be completed by at least one Student A Course must have at least one Assessment
  - An Assessment is for only one Course

Walkthrough

- Want to track information about students
  - Student {StudentId, LastName, FirstName, Sex, Email, HTel, WTel}
  - Course {Code, ShortName, FullName, Description}
  - Assessment {AssessmentNo, Description, Weighting}

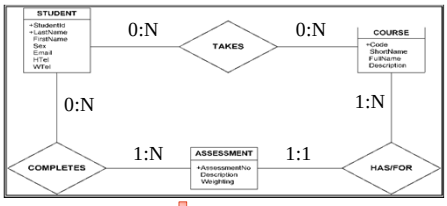
WALKTHROUGH

- Business Rules
  - A Student can take many Courses
  - A Course can be taken by many Students
  - A Student can complete many Assessments
  - An Assessment must be completed by at least one Student
  - A Course must have at least one Assessment
  - An Assessment is for only one Course

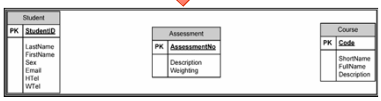


WALKTHROUGH

ER Diagram

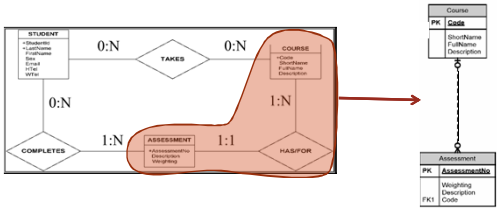


Relational



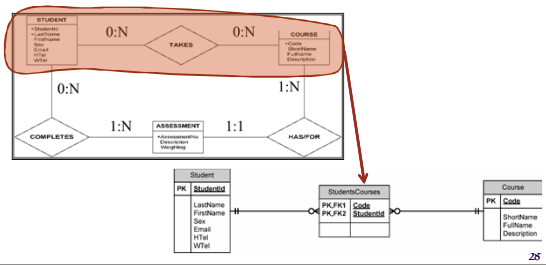
WALKTHROUGH

- Group together tables (formerly entities) and their relationships that have a cardinality of 0:1 or 1:1



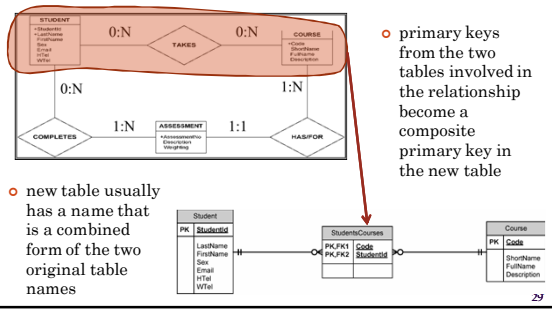
WALKTHROUGH

- The remaining relationships whose cardinalities are N (1 :N or 0:N) on both sides become new tables in the new relational model.



WALKTHROUGH

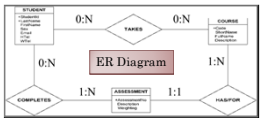
- remaining relationships whose cardinalities are 1:N or 0:N on both sides become new tables in the new relational model.

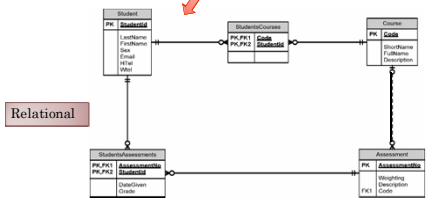


- primary keys from the two tables involved in the relationship become a composite primary key in the new table
- new table usually has a name that is a combined form of the two original table names

WALKTHROUGH

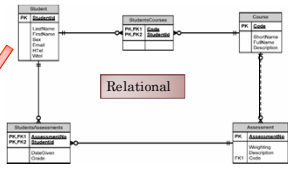
- Final tables
- Create in specific order?





WALKTHROUGH

- Final tables
- Create entities with no dependencies first

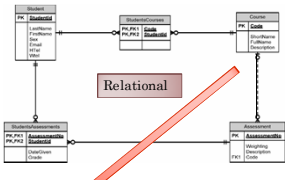


SQL

```
CREATE TABLE Student (
  StudentID BIGINT,
  LastName VARCHAR(100),
  FirstName VARCHAR(100),
  Sex CHAR(1),
  EMail VARCHAR(100),
  HTel VARCHAR(20),
  WTel VARCHAR(20),
  PRIMARY KEY (StudentID) );
```


WALKTHROUGH

- Final tables
- Create entities with no dependencies first

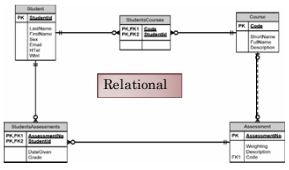


SQL

```
CREATE TABLE Course(
  Code VARCHAR(20),
  ShortName VARCHAR(100),
  FullName VARCHAR(100),
  Description VARCHAR(8000),
  PRIMARY KEY (Code) );
```

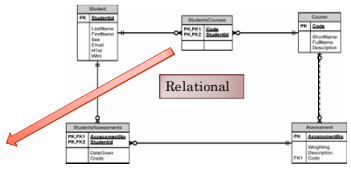

WALKTHROUGH

- Final tables
- Create tables dependent on entities.
- *Can we create StudentsAssessments?*



WALKTHROUGH

- Final tables



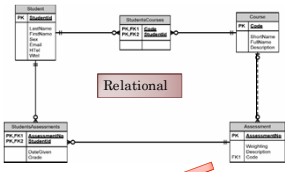
SQL

```
CREATE TABLE StudentsCourses(
  Code VARCHAR(20),
  StudentID BIGINT,
PRIMARY KEY (Code, StudentID),
  FOREIGN KEY (Code)
      REFERENCES Course,
  FOREIGN KEY (StudentID)
      REFERENCES Student);
```

Data types must be identical in all tables referencing the same field!

WALKTHROUGH

- Final tables

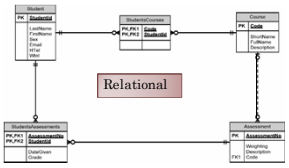


SQL

```
CREATE TABLE Assessment(
  AssessmentNo INTEGER,
  Code VARCHAR(20),
  Weighting DECIMAL(4,2),
  Description VARCHAR(100),
  PRIMARY KEY (AssessmentNo),
  FOREIGN KEY (AssessmentNo)
      REFERENCES Assessment);
```


WALKTHROUGH

- Final tables



SQL

```
CREATE TABLE StudentsAssessments(
  AssessmentNo INTEGER,
  StudentID BIGINT,
  DateGive DATE,
  Grade DECIMAL(4,2),
  PRIMARY KEY (AssessmentNo , StudentID),
  FOREIGN KEY (AssessmentNo)
      REFERENCES Assessment,
  FOREIGN KEY (StudentID)
      REFERENCES Student);
```