Relational Database Design

Translation of ER-diagram into Relational Schema

Dr. Sunnie S. Chung

CIS430/530
Learning Objectives

✓ Define each of the following database terms
  ✓ Relation
  ✓ Primary key
  ✓ Foreign key
  ✓ Referential integrity
  ✓ Field
  ✓ Data type
  ✓ Null value

✓ Discuss the role of designing databases in the analysis and design of an information system

✓ Learn how to transform an entity-relationship (ER) Diagram into an equivalent set of well-structured relations
Databanks

Real World → Model → Databank

Databank management system → Processing of queries and updates → Access to stored data

Physical databank

Queries → Answers
ER/EER to database schema
### Process of Database Design

- **Steps in translation:**
  - Entity sets to tables
  - Relationships to tables
  - Constraints
  - Weak entity sets

- **Logical Design**
  - Based upon the conceptual data model
  - Four key steps

  1. Develop a logical data model for each known user interface for the application using normalization principles.

  2. Combine normalized data requirements from all user interfaces into one consolidated logical database model

  3. Translate the conceptual E-R data model for the application into normalized data requirements

  4. Compare the consolidated logical database design with the translated E-R model and produce one final logical database model for the application
Entity Sets to Tables

- Each attribute of the E. S. becomes an attribute of the table

Relations:
- CUSTOMER(Customer_ID, Name, Address)
- PRODUCT(Product_ID, Description)
- ORDER(Order_Number, Customer_ID, Order_Date)
- LINE_ITEM(Order_Number, Product_ID, Order_Quantity)
- INVOICE(Invoice_Number, Order_Number)
- SHIPMENT(Invoice_Number, Product_ID, Ship_Quantity)
Relational Database Model

- Data represented as a set of related tables or relations

- Relation
  - A named, two-dimensional table of data. Each relation consists of a set of named columns and an arbitrary number of unnamed rows

- Properties
  - Entries in cells are simple
  - Entries in columns are from the same set of values
  - Each row is unique
  - The sequence of columns can be interchanged without changing the meaning or use of the relation
  - The rows may be interchanged or stored in any sequence
Relational Database Model

• Well-Structured Relation
  - A relation that contains a minimum amount of redundancy and allows users to insert, modify and delete the rows without errors or inconsistencies
Transforming E-R Diagrams into Relations

- It is useful to transform the conceptual data model into a set of normalized relations

- Steps
  1. Represent entities
  2. Represent relationships
  3. Normalize the relations
  4. Merge the relations
Refining the ER Design for the COMPANY Database

- Change attributes that represent relationships into relationship types
- Determine cardinality ratio and participation constraint of each relationship type
ER Diagrams, Naming Conventions, and Design Issues
Figure 7.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter and is summarized in Figure 7.14.
Design Choices for ER Conceptual Design

- Model concept first as an attribute
  - Refined into a relationship if attribute is a reference to another entity type
- Attribute that exists in several entity types may be elevated to an independent entity type
  - Can also be applied in the inverse
Attributes

- An entity is represented by a set of attributes, that is descriptive properties possessed by all members of an entity set.

Example:

Customer = (customer-id, customer-name, customer-street, customer-city)

Loan = (loan-number, amount)

- Domain - the set of permitted values for each attribute
Attributes

- **Composite Key (identifier):** Primary key composed of more than one attribute
- **Composite attribute:** Attribute that can be subdivided to yield additional attributes
- **Simple attribute:** Attribute that cannot be subdivided
- **Single-valued attribute:** Attribute that has only a single value
- **Multivalued attributes:** Attributes that have many values
Attributes

• **Multivalued (Set Valued) attributes**: Attributes that have many values and require creating:
  - Several new attributes, one for each component of the original multivalued attribute
  - A new entity composed of the original multivalued attribute’s components

• **Derived attribute**: Attribute whose value is calculated from other attributes
  - Derived using an algorithm
Figure 4.6 - Depiction of a Derived Attribute
Attribute types:

- **Simple** and **Composite** attributes.

- **Single-valued** and **Multi-valued** attributes
  - E.g. multivalued attribute: *phone-numbers*

- **Derived** attributes
  - Can be computed from other attributes
  - E.g. *age*, given date of birth
Multivalued Attributes

- Normalization Required to First Normal Form
- A multivalued attribute $M$ of an entity $E$ is represented by a separate table $EM$
  - Table $EM$ has attributes corresponding to the primary key of $E$ and an attribute corresponding to multivalued attribute $M$
  - Eg: Create Dept_Location table from Department table
    Dept_Location( Dnumber, Dlocation)
Multivalued Attribute: DLocations

Figure 10.8 Normalization into 1NF

(a) DEPARTMENT

(b) DEPARTMENT

(c) DEPARTMENT

Remember Pack₁(r) and Unpack₂(r) operators?
### DEPARTMENT

<table>
<thead>
<tr>
<th>DNAME</th>
<th>DNUMBER</th>
<th>MGRSSN</th>
<th>MGRSTARTDATE</th>
<th>MGRSTARTDATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Headquarters</td>
<td>1</td>
<td>888665555</td>
<td>19-Jun-71</td>
<td></td>
</tr>
<tr>
<td>Administration</td>
<td>4</td>
<td>987654321</td>
<td>01-Jan-85</td>
<td></td>
</tr>
<tr>
<td>Research</td>
<td>5</td>
<td>333445555</td>
<td>22-May-78</td>
<td></td>
</tr>
<tr>
<td>Automation</td>
<td>7</td>
<td>123456789</td>
<td>06-Oct-05</td>
<td></td>
</tr>
</tbody>
</table>

### DEPT_LOCATIONS

<table>
<thead>
<tr>
<th>DNUMBER</th>
<th>DLOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Houston</td>
</tr>
<tr>
<td>4</td>
<td>Stafford</td>
</tr>
<tr>
<td>5</td>
<td>Bellaire</td>
</tr>
<tr>
<td>5</td>
<td>Sugarland</td>
</tr>
<tr>
<td>5</td>
<td>Houston</td>
</tr>
</tbody>
</table>
Composite Attributes

Composite Attributes

name

first-name middle-initial last-name

address

street city state postal-code

Component Attributes

street-number street-name apartment-number
Three Ways in Transforming Composite Attribute

1. Flatten the Structure to One Attribute
eg: Address of Employee Table

2. Flatten the Structure to Multiple Attributes
eg: Name ➔ Lname, MI, Fname of Employee Table

3. Making the Composite attribute to a Separate Table
Flatten the Structure to Transform Composite Attribute

<table>
<thead>
<tr>
<th>FNAME</th>
<th>MINIT</th>
<th>LNAME</th>
<th>SSN</th>
<th>BDATE</th>
<th>ADDRESS</th>
<th>SEX</th>
<th>SALARY</th>
<th>SUPERSSN</th>
<th>DNO</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>B</td>
<td>Smith</td>
<td>1234567</td>
<td>09–Jan–55</td>
<td>731 Fondren, Houston, TX</td>
<td>M</td>
<td>30000</td>
<td>987654321</td>
<td>5</td>
</tr>
<tr>
<td>Franklin</td>
<td>T</td>
<td>Wong</td>
<td>333445</td>
<td>08–Dec–45</td>
<td>638 Voss, Houston, TX</td>
<td>M</td>
<td>40000</td>
<td>888665555</td>
<td>5</td>
</tr>
<tr>
<td>Joyce</td>
<td>A</td>
<td>English</td>
<td>453453</td>
<td>31–Jul–62</td>
<td>5631 Rice, Houston, TX</td>
<td>F</td>
<td>25000</td>
<td>333445555</td>
<td>5</td>
</tr>
<tr>
<td>Ramesh</td>
<td>K</td>
<td>Narayan</td>
<td>666884</td>
<td>15–Sep–52</td>
<td>975 Fire Oak, Humble, TX</td>
<td>M</td>
<td>38000</td>
<td>333445555</td>
<td>5</td>
</tr>
<tr>
<td>James</td>
<td>E</td>
<td>Borg</td>
<td>888665</td>
<td>10–Nov–27</td>
<td>450 Stone, Houston, TX</td>
<td>M</td>
<td>55000</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Jennifer</td>
<td>S</td>
<td>Wallace</td>
<td>987654</td>
<td>20–Jun–31</td>
<td>291 Berry, Bellaire, TX</td>
<td>F</td>
<td>43000</td>
<td>888665555</td>
<td>4</td>
</tr>
<tr>
<td>Ahmad</td>
<td>V</td>
<td>Jabbar</td>
<td>987987</td>
<td>29–Mar–59</td>
<td>980 Dallas, Houston, TX</td>
<td>M</td>
<td>25000</td>
<td>987654321</td>
<td>4</td>
</tr>
<tr>
<td>Alicia</td>
<td>J</td>
<td>Zelaya</td>
<td>999887</td>
<td>19–Jul–58</td>
<td>3321 Castle, Spring, TX</td>
<td>F</td>
<td>25000</td>
<td>987654321</td>
<td>4</td>
</tr>
</tbody>
</table>
Option 3: Transforming Composite Attribute to a Table

Option 3: Make a Composite Attribute into Separate Table Customer_Address

Customer_Address

(Customer_Id, Street_Number, Street_Name, Apt_Num, City, State, Zip, Country)

• More Query Power for Data Analytic on Address
• No Need to Access Big Customer Table for Address Analytics
• Required Join with Address Table and Customer Table for other Customer Information
Transforming Composite and Multivalued Attribute

- Make the Composite and Multi Valued Attribute to a Separate Table

Customer_Address

(Customer_Id, Street_Number, Street_Name, Apt_Num, City, State, Zip, Country)
Alternative Notations for ER Diagrams

- Specify structural constraints on Relationships
  - Replaces Cardinality ratio (1:1, 1:N, M:N) and single/double line notation for Participation constraints
  - Associate a pair of integer numbers \((\text{min}, \text{max})\) with each participation of an entity type \(E\) in a relationship type \(R\), where \(0 \leq \text{min} \leq \text{max}\) and \(\text{max} \geq 1\)
Cardinality Ratio
(1:1, 1:N, M:N)

- **1:N**: Each dept has at most one manager on Manages.

Translation to relational model?
Figure 7.15
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.
Transforming E-R Diagrams into Relations

- In translating a relationship set to a relation, attributes of the relation must include:
  - The primary key for each participating entity set (as foreign keys).
    - This set of attributes forms a superkey for the relation.
  - All descriptive attributes of the relationship set

- The primary key must satisfy the following two conditions
  a. The value of the key must uniquely identify every row in the relation
  b. The key should be nonredundant
Transforming E-R Diagrams into Relations

Represent Relationships

- Binary 1:N Relationships
  - Add the Primary key attribute (or attributes) of the entity on the one side of the relationship as a Foreign key in the relation on the other (N) side
  - The one side migrates to the many side
Constraints on Binary Relationship Types

- **Cardinality ratio** for a binary relationship
  - Specifies maximum number of relationship instances that entity can participate in

- **Participation Constraint**
  - Specifies whether existence of entity depends on its being related to another entity
  - Types: total and partial
Attributes of Relationship Types

- Attributes of 1:1 or 1:N relationship types can be migrated to one entity type
- For a $1:N$ relationship type
  - Relationship attribute can be migrated only to entity type on N-side of relationship
- For $M:N$ relationship types
  - Some attributes may be determined by combination of participating entities
  - Must be specified as relationship attributes
Weak Entity Types

- Do not have key attributes of their own
  - Identified by being related to specific entities from another entity type
- Identifying relationship
  - Relates a weak entity type to its owner
- Always has a total participation constraint
Transforming **Binary 1:N Relationships** into Relations

- **Relationship:**
  
  CUSTOMER Places ORDER(s)

- **ORDER Table BEFORE Relationship:**
  
  `(Order_Number, Order_Date, Promised_Date)`

- **ORDER Table AFTER Relationship:**
  
  `(Order_Number, Order_Date, Promised_Date, Customer_ID)`

```sql
CREATE TABLE ORDER(
    Order_Number CHAR(1),
    Order_Date DATE,
    Promised_Date DATE,
    Customer_ID CHAR(1),
    PRIMARY KEY (Order_Number),
    FOREIGN KEY (Customer_ID) REFERENCES CUSTOMER(Customer_ID));
```
Transforming E-R Diagrams into Relations

- Binary or Unary 1:1
  - Three possible options
    a. Add the primary key of A as a foreign key of B
    b. Add the primary key of B as a foreign key of A
    c. Both
Binary 1 To 1 Relationship

**Employee(Manager) MANAGES Department**

### EMPLOYEE

<table>
<thead>
<tr>
<th>FNAME</th>
<th>MINIT</th>
<th>LNAME</th>
<th>SSN</th>
<th>BDATE</th>
<th>ADDRESS</th>
<th>SEX</th>
<th>SALARY</th>
<th>SUPERSSN</th>
<th>DNO</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>B</td>
<td>Smith</td>
<td>123456789</td>
<td>09–Jan–55</td>
<td>731 Fondren, Houston, TX</td>
<td>M</td>
<td>30000</td>
<td>987654321</td>
<td>5</td>
</tr>
<tr>
<td>Franklin</td>
<td>T</td>
<td>Wong</td>
<td>333445555</td>
<td>08–Dec–45</td>
<td>638 Voss, Houston, TX</td>
<td>M</td>
<td>40000</td>
<td>888665555</td>
<td>5</td>
</tr>
<tr>
<td>Joyce</td>
<td>A</td>
<td>English</td>
<td>453453453</td>
<td>31–Jul–62</td>
<td>5631 Rice, Houston, TX</td>
<td>F</td>
<td>25000</td>
<td>333445555</td>
<td>5</td>
</tr>
<tr>
<td>Ramesh</td>
<td>K</td>
<td>Narayan</td>
<td>666884444</td>
<td>15–Sep–52</td>
<td>975 Fire Oak, Humble, TX</td>
<td>M</td>
<td>38000</td>
<td>333445555</td>
<td>5</td>
</tr>
<tr>
<td>James</td>
<td>E</td>
<td>Borg</td>
<td>888665555</td>
<td>10–Nov–27</td>
<td>450 Stone, Houston, TX</td>
<td>M</td>
<td>55000</td>
<td>888665555</td>
<td>1</td>
</tr>
<tr>
<td>Jennifer</td>
<td>S</td>
<td>Wallace</td>
<td>987654321</td>
<td>20–Jun–31</td>
<td>291 Berry, Bellaire, TX</td>
<td>F</td>
<td>43000</td>
<td>888665555</td>
<td>4</td>
</tr>
<tr>
<td>Ahmad</td>
<td>V</td>
<td>Jabbar</td>
<td>987987987</td>
<td>29–Mar–59</td>
<td>980 Dallas, Houston, TX</td>
<td>M</td>
<td>25000</td>
<td>987654321</td>
<td>4</td>
</tr>
<tr>
<td>Alicia</td>
<td>J</td>
<td>Zelaya</td>
<td>999887777</td>
<td>19–Jul–58</td>
<td>3321 Castle, SPring, TX</td>
<td>F</td>
<td>25000</td>
<td>987654321</td>
<td>4</td>
</tr>
</tbody>
</table>

### DEPARTMENT

<table>
<thead>
<tr>
<th>DNAME</th>
<th>DNUMBER</th>
<th>MGRSSN</th>
<th>MGRSTARTDATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Headquarters</td>
<td>1</td>
<td>888665555</td>
<td>19–Jun–71</td>
</tr>
<tr>
<td>Administration</td>
<td>4</td>
<td>987654321</td>
<td>01–Jan–85</td>
</tr>
<tr>
<td>Research</td>
<td>5</td>
<td>333445555</td>
<td>22–May–78</td>
</tr>
<tr>
<td>Automation</td>
<td>7</td>
<td>123456789</td>
<td>06–Oct–05</td>
</tr>
</tbody>
</table>
Transforming E-R Diagrams into Relations

Represent Relationships

- Binary and higher $M:N$ relationships
  - Create another relation and include primary keys of all relations as primary key of new relation
## Association Class to Table

### ORDER

<table>
<thead>
<tr>
<th>Order_Number</th>
<th>Order_Date</th>
<th>Promised_Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>61384</td>
<td>2/17/2002</td>
<td>3/01/2002</td>
</tr>
<tr>
<td>62807</td>
<td>2/15/2002</td>
<td>3/01/2002</td>
</tr>
</tbody>
</table>

### ORDER LINE

<table>
<thead>
<tr>
<th>Order_Number</th>
<th>Product_ID</th>
<th>Quantity_Ordered</th>
</tr>
</thead>
<tbody>
<tr>
<td>61384</td>
<td>M128</td>
<td>2</td>
</tr>
<tr>
<td>61384</td>
<td>A261</td>
<td>1</td>
</tr>
<tr>
<td>62807</td>
<td>A261</td>
<td>2</td>
</tr>
<tr>
<td>62807</td>
<td>R149</td>
<td>1</td>
</tr>
<tr>
<td>62009</td>
<td>R149</td>
<td>2</td>
</tr>
</tbody>
</table>

### PRODUCT

<table>
<thead>
<tr>
<th>Product_ID</th>
<th>Description</th>
<th>(Other Attributes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>M128</td>
<td>Bookcase</td>
<td>—</td>
</tr>
<tr>
<td>A261</td>
<td>Wall unit</td>
<td>—</td>
</tr>
<tr>
<td>R149</td>
<td>Cabinet</td>
<td>—</td>
</tr>
<tr>
<td>62009</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Transforming Binary M:N Relationships into Relations

- Relationship Requests: Order Requests Products
  1. Create Table ORDERLINE for Relationship Requests
  2. Add PK of each side of Tables (Order_Number, Product_ID) as Foreign Keys
  3. Make composite of both attributes as Primary Key of the Table ORDERLINE:

```sql
CREATE TABLE ORDERLINE ( 
  Order_Number CHAR(10),
  Product_ID CHAR(10),
  Quantity_Ordered INTEGER,
  PRIMARY KEY (Order_Number, Product_ID),
  FOREIGN KEY (Order_Number) REFERENCES ORDER(Order_Number),
  FOREIGN KEY (Product_ID) REFERENCES PRODUCT(Product_ID));
```
Transforming E-R Diagrams into Relations

- **Unary 1:N Relationships**
  - Relationship between instances of a single entity type
  - Utilize a recursive foreign key
    - A foreign key in a relation that references the primary key values of that same relation

- **Unary M:N Relationships**
  - Create a separate relation
  - Primary key of new relation is a composite of two attributes that both take their values from the same primary key
**Figure 9.13b** Two Unary Relations — Bill-of-Materials Structure (M:N)

**Figure 9.13a** Two Unary Relations — EMPLOYEE with Manages Relationship (1:N)
Unary 1 to M to Table: MANAGES

**Figure 9-13a** Two Unary Relations —
EMPLOYEE with Manages Relationship (1:N)

<table>
<thead>
<tr>
<th>Emp_ID</th>
<th>Name</th>
<th>Mgr_ID</th>
<th>Birthdate</th>
</tr>
</thead>
<tbody>
<tr>
<td>11111</td>
<td>John Smith</td>
<td>33333</td>
<td>12/30</td>
</tr>
<tr>
<td>22222</td>
<td>Jane Doe</td>
<td>33333</td>
<td>03/23</td>
</tr>
<tr>
<td>33333</td>
<td>Anne Hana</td>
<td>99999</td>
<td>05/05</td>
</tr>
<tr>
<td>44444</td>
<td>Nick Shaw</td>
<td>99999</td>
<td>11/23</td>
</tr>
<tr>
<td>99999</td>
<td>Ron Birkman</td>
<td></td>
<td>07/20</td>
</tr>
</tbody>
</table>
Transforming Unary 1:N Relationships into Relations

- Relationship:
  EMPLOYEE (as Manager) Manages EMPLOYEE

- EMPLOYEE Table
  BEFORE Relationship:
  (Emp_ID, Name, Birthday)

- EMPLOYEE Table AFTER Relationship:
  (Emp_ID, Name, Birthday, Mgr_ID)

CREATE TABLE EMPLOYEE
  (Emp_ID CHAR(1),
   Name Varchar(30),
   Birthday DATE,
   Mgr_ID CHAR(1),
   PRIMARY KEY (Emp_ID),
   FOREIGN KEY (Mgr_ID)
   REFERENCES EMPLOYEE (Emp_ID));
**Unary Association Class to Table: CONTAINS**

**Table:**

<table>
<thead>
<tr>
<th>Contains Item_ Number</th>
<th>Contained Item_ Number</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>61384</td>
<td>M128</td>
<td>2</td>
</tr>
<tr>
<td>61384</td>
<td>A261</td>
<td>1</td>
</tr>
<tr>
<td>62807</td>
<td>A261</td>
<td>2</td>
</tr>
<tr>
<td>62807</td>
<td>R149</td>
<td>1</td>
</tr>
<tr>
<td>62009</td>
<td>R149</td>
<td>2</td>
</tr>
</tbody>
</table>

**Item_ Number**

<table>
<thead>
<tr>
<th>Item_ Number</th>
<th>Name</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>A261</td>
<td>Nail</td>
<td>0.99</td>
</tr>
<tr>
<td>61384</td>
<td>AAA</td>
<td>21.11</td>
</tr>
<tr>
<td>M128</td>
<td>Screw</td>
<td>2.99</td>
</tr>
<tr>
<td>62807</td>
<td>BBB</td>
<td>200.11</td>
</tr>
<tr>
<td>62009</td>
<td>CCC</td>
<td>99.00</td>
</tr>
</tbody>
</table>

**Figure 9.13b** Two Unary Relations — Bill-of-Materials Structure (M:N)
Transforming Unary M:N Relationships into Relations

- Relationship **CONTAINS**:

  ITEM Contains ITEM

1. Create Table for Relationship **CONTAINS**

2. Add PK of each side of Tables (Containing_Item_Number, Contained_Item_Number) as Foreign Keys

3. Make composite of both attributes as Primary Key of the Table **CONTAINS**:

   CREATE TABLE **CONTAINS** (  
   Containing_Item_Number CHAR(10),  
   Contained_Item_Number CHAR(10),  
   Quantity Integer,  
   PRIMARY KEY  
   (Containing_Item_Number,  
    Contained_Item_Number),  
   FOREIGN KEY  
   (Containing_Item_Number)  
   REFERENCES  
   ITEM(Item_Number),  
   FOREIGN KEY  
   (Contained_Item_Number)  
   REFERENCES  
   ITEM(Item_Number));

- Relationship **CONTAINS**:

  ITEM Contains ITEM
Weak Entities

- 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.
Review: Weak Entities

- 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 entities

- Weak entity set and identifying relationship set are translated into a single table --- it has a (1,1) cardinality constraint.

```sql
CREATE TABLE Dep_Policy (  
dname CHAR(20),  
age INTEGER,  
cost REAL,  
parent_ssn CHAR(9) NOT NULL,  
PRIMARY KEY (dname, parent_ssn),  
FOREIGN KEY (parent_ssn) REFERENCES Employees,  
ON DELETE CASCADE)
```

- When an owner entity is deleted all owned entity should also be deleted.
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 (  
dname CHAR(20),  
age INTEGER,  
cost REAL,  
ssn CHAR(11) NOT NULL,  
PRIMARY KEY (pname, ssn),  
FOREIGN KEY (ssn) REFERENCES Employees,  
ON DELETE CASCADE)
```
Translating Class Hierarchies

- **Example:**
  - Employees could have two subclasses:
    1. Hourly Employees (Hourly_Emps)
       - Characterized by: hourly wages and hours worked
    2. Contract Employees (Contract_Emps)
       1. Characterized by: contract id.

![Diagram showing class hierarchy](image-url)
Translating Class Hierarchies

- Two approaches
  - Three tables: 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*);
    - We must delete Hourly_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*.
    - Each employee must be in one of these two subclasses.
Primary Key Constraints

• A set of fields is a key for a relation if:
  1. No two distinct tuples can have same values in all key fields, and
  2. This is not true for any subset of the key. – Key is minimal.
     - However, 2 does not hold (so false) for superkey – which is not minimal.
     - If there’s more than one keys for a relation, one of the keys is chosen (by DBA) to be the primary key.

• E.g., customer_id is a key for Customer. (What about name?) The set \{customer_id, name\} could be a superkey.

Primary key can not have null value
Domain Constraint

- The value of each Attribute $A$ with Domain Type $D(A_i)$ must be a atomic value from the domain type $D(A_i)$. 
Definitions of Keys and Attributes Participating in Keys

• A superkey of a relation schema \( R = \{A_1, A_2, \ldots, A_n\} \) is a set of attributes \( S \), subset-of \( R \), with the property that no two tuples \( t_1 \) and \( t_2 \) in any legal relation state \( r \) of \( R \) will have \( t_1[S] = t_2[S] \). That is, for any given two tuples \( t_1, t_2 \) in data (extensions) of Relation schema \( R \), \( t_1[S] \) is not identical to \( t_2[S] \).

• A key \( K \) is a superkey with the additional property that removal of any attribute from \( K \) will cause \( K \) not to be a superkey any more; Key is minimal.
Definitions of Keys and Attributes Participating in Keys

- If a relation schema has more than one key, each is called a **candidate** key.
- One of the candidate keys is *arbitrarily* designated to be the **primary key**, and the others are called **secondary keys**.
- A **Prime attribute** must be a member of any (candidate) key.
- A **Nonprime attribute** is not a prime attribute—that is, it is not a member of any (candidate) key.
Foreign Keys, Referential Integrity

- **Foreign key**: Set of fields in one relation that is used to `refer` to a tuple in another relation. (Must correspond to primary key of the second relation.) Like a `logical pointer`.

- E.g. `customer_id` in `Order` is a foreign key referring to `Customer`:

  ```
  Order (order_number, order_date, promised_date, customer_id)
  ```
Foreign Keys, Referential Integrity

• If all foreign key constraints are enforced, referential integrity is achieved; all foreign key values should refer to existing values, i.e., no dangling references.

• Can you name a data model w/o referential integrity?
  - Links in HTML!
Enforcing Referential Integrity

- Consider Students\(\text{\texttt{sid, name, gpa}}\) and Enrolled \(\text{\texttt{rid, semester, sid}}\);
- \texttt{sid} in Enrolled is a foreign key that references Students.
- What should be done if an Enrolled tuple with a non-existent student id is inserted? Reject it!
- What should be done if a Students tuple is deleted?
  - Also delete all Enrolled tuples that refer to it.
  - Disallow deletion of a Students tuple that is referred to.
  - Set \texttt{sid} in Enrolled tuples that refer to it to a \textit{default \texttt{sid}}.
  - (In SQL, also: Set \texttt{sid} in Enrolled tuples that refer to it to a special value \texttt{null}, denoting `unknown` or `inapplicable`).
- Similar if primary key of Students tuple is updated.
Logical DB Design: ER to Relational

- Entity sets to tables.

CREATE TABLE Employees
(ssn CHAR(11),
  name CHAR(20),
  lot INTEGER,
  PRIMARY KEY (ssn))
Review: Key Constraints

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

Translation to relational model?

1-to-1  1-to Many  Many-to-1  Many-to-Many
Transforming 1:N, M:N Relationships with Key Constraints

ER Diagram:
Translating ER Diagrams with Key Constraints

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

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

```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
)
```
Transforming Realntionship to Tables

Example E-R diagram:
Relationship Sets to Tables

- In translating a relationship Works_In (M-N) to a relation, attributes of the relation must include:
  - Keys for each participating entity set (as foreign keys).
  - This set of attributes forms a superkey for the relation.
  - All descriptive attributes.

```sql
CREATE TABLE Works_In(
  ssn  CHAR(1),
  did INTEGER,
  since DATE,
  PRIMARY KEY (ssn, did),
  FOREIGN KEY (ssn)
    REFERENCES Employees,
  FOREIGN KEY (did)
    REFERENCES Departments)
```
**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)
```
Review: Binary vs. Ternary Relationships

• If each policy is owned by just 1 employee:
  - Key constraint on Policies would mean policy can only cover 1 dependent!

• What are the additional constraints in the 2nd diagram?

Bad design

Better design
Binary vs. Ternary Relationships
(Contd.)

- The key constraints allow us to combine Purchaser with Policies and Beneficiary with Dependents.
- Participation constraints lead to NOT NULL constraints.
- What if Policies is a weak entity set?

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

**CREATE TABLE Dependents (**
  dname CHAR(20),
  age INTEGER,
  policyid INTEGER,
  PRIMARY KEY (dname, policyid).
  FOREIGN KEY (policyid) REFERENCES Policies, 
  ON DELETE CASCADE);
An Example

CREATE TABLE Student (  
  ID NUMBER,  
  Fname VARCHAR2(20),  
  Lname VARCHAR2(20),  
);
Constraints in Create Table

• Adding constraints to a table enables the database system to enforce data integrity.

• Different types of constraints:
  * Not Null
  * Unique
  * Foreign Key
  * Default Values
  * Primary Key
  * Check Condition
Not Null Constraint

CREATE TABLE Student (  
  ID NUMBER,  
  Fname VARCHAR2(20) NOT NULL,  
  Lname VARCHAR2(20) NOT NULL,  
);
Primary Key Constraint

CREATE TABLE Student (
   ID       NUMBER PRIMARY KEY,
   Fname    VARCHAR2(20) NOT NULL,
   Lname    VARCHAR2(20) NOT NULL,
);

• Primary Key implies: * NOT NULL * UNIQUE.
• There can only be one primary key.
Primary Key Constraint
(Syntax 2)

CREATE TABLE Students (  
    ID NUMBER,  
    Fname VARCHAR2(20) NOT NULL,  
    Lname VARCHAR2(20) NOT NULL,  
    PRIMARY KEY(ID)  
);  

Needed when the primary key is made up of two or more attributes (fields)
Foreign Key Constraint

CREATE TABLE Studies(
    Course    NUMBER,
    Student   NUMBER,
    FOREIGN KEY (Student) REFERENCES Students(ID)
);

NOTE: ID must be unique (or primary key) in Students table