COP-5725 PRACTICE EXERCISES

Chapter 2: Database Design

Chapter 3: Relational Model

M. Amanda Crick

Exercise 2.4

Problem

- A company database needs to store information about employees (identified by *ssn, with salary and phone as attributes), departments (identified by dno,* with *dname and budget as attributes), and children of employees (with name and age* as attributes).

Exercise 2.4

Problem

- Employees *work in departments; each department is managed by an* employee; a child must be identified uniquely by *name when the parent (who is an* employee; assume that only one parent works for the company) is known. We are not interested in information about a child once the parent leaves the company.
- Draw an ER diagram that captures this information.

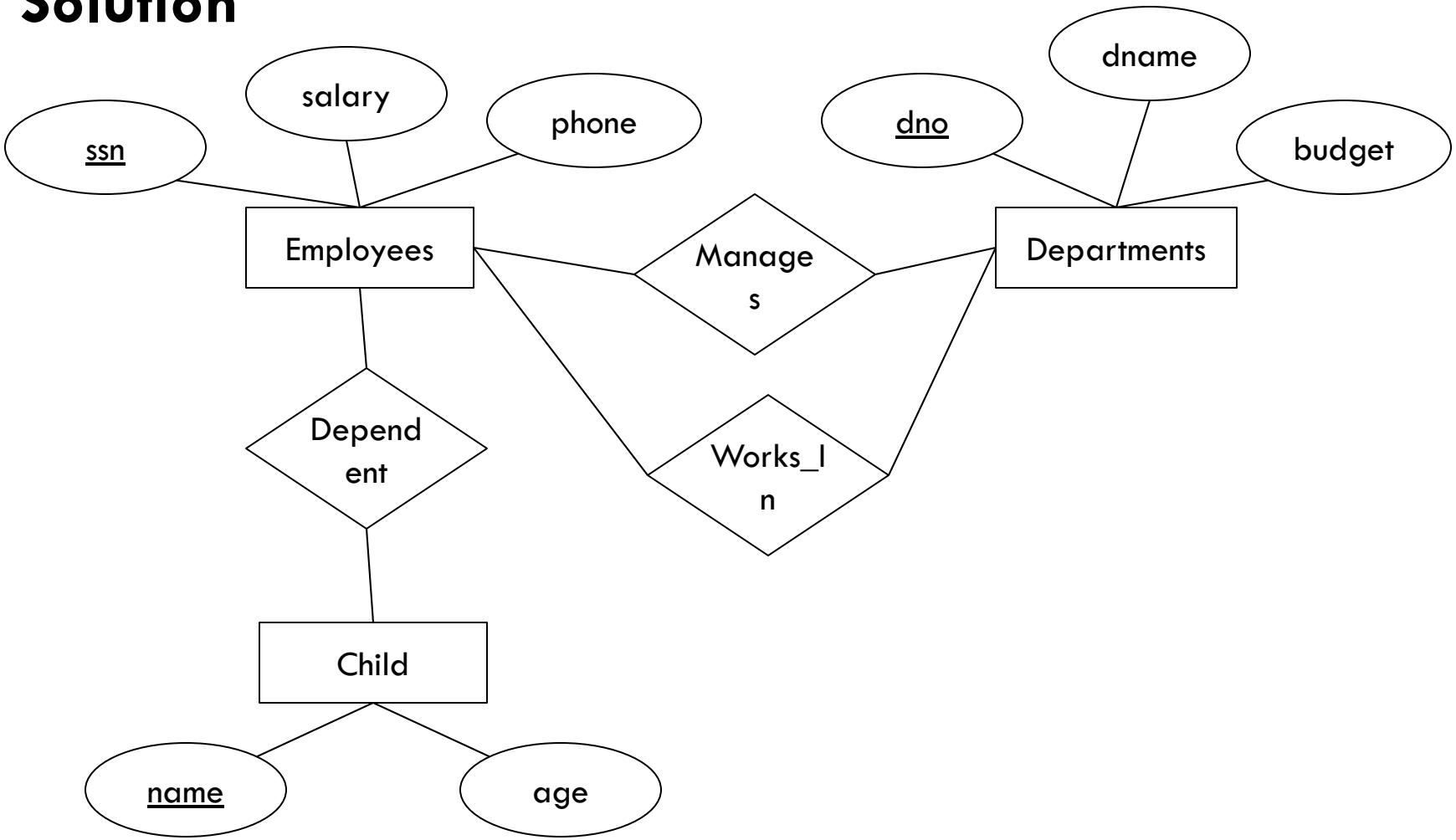
Exercise 2.4

Solution

- First, we shall design the entities and relationships.
  - "Employees *work in departments*…"
  - "…*each department is managed by an* employee…"
  - "…a child must be identified uniquely by *name when the parent (who is an* employee; assume that only one parent works for the company) is known."

Exercise 2.4

Solution



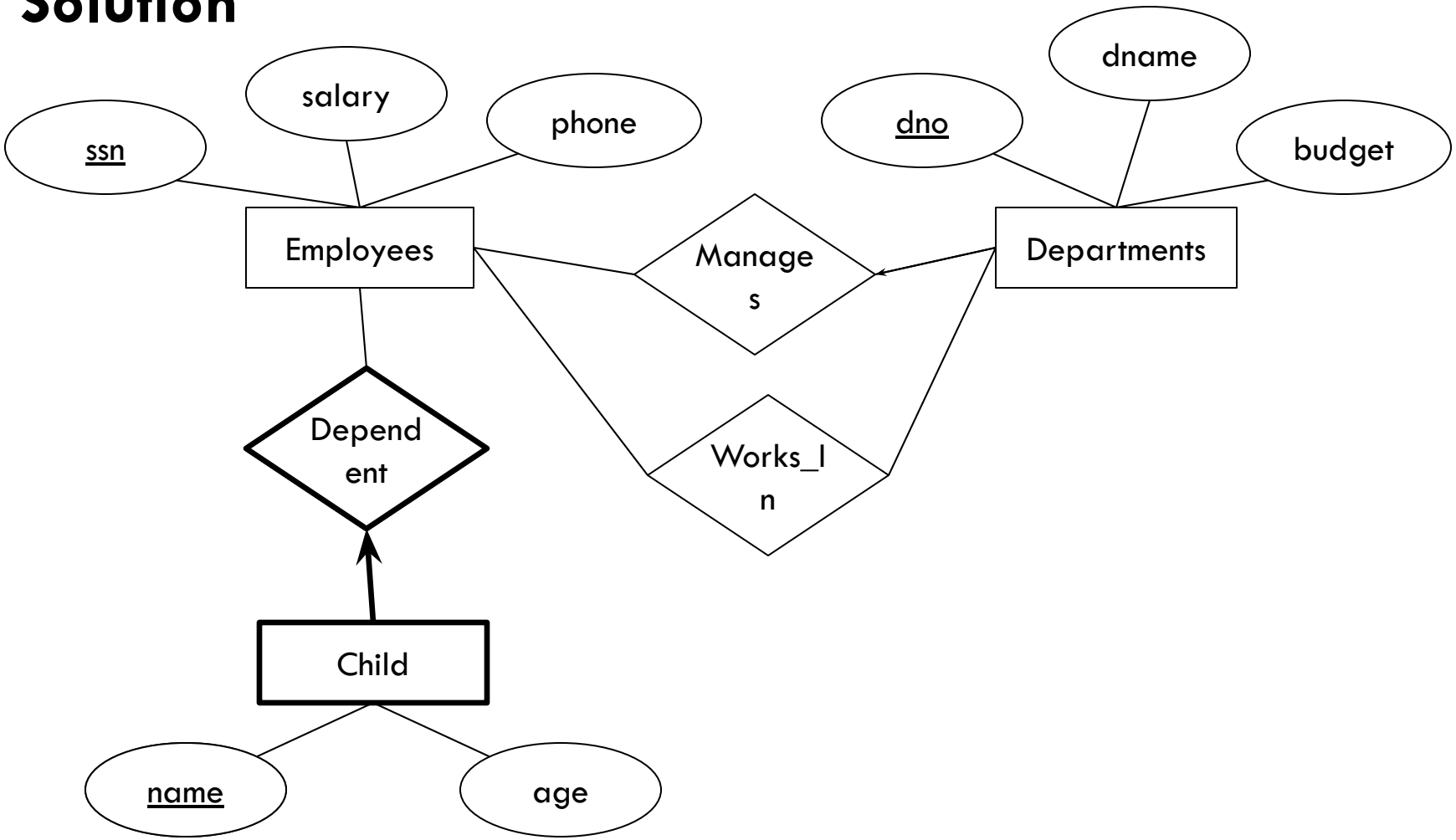
Exercise 2.4

Solution

- Now, we will design the constraints.
  - "…*each department is managed by* ***an*** employee…"
  - "…a child must be identified uniquely by *name when the parent (who is an* employee; assume that only one parent works for the company) is known. "
  - "We are not interested in information about a child once the parent leaves the company."

Exercise 2.4

Solution



Exercise 2.8

Problem

- Although you always wanted to be an artist, you ended up being an expert on databases because you love to cook data and you somehow confused *database* with *data baste*. *Your old love is still there, however, so you set up a database company,* ArtBase, that builds a product for art galleries. The core of this product is a database with a schema that captures all the information that galleries need to maintain.

Exercise 2.8

Problem

- Galleries keep information about artists, their names (which are unique), birthplaces, age,and style of art. For each piece of artwork, the artist, the year it was made, its unique title, its type of art (e.g., painting, lithograph, sculpture, photograph), and its price must be stored. Pieces of artwork are also classified into groups of various kinds, for example, portraits, still lifes, works by Picasso, or works of the 19th century; a given piece may belong to more than one group.

Exercise 2.8

Problem

- Each group is identified by a name (like those just given) that describes the group. Finally, galleries keep information about customers. For each customer, galleries keep that person's unique name, address, total amount of dollars spent in the gallery (very important!), and the artists and groups of art that the customer tends to like.
- Draw the ER diagram for the database.

Exercise 2.8

Solution

- Like before, we begin with the entities and relationships.
- "…artists, their names (which are unique), birthplaces, age, and style of art."
- "For each piece of artwork, the artist, the year it was made, its unique title, its type of art … and its price must be stored."

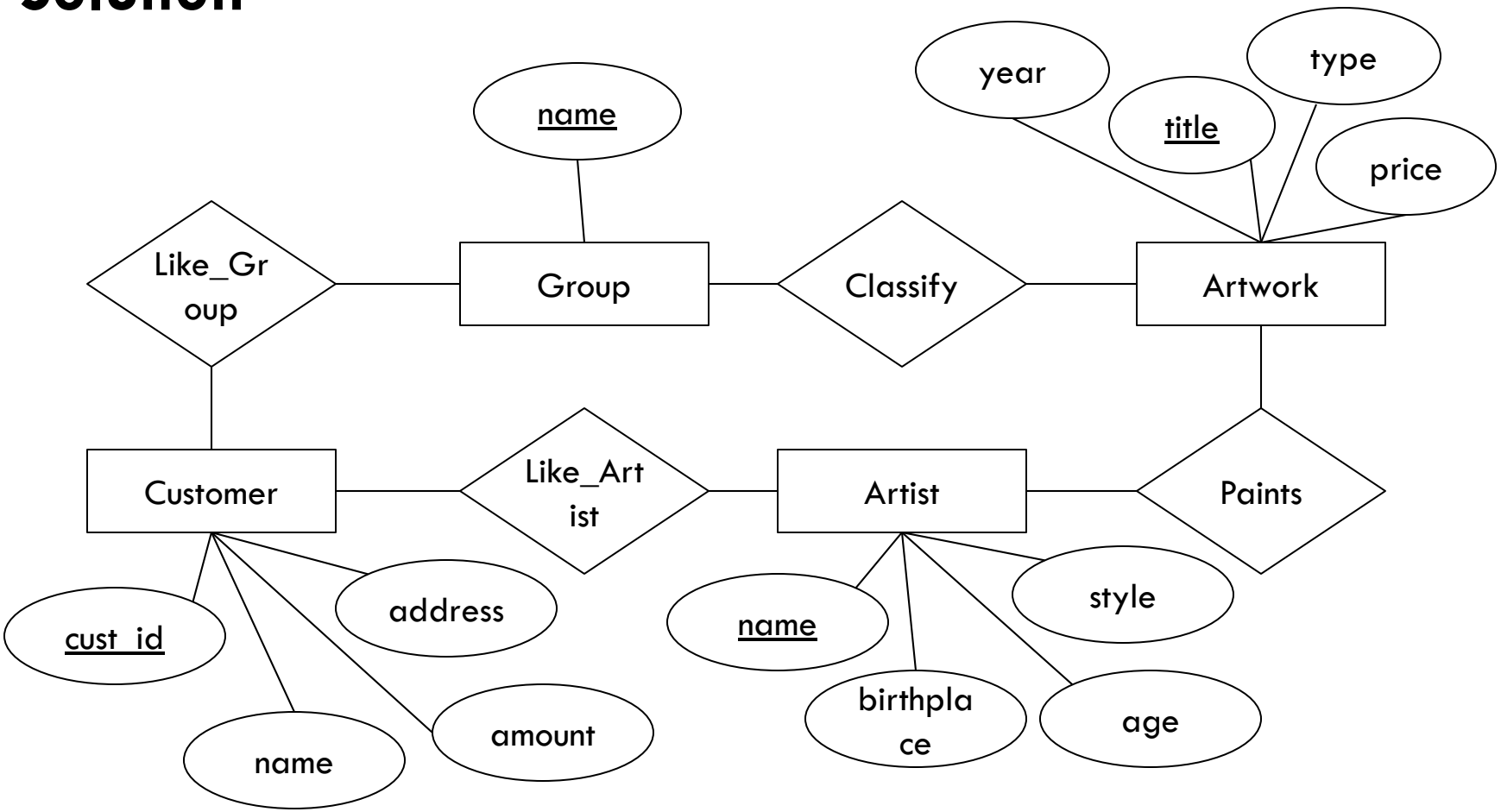
Exercise 2.8

Solution

- "Pieces of artwork are also classified into groups of various kinds, … Each group is identified by a name (like those just given) that describes the group. "
- For each customer, galleries keep that person's unique name, address, total amount of dollars spent in the gallery (very important!), and the artists and groups of art that the customer tends to like.

Exercise 2.8

Solution



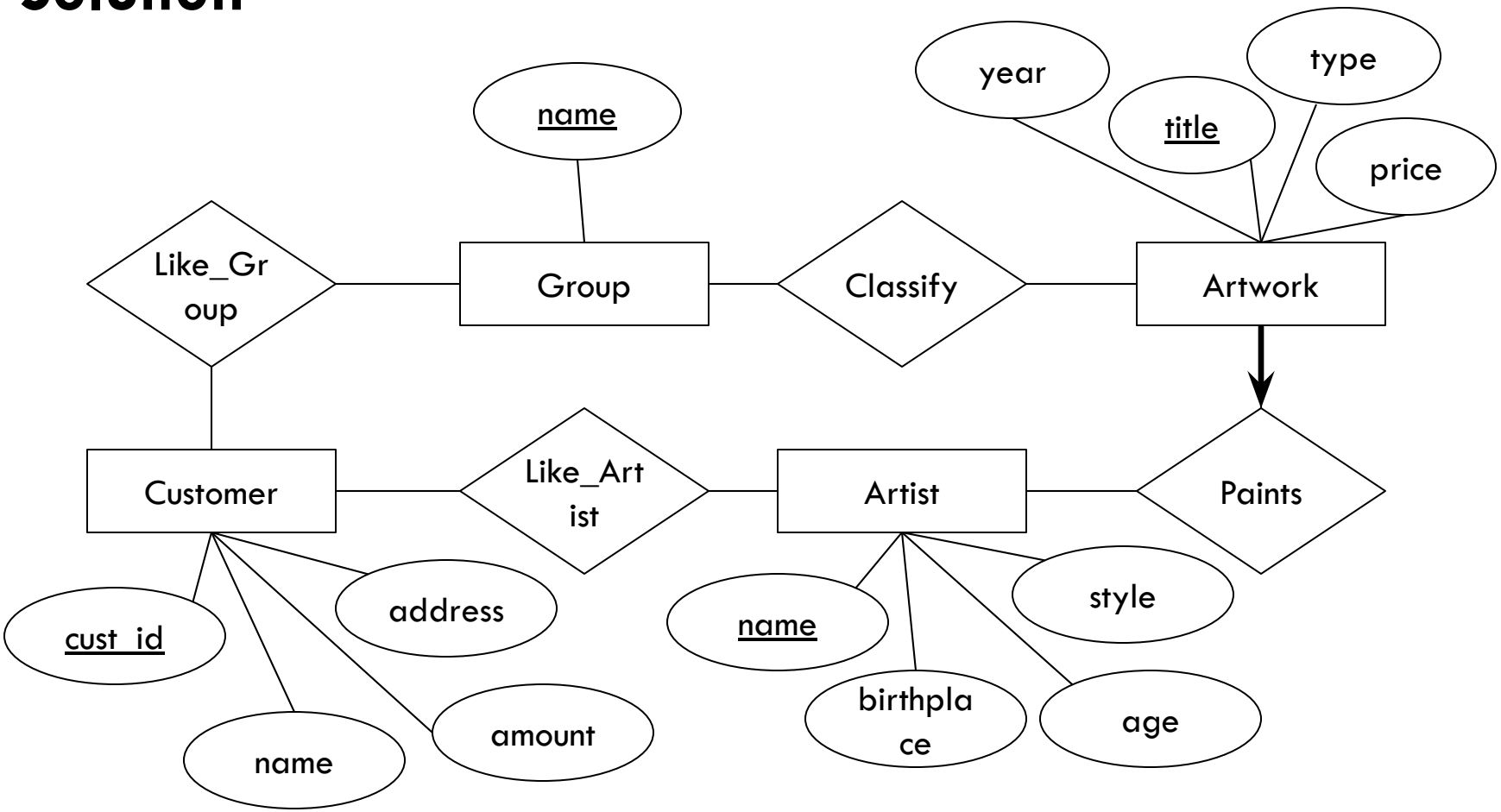
Exercise 2.8

Solution

- Now we look at constraints.
  - Although not explicitly mentioned in the problem, we assume that each piece of artwork had to be painted by an artist.
  - We also assume that each piece of artwork was created by exactly one artist.

Exercise 2.8

Solution



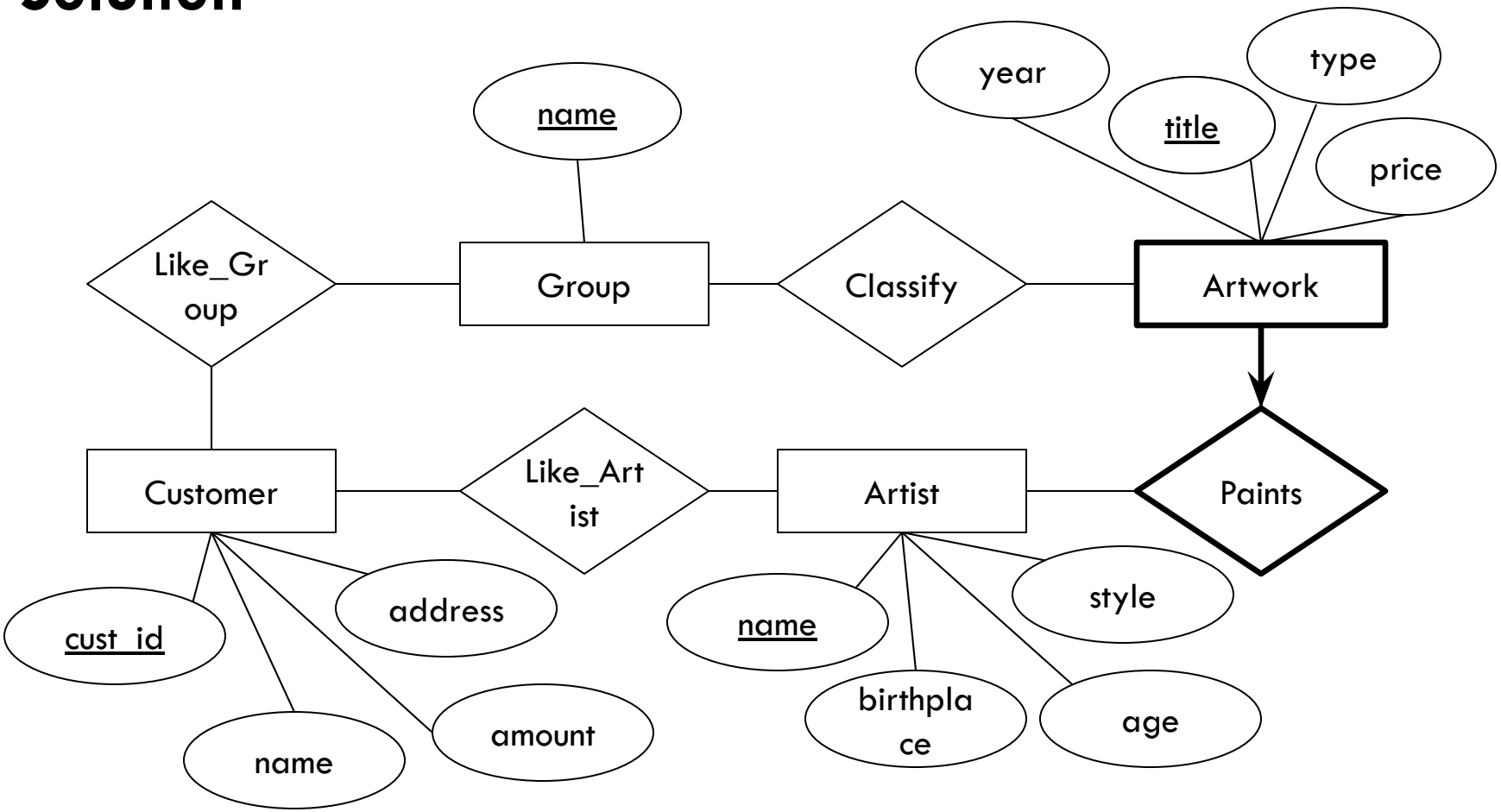
Exercise 2.8

Solution

- Suppose we had several piece of artwork with the same title, and we told them apart by artist?
- Example: "What is Love?" by Cheryl D, "What is Love?" by Joe Brown, etc.

Exercise 2.8

Solution



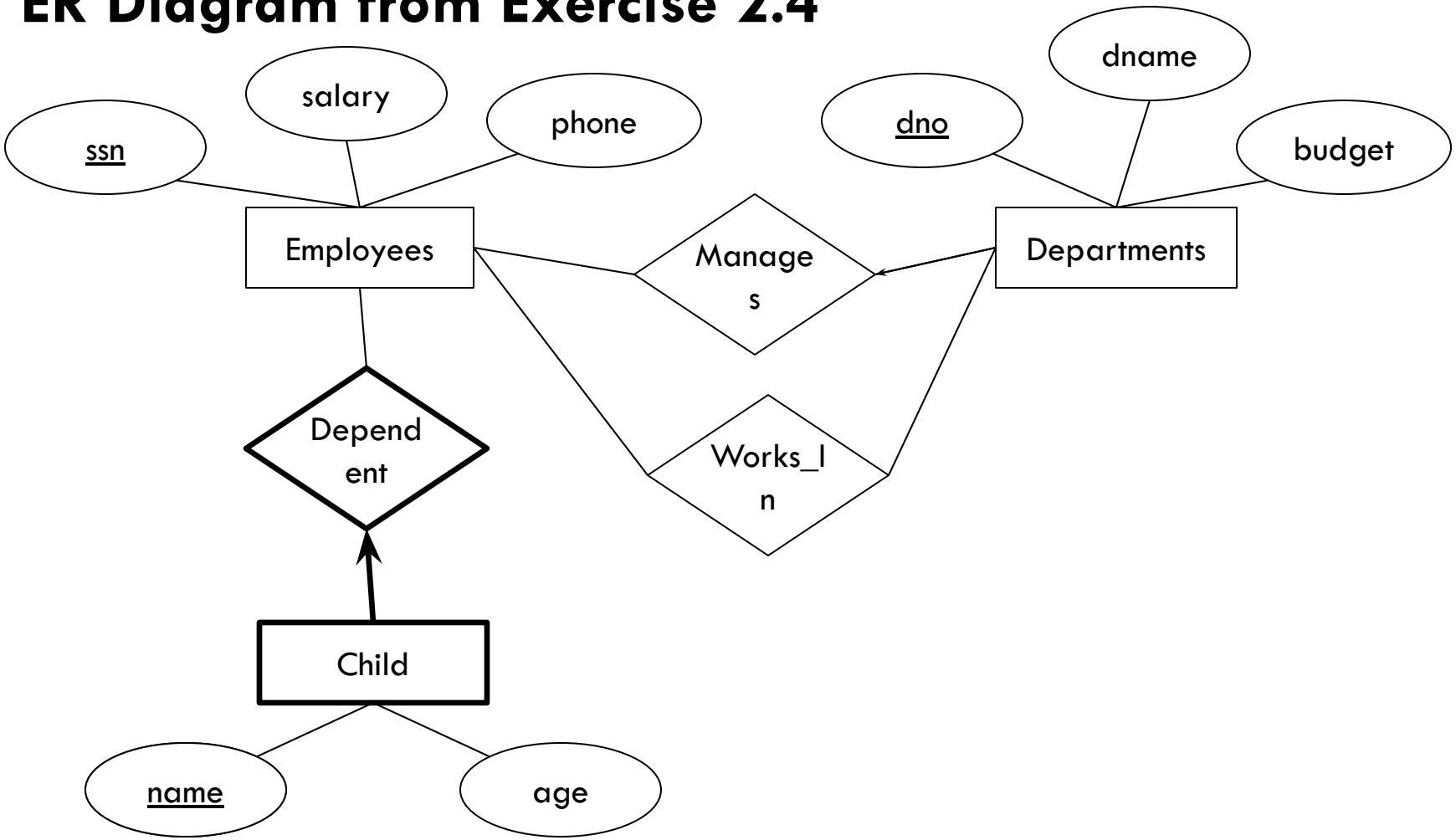
Exercise 3.14

Problem

- Consider the scenario from Exercise 2.4, where you designed an ER diagram for a company database. Write SQL statements to create the corresponding relations and capture as many of the constraints as possible. If you cannot capture some constraints, explain why.

Exercise 3.14

ER Diagram from Exercise 2.4



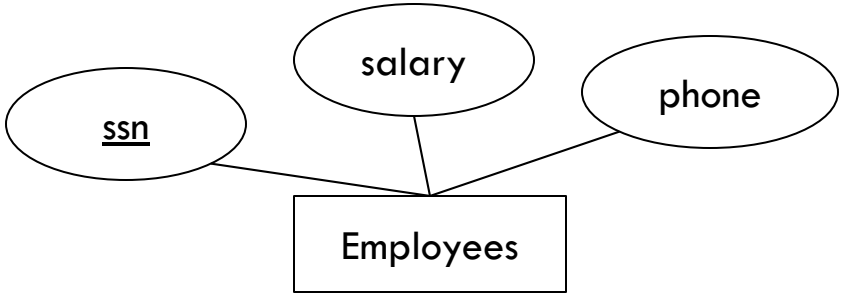
Exercise 3.14

Solution

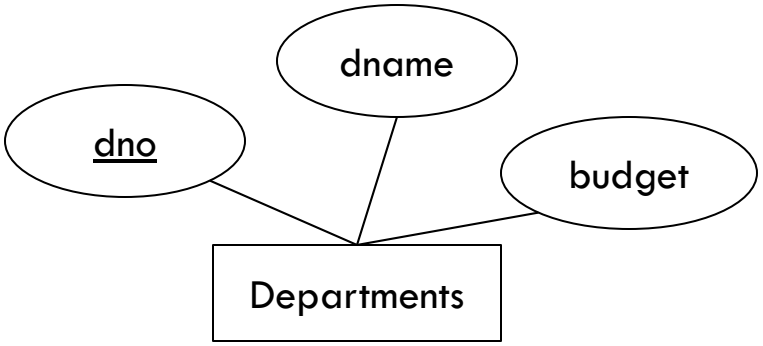
- First we begin with the entities “Employees” and “Departments.
- Translating these to SQL is straightforward.

Exercise 3.14

Solution




```
CREATE TABLE Employees(
        ssn CHAR(10),
                sal INTEGER,
                phone CHAR(13),
                PRIMARY KEY (ssn) )
```




```
CREATE TABLE Departments (
        dno INTEGER,
                budget INTEGER,
                dname CHAR(20),
                PRIMARY KEY (dno) )
```

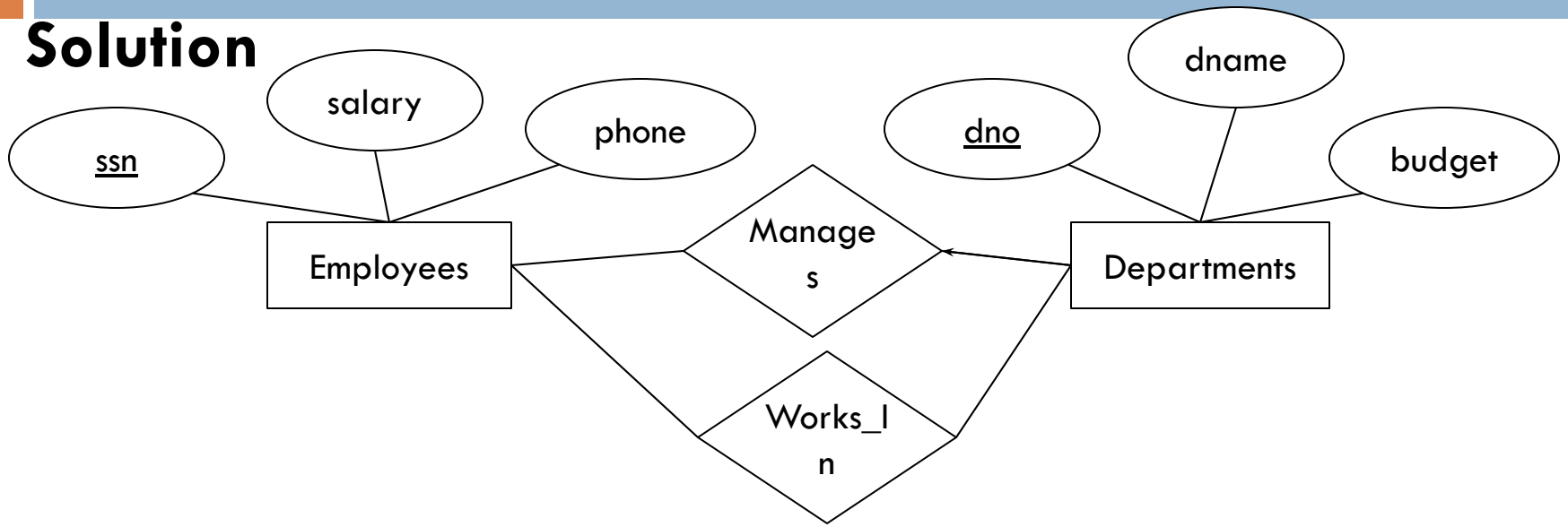
Exercise 3.14

Solution

- Next, we translate the relationships, Manages and Dependents.
- We translate each these to a table mapping one entity to another.
- We also use foreign constraints to make sure every row in the relationship tables refers only to rows that exist in the entity tables.

Exercise 3.14

Solution




```
CREATE TABLE Works_in(
        ssn CHAR(10),
        dno INTEGER,
        PRIMARY KEY (ssn, dno),
        FOREIGN KEY (ssn)
                REFERENCES  Employees,
        FOREIGN KEY (dno)
                REFERENCES  Departments)
```

```
CREATE TABLE Manages (
        ssn CHAR(10),
        dno INTEGER,
        PRIMARY KEY (dno),
        FOREIGN KEY (ssn)
                REFERENCES  Employees,
        FOREIGN KEY (dno)
                REFERENCES Departments)
```

Exercise 3.14

Solution

- Why did we make *dno* the primary key for Manages?
- Since each department can have at most one manager, each dno can appear at most once in the Manages table, making it a key for Manages.
- Note that if we had made (ssn, dno) the key for Manages, a department could have more than one Manager.

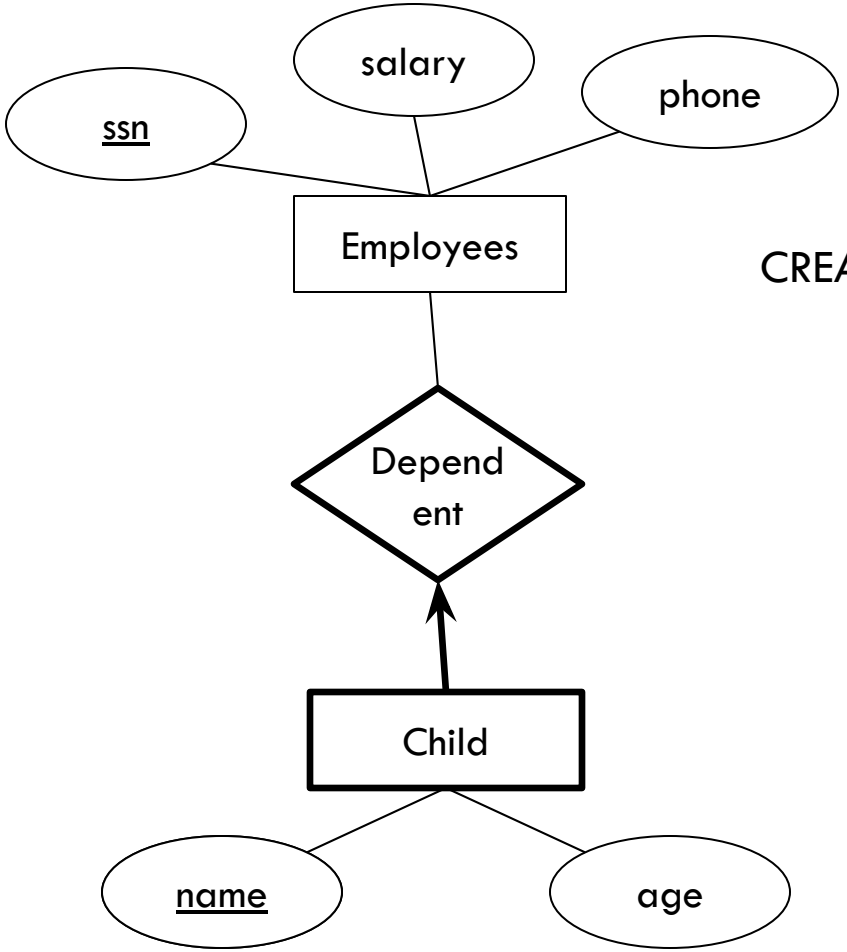
Exercise 3.14

Solution

- Finally, we translate the weak entity “Child” and its corresponding relationship “Dependent”

Exercise 3.14

Solution




```
CREATE TABLE Dependents(
        ssn CHAR(10),
        name CHAR(10),
        age INTEGER,
                PRIMARY KEY (ssn, name),
                FOREIGN KEY (ssn)
                        REFERENCES Employees,
                        ON DELETE CASCADE )
```

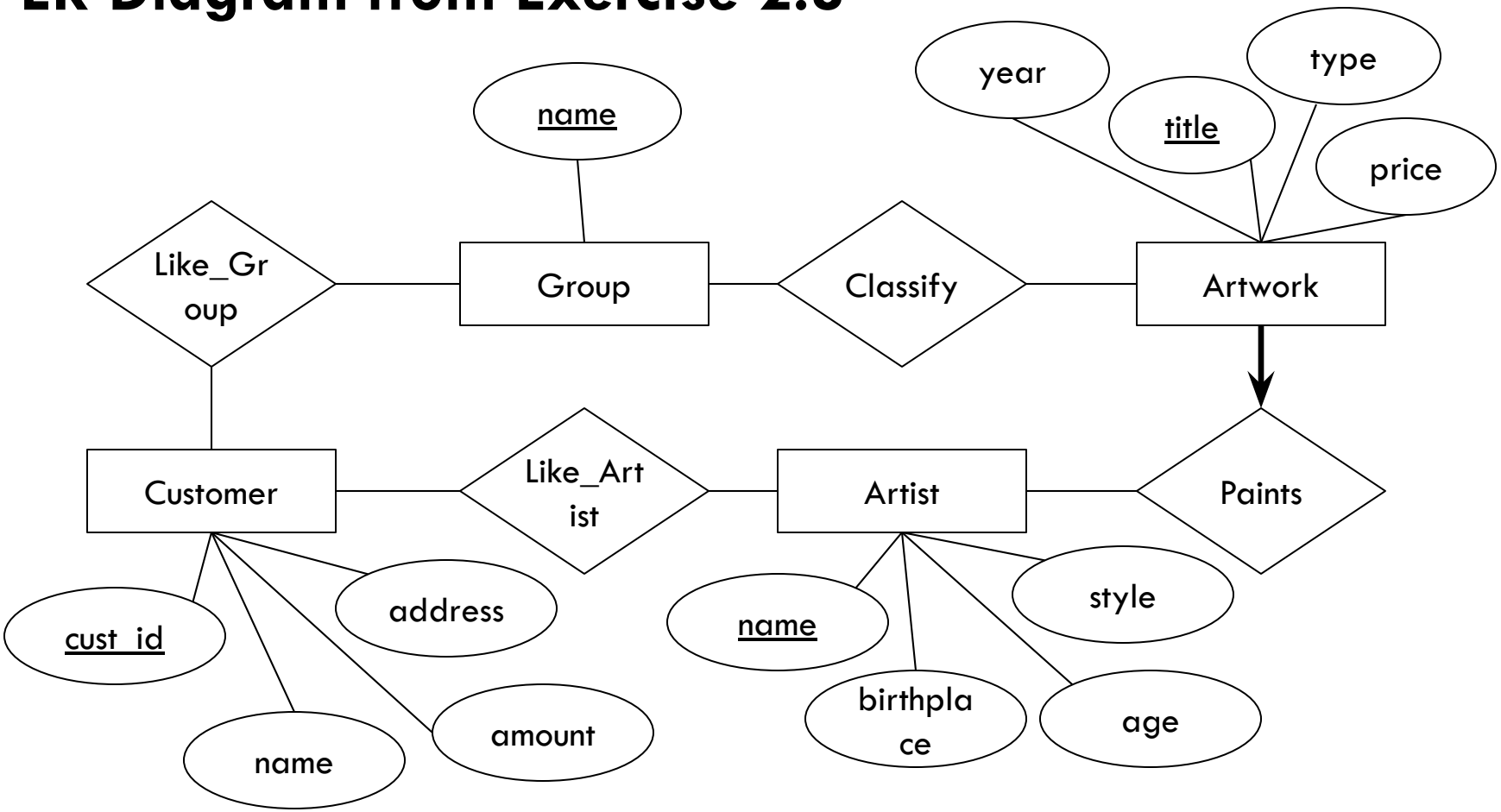
Exercise 3.18

Problem

- Write SQL statements to create the corresponding relations to the ER diagram you designed for Exercise 2.8. If your translation cannot capture any constraints in the ER diagram, explain why.

Exercise 3.18

ER Diagram from Exercise 2.8



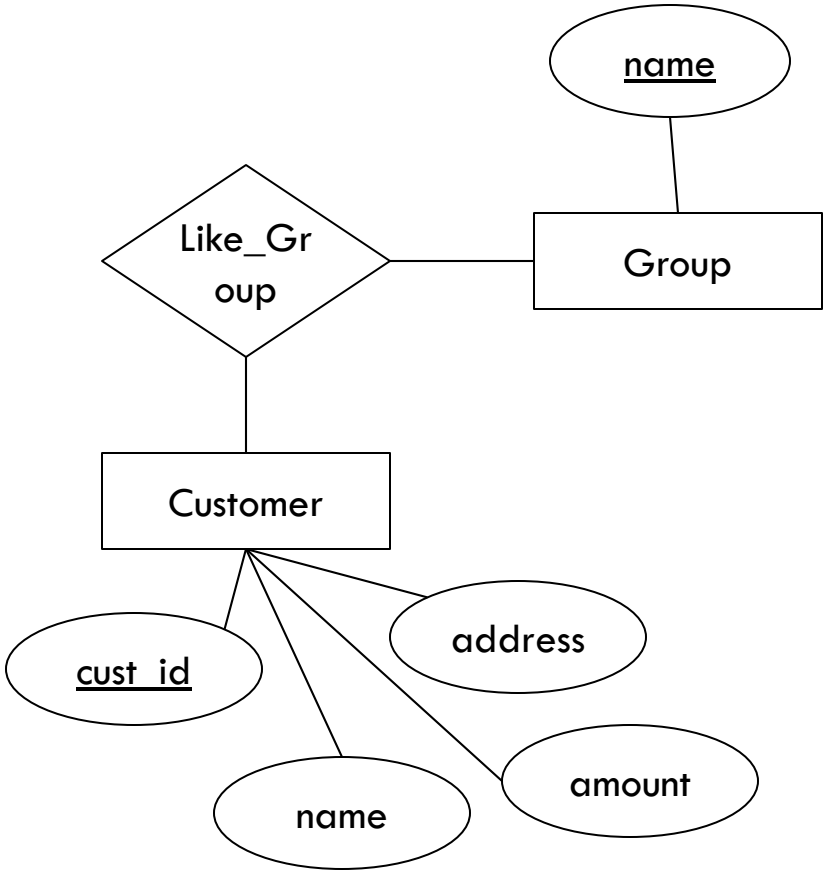
Exercise 3.18

Solution

- The entities are translated similarly to Exercise 3.4. Since these are fairly simple, we shall skip them.
- Now, we shall translate the relationships.

Exercise 3.18

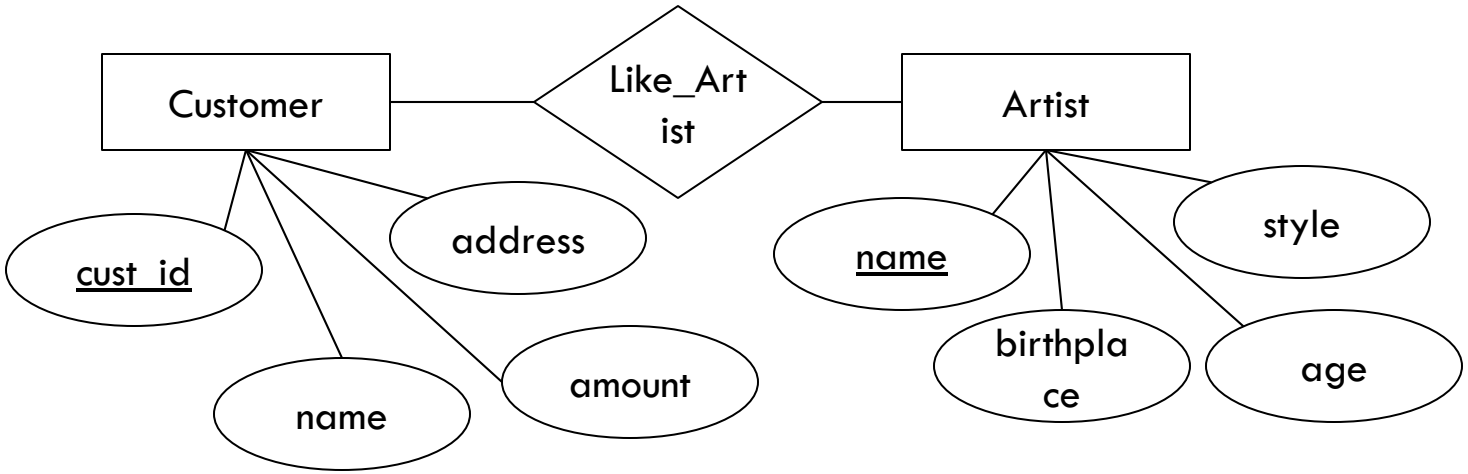
Solution




```
CREATE TABLE Like Group (
        name CHAR(20),
                cust name CHAR(20),
                PRIMARY KEY (name, cust_name),
                FOREIGN KEY (name)
                REFERENCES Group,
                FOREIGN KEY (cust name)
                REFERENCES Customer)
```

Exercise 3.18

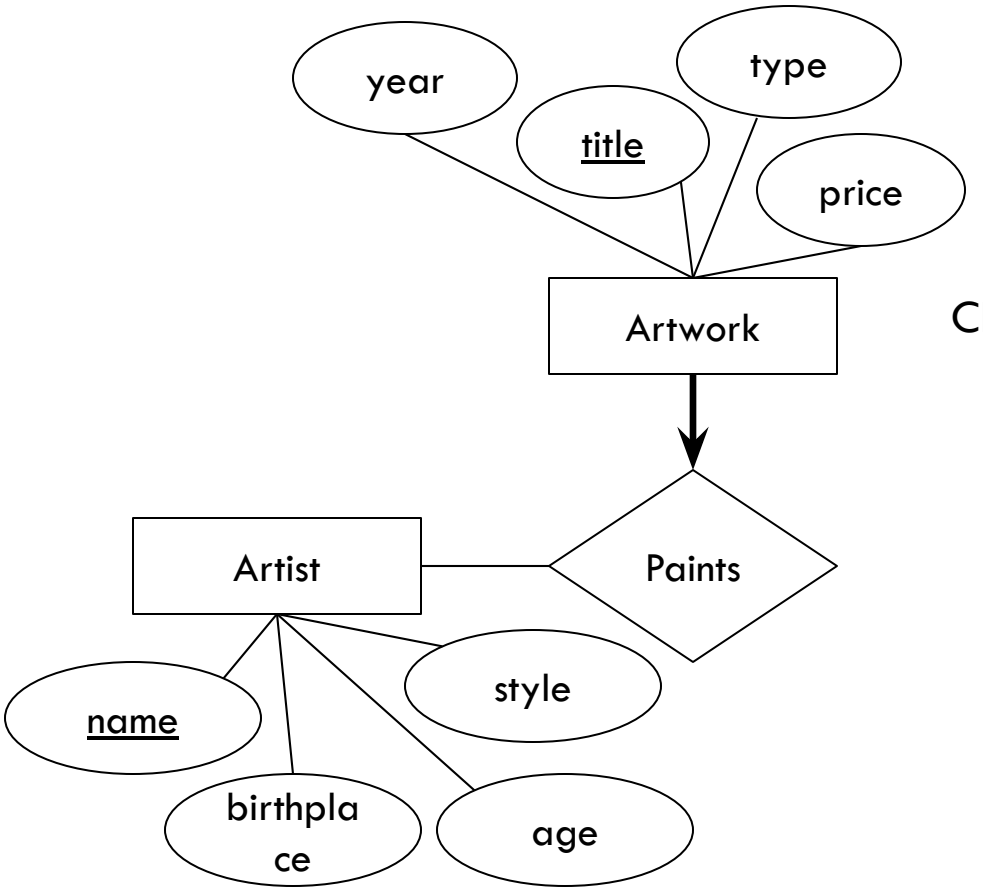
Solution




```
CREATE TABLE Like Artist (
        name CHAR(20),
        cust name CHAR(20),
        PRIMARY KEY (name, cust name),
        FOREIGN KEY (name) REFERENCES Artist,
        FOREIGN KEY (cust name) REFERENCES Customer)
```

Exercise 3.18

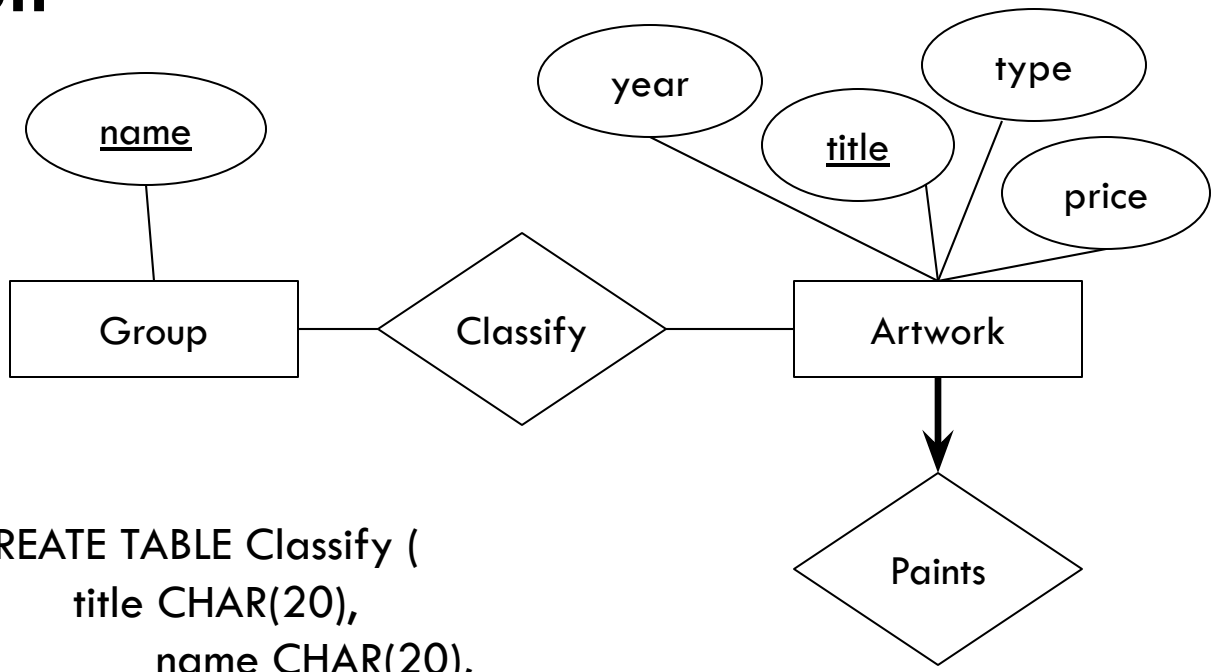
Solution




```
CREATE TABLE Artwork Paints(
        title CHAR(20),
        artist name CHAR(20),
        type CHAR(20),
        price INTEGER,
        year INTEGER,
        PRIMARY KEY (title),
        FOREIGN KEY (artist name)
                REFERENCES Artist)
```

Exercise 3.18

Solution




```
CREATE TABLE Classify (
        title CHAR(20),
              name CHAR(20),
              PRIMARY KEY (title, name),
              FOREIGN KEY (title) REFERENCES Artwork_Paints,
              FOREIGN KEY (name) REFERENCES Group )
```

Exercise 3.8

Problem

- Answer each of the following questions briefly. The questions are based on the following relational schema:
  - Emp(*eid: integer, ename: string, age: integer, salary: real*)
  - Works(*eid: integer, did: integer, pcttime: integer*)
  - Dept(*did: integer, dname: string, budget: real, managerid: integer*)

Exercise 3.8

Problem

Emp(*eid: integer*, *ename: string*, *age: integer*, *salary: real*)

Works(*eid: integer*, *did: integer*, *pcttime: integer*)

Dept(*did: integer*, *dname: string*, *budget: real*, *managerid: integer*

1. Give an example of a foreign key constraint that involves the Dept relation. What are the options for enforcing this constraint when a user attempts to delete a Dept tuple?

Exercise 3.8

Solution for (1)

An example of a foreign constraint that involves Dept is:

```
CREATE TABLE Works (
        eid INTEGER NOT NULL ,
        did INTEGER NOT NULL ,
        pcttime INTEGER,
        PRIMARY KEY (eid, did),
        UNIQUE (eid),
        FOREIGN KEY (did) REFERENCES Dept )
```

Exercise 3.8

Solution for (1)

Furthermore, when a user attempts to delete a tuple from Dept, we can

- also delete all Works tuples that refer to it.
- disallow the deletion of the Dept tuple if some Works tuple refers to it.
- for every Works tuple that refers to it, set the *did field to the did of some* (existing) 'default' department.
- for every Works tuple that refers to it, set the *did field to null*.

Exercise 3.8

Problem

Emp(*eid: integer*, *ename: string*, *age: integer*, *salary: real*)

Works(*eid: integer*, *did: integer*, *pcttime: integer*)

Dept(*did: integer*, *dname: string*, *budget: real*, *managerid: integer*

2. Write the SQL statements required to create the preceding relations, including appropriate versions of all primary and foreign key integrity constraints.

Exercise 3.8

Solution for (2)

Emp(*eid: integer*, *ename: string*, *age: integer*, *salary: real*)

Works(*eid: integer*, *did: integer*, *pcttime: integer*)

```
CREATE TABLE Emp (
        eid INTEGER,
        ename CHAR(10),
        age INTEGER,
        salary REAL,
        PRIMARY KEY (eid) )
```

```
CREATE TABLE Works (
        eid INTEGER,
        did INTEGER,
        pcttime INTEGER,
        PRIMARY KEY (eid, did),
        FOREIGN KEY (did) REFERENCES Dept,
        FOREIGN KEY (eid) REFERENCES Emp,
                ON DELETE CASCADE)
```

Exercise 3.8

Solution for (2)

Dept(*did: integer*, *dname: string, budget: real, managerid: integer*

```
CREATE TABLE Dept (
        did INTEGER,
        budget REAL,
        managerid INTEGER ,
        PRIMARY KEY (did),
        FOREIGN KEY (managerid) REFERENCES Emp,
                ON DELETE SET NULL)
```

Exercise 3.8

Problem

Emp(*eid: integer*, *ename: string*, *age: integer*, *salary: real*)

Works(*eid: integer*, *did: integer*, *pcttime: integer*)

Dept(*did: integer*, *dname: string*, *budget: real*, *managerid: integer*

3. Define the Dept relation in SQL so that every department is guaranteed to have a manager.

Example of a Solution for (3)

```
CREATE TABLE Dept (
        did INTEGER,
        budget REAL,
        managerid INTEGER NOT NULL ,
        PRIMARY KEY (did),
        FOREIGN KEY (managerid) REFERENCES Emp)
```

Exercise 3.8

Problem

Emp(*eid: integer*, *ename: string*, *age: integer*, *salary: real*)

Works(*eid: integer*, *did: integer*, *pcttime: integer*)

Dept(*did: integer*, *dname: string*, *budget: real*, *managerid: integer*

4. Write an SQL statement to add John Doe as an employee with *eid* = *101*, *age* = *32* and *salary* = *15, 000*.

Solution for (4)

```
INSERT
INTO Emp (eid, ename, age, salary)
VALUES (101, 'John Doe', 32,
15000)
```

Exercise 3.8

Problem

Emp(*eid: integer*, *ename: string*, *age: integer*, *salary: real*)

Works(*eid: integer*, *did: integer*, *pcttime: integer*)

Dept(*did: integer*, *dname: string*, *budget: real*, *managerid: integer*

5. Write an SQL statement to give every employee a 10 percent raise.

Solution for (5)

```
UPDATE Emp E
SET E.salary = E.salary * 1.10
```

Exercise 3.8

Problem

Emp(*eid: integer*, *ename: string, age: integer, salary: real)*

Works(*eid: integer, did: integer*, *pcttime: integer)*

Dept(*did: integer*, *dname: string, budget: real, managerid: integer*

6. Write an Write an SQL statement to delete the Toy department. Given the referential integrity constraints you chose for this schema, explain what happens when this statement is executed.

Exercise 3.8

Solution for (6)

```
DELETE
FROM Dept D
WHERE D.dname = 'Toy'
```

These are the example integrity constraints that affect Dept.

```
CREATE TABLE Works (
    ...
    FOREIGN KEY (did) REFERENCES Dept,
    ...)
```

Since the action to take on deletion was not specified, the database takes no action by default That is, it rejects the deletion.

Exercise 3.8

Solution for (6)

- What other actions can the system take on deleting a Dept tuple? What are the pros and cons of each action?
  - On delete set null
  - On delete set default
  - On delete cascade

This is the end of the lecture!

I hope you enjoyed it.