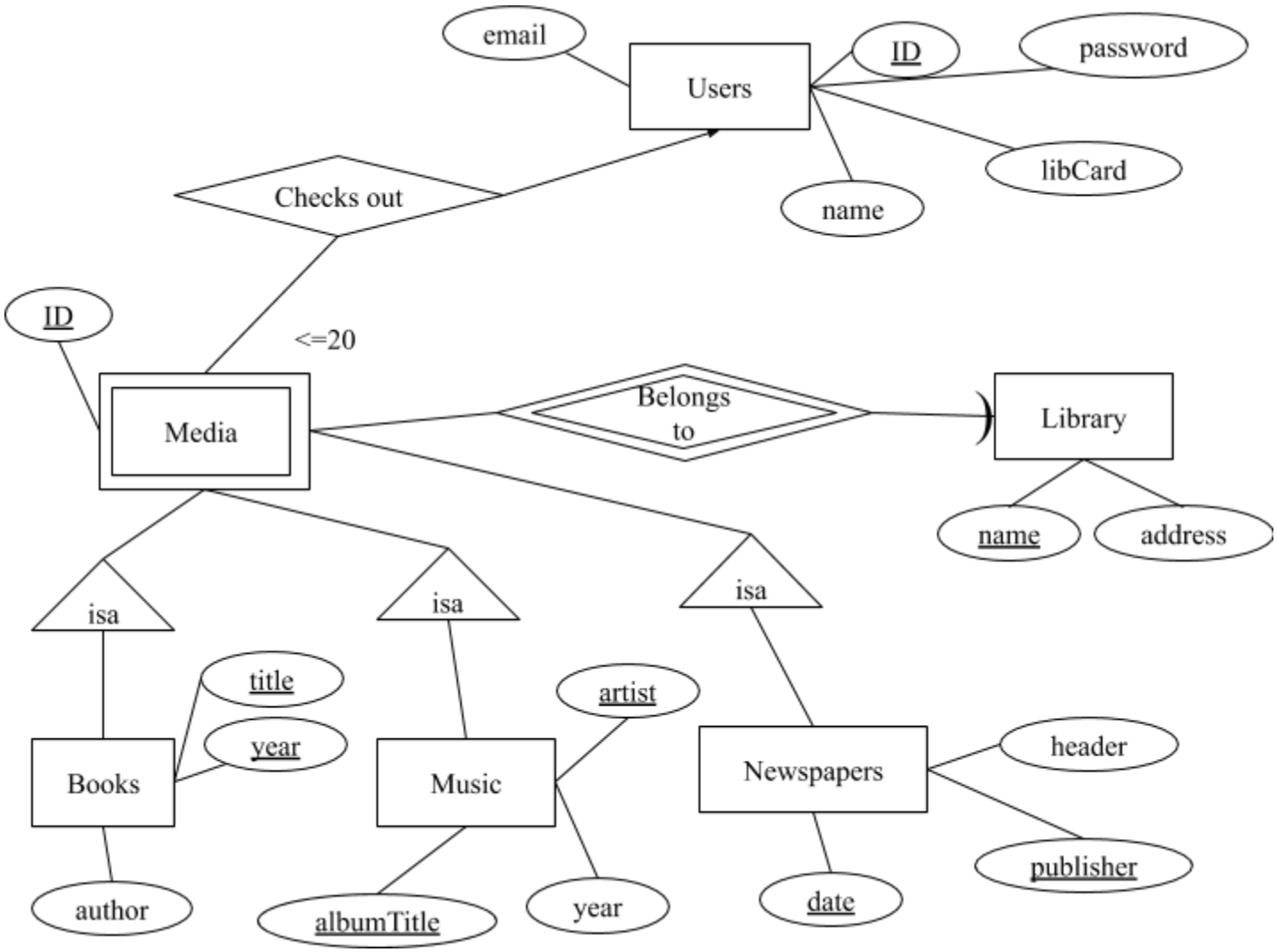
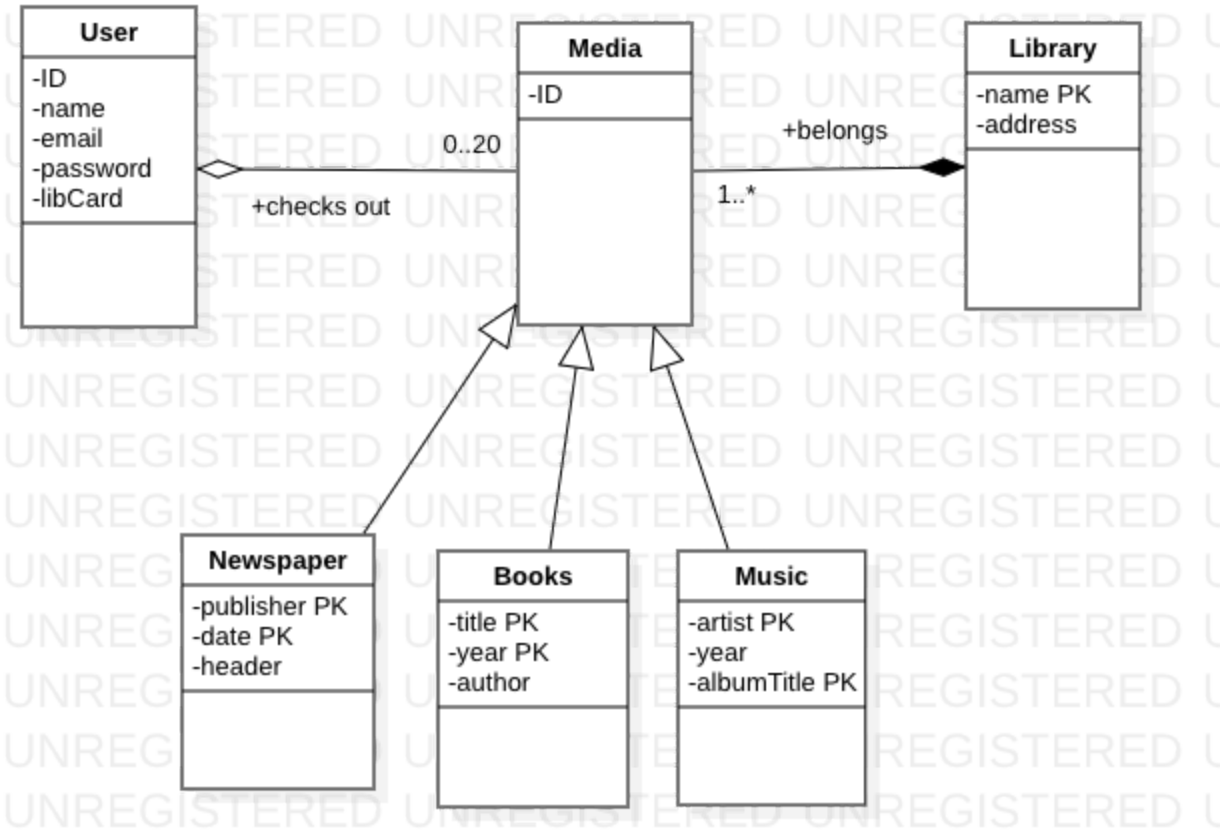
CS157A HW 4
SJSU Students
11/13/19

1. E/R diagram



UML diagram



2. Mapping given E/R diagram to relational schema:

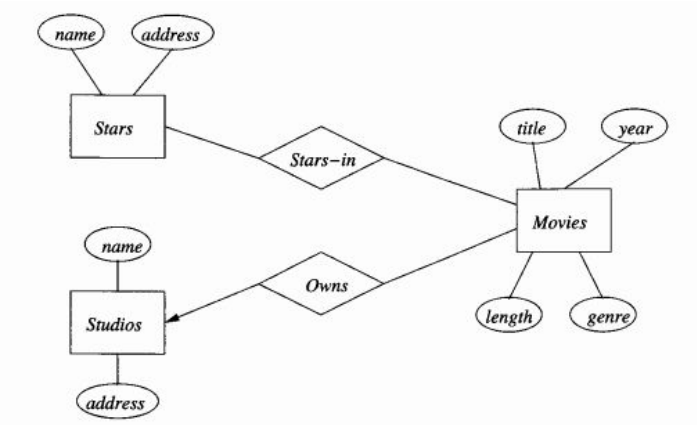


Figure 4.2: An entity-relationship diagram for the movie database

A many-one relationship exists from Movies to Studios. To map this to a relational schema, the Movies entity set creates a new Movies relation with the same name and attributes as the entity set. To represent the many-one relationship of Owns, the Movies relation also includes studioName, the foreign key reference to Studios.

Movies

title	year	length	genre	studioName

The Studios entity set maps to a Studios relation with the same name and address attributes as the entity set.

Studios

name	address

The Stars entity set maps to a Stars relation with the same name and address attributes as the entity set.

Stars

name	address

A many-many relationship exists between Movies and Stars. To map this to a relational schema, a Stars-in relation is created to represent the Stars-in relationship. The attributes of this relation are the primary keys of the entity sets participating in this relation, Star and Movies. These attributes are name from the Stars entity set and title, year from the Movies entity set.

Stars-in

title	year	name

* The title and year attributes are a foreign key references to the title and year attributes of Movies, while the name attribute is a foreign key reference to the name attribute of Stars.

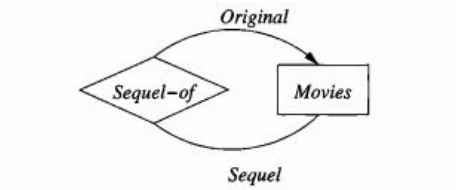


Figure 4.5: A relationship with roles

Any sequel can only be the sequel to one movie, so this is a one-one relationship. In this case, the foreign key approach is used. We create a relation from the Movies relationship with the same name. Then, we add an extra attribute that will reference a foreign key of the movie is a sequel to the current movie.

Movies

title	year	length	genre	sequel_title	sequel_year

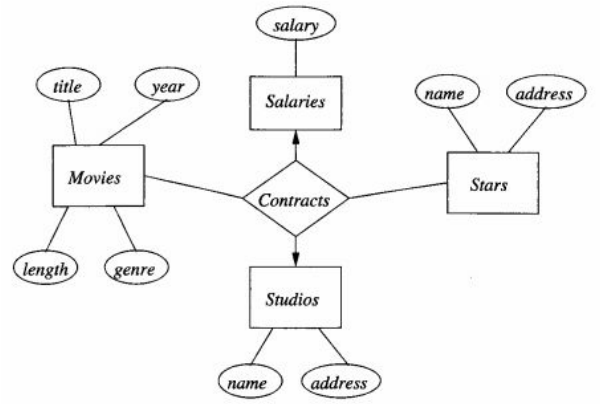


Figure 4.8: Moving the attribute to an entity set

The Movies entity set maps to a Movies relation with the same title, year, length, and genre attributes as the entity set.

Movies

title	year	length	genre

The Studios entity set maps to a Studios relation with the same name and address attributes as the entity set.

Studios

name	address

The Stars entity set maps to a Stars relation with the same name and address attributes as the entity set.

Stars

name	address

The Salaries entity set maps to a Salaries relation with the same salary attribute as the entity set.

Salaries

salary

To map a multiway (4-way) relationship, a new Contracts relation is created to represent the Contracts relationship. The attributes of this relation are the foreign keys that reference to the connected entity sets, Movies, Studios, Stars, and Salaries. These foreign keys are the primary keys of each entity set: title, year for Movies, name for Studios, name for Stars, and salary for Salaries.

Contracts

title	year	studioName	starName	salary

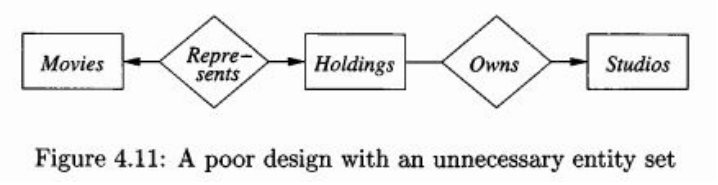


Figure 4.11: A poor design with an unnecessary entity set

A one-one relationship exists between Movies and Holdings. To map this to a relational schema, the Movies entity set creates a new Movies relation with the assumed attributes of title, year, length, and genre. Likewise, the Holdings entity set creates a new Holdings relation with the assumed attributes of size and address. A Represents relation is also created to show the one-one relationship between Movies and Holdings and carries the primary keys of Movies and Holdings (title, year and address, respectively) for cross-referencing.

Movies

title	year	length	genre

Holdings

size	address

Represents

title	year	address

* The title and year attributes are a foreign key references to the title and year attributes of Movies, while the address attribute is a foreign key reference to the address attributes of Holdings.

A many-one relationship exists between Holdings and Studios. To map this to a relational schema, the many-side (Holdings) carries a foreign key reference to the connecting entity set, Studios.

Studios

name	address

Holdings

size	address	studioName

*The studioName attribute is a foreign key reference to the name attribute of Studios.

ANSWER:

Movies

title	year	length	genre

Represents

title	year	address

Studios

name	address

Holdings

size	address	studioName

3. E/R diagram -> UML diagram

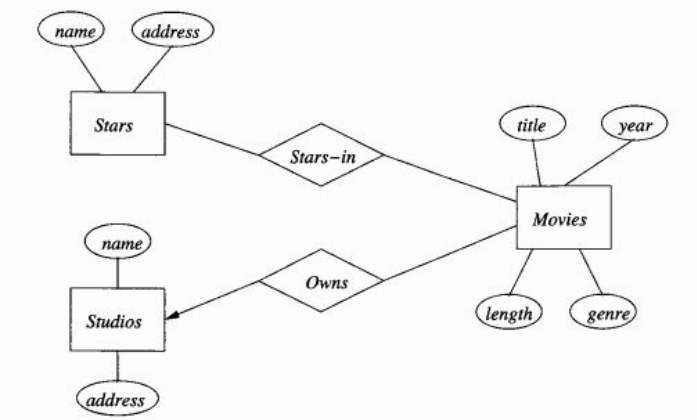
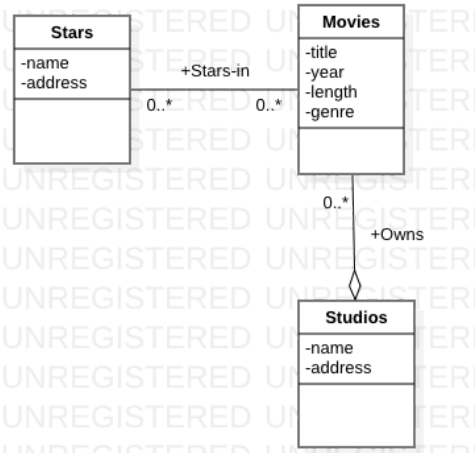


Figure 4.2: An entity-relationship diagram for the movie database



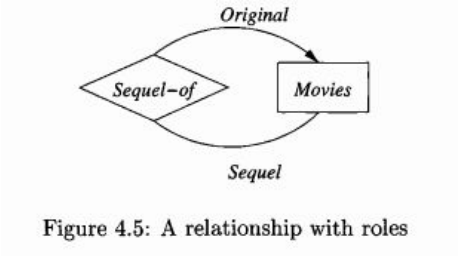


Figure 4.5: A relationship with roles

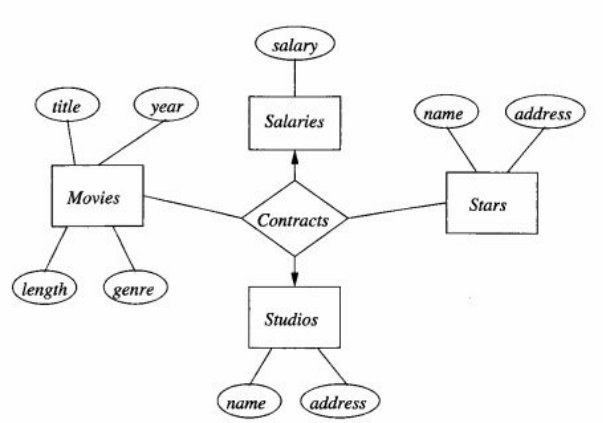
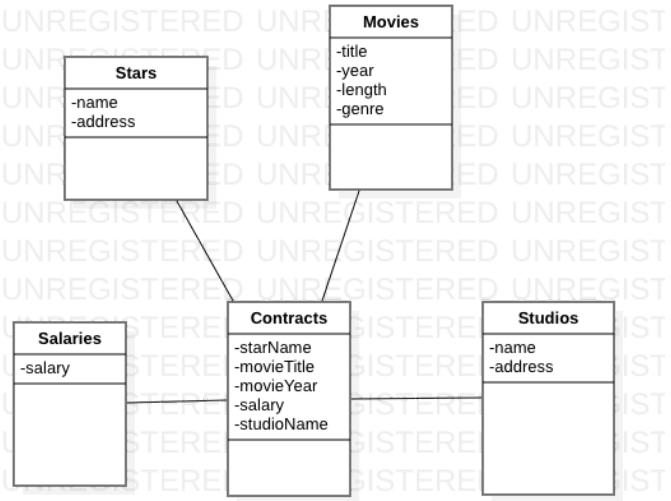


Figure 4.8: Moving the attribute to an entity set



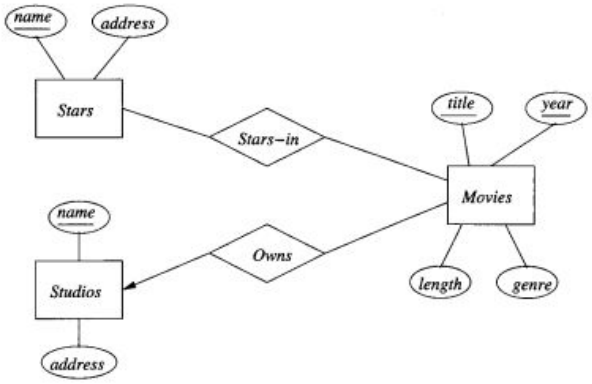


Figure 4.17: E/R diagram; keys are indicated by underlines

A relation is created for each class in the UML diagram, where the name and attributes of the class are the name and attributes of the relation.

Movies (title, year, length, genre)
Stars (name, address)
Studios (name, address)

For the Stars-in association, a Stars-in relation is created. The attributes of the Stars-in relation are the key attributes of the classes connected, which in this case are Stars and Movies with the key attributes of name and title and year, respectively.

Stars-in (name, title, year)
*The name attribute is a foreign key reference to the name attribute in Stars, while title and year attributes are the foreign key references to the title and year attributes in Movies.

For the aggregation relationship, the many-one relationship mapping of E/R diagrams to relational is followed. S denotes the many-side of the relationship, which in this case is Movies. Movies will then include a foreign key reference to Studios in its relational schema. Therefore, creating an Owns relation is unnecessary.

Movies (title, year, length, genre, name)
* The name attribute is a foreign key reference to the name attribute in Studios.

ANSWER:
Movies (title, year, length, genre, name)
Stars (name, address)

Studios (<u>name</u>, address)
Stars-in (<u>name</u>, <u>title</u>, <u>year</u>)

4. Exercise 5.2.1

R(A, B): {(0, 1), (2, 3), (0, 1), (2, 4), (3, 4)}
S(B, C): {(0, 1), (2, 4), (2, 5), (2, 7), (3, 4), (0, 2), (3, 4)}

a. $\pi_{A+B,\ A\hat{}2,\ B\hat{}2}$ (R)

A + B	A^2	B^2
1	0	1
5	4	9
1	0	1
6	4	16
7	9	16

d. $\tau_{B,C}$ (S)

B	C
0	1
0	2
2	4
2	5
2	7
3	4
3	4

h. $\gamma_{B,\ AVG(C)}$(S)

B	AVG (C)
0	1.5

2	5.3
3	4

m. R ⋉ S

A	B	C
0	1	NULL
0	1	NULL
2	3	4
2	3	4
2	4	NULL
3	4	NULL
NULL	0	1
NULL	2	4
NULL	2	5
NULL	0	2
NULL	2	7

5. Exercise 5.3.1
   Write the queries of Exercise 2.4.1 in Datalog.

   Product (maker, model, type)
   PC (model, speed, ram, hd, price)
   Laptop (model, speed, ram, hd, screen, price)
   Printer (model, color, type, price)

   a. What PC models have a speed of at least 3.00?
      FastPC (speed) ← PC (model, speed, ram, hd, price), speed ≥ 3.00

   b. Which manufacturers make laptops with a hard disk of at least 100GB?
      Manufacturers (maker) ← Product (maker, model, type), Laptop (model, speed, ram, hd, screen, price), hd ≥ 100

6. Exercise 6.1.3

Product (maker, model, type)
PC (model, speed, ram, hd, price)
Laptop (model, speed, ram, hd, screen, price)
Printer (model, color, type, price)

a. Find the model number, speed, and hard-disk size for all PC's whose price is under $1000.

```
SELECT model, speed, hd FROM PC WHERE price < 1000;
```

b. Do the same as (a), but rename the speed column gigahertz and the hd column gigabytes.

```
SELECT model, speed AS gigahertz, hd AS gigabytes FROM PC
WHERE price < 1000;
```

c. Find the manufacturers of printers.

```
SELECT maker AS manufacturer FROM Product, Printer WHERE
Product.model = Printer.model;
```

d. Find the model number, memory size, and screen size for laptops costing more than $1500.

```
SELECT model, ram, screen FROM Laptop WHERE price > 1500;
```

e. Find all the tuples in the Printer relation for color printers. Remember that color is a boolean-valued attribute.

```
SELECT * FROM Printer WHERE color = true;
```

f. Find the model number and hard-disk size for those PC's that have a speed of 3.2 and a price less than $2000.

```
SELECT model, hd FROM PC WHERE speed = 3.2 AND price < 2000;
```

Commands to create tables:

```
CREATE DATABASE ComputerStore;
USE ComputerStore;
CREATE TABLE Product(maker VARCHAR(20), model INT, type VARCHAR(20));
CREATE TABLE PC(model INT, speed FLOAT, ram FLOAT, hd FLOAT, price INT);
CREATE TABLE Laptop(model INT, speed FLOAT, ram FLOAT, hd FLOAT, screen
FLOAT, price INT);
```

```
CREATE TABLE Printer(model INT, color BOOLEAN, type VARCHAR(20), price
INT);

INSERT INTO Product VALUES("Samsung", 1, "small laptop");
INSERT INTO Product VALUES("Apple", 10, "small laptop");
INSERT INTO Product VALUES("Dell", 100, "small laptop");

INSERT INTO PC VALUES(1, 1.5, 4.5, 5.5, 2000);
INSERT INTO PC VALUES(10, 3.2, 3.5, 4.5, 3000);
INSERT INTO PC VALUES(100, 3.2, 2.5, 3.5, 900);

INSERT INTO Laptop VALUES(1, 1.5, 4.5, 5.5, 20.5, 2000);
INSERT INTO Laptop VALUES(10, 2.5, 3.5, 4.5, 22.5, 3000);
INSERT INTO Laptop VALUES(100, 1.0, 2.5, 3.5, 15.0, 1000);

INSERT INTO Printer VALUES(1, true, "a", 100);
INSERT INTO Printer VALUES(10, false, "b", 200);
INSERT INTO Printer VALUES(100, true, "c", 90);
```

Results from query

```
mysql> SELECT model, speed, hd FROM PC WHERE price < 1000;
+-------+-------+------+
| model | speed | hd   |
+-------+-------+------+
|   100 |   3.2 |  3.5 |
+-------+-------+------+
1 row in set (0.01 sec)

mysql> SELECT model, speed AS gigahertz, hd AS gigabytes FROM PC WHERE price <
1000;
+-------+-----------+-----------+
| model | gigahertz | gigabytes |
+-------+-----------+-----------+
|   100 |       3.2 |       3.5 |
+-------+-----------+-----------+
1 row in set (0.00 sec)

mysql> SELECT maker AS manufacturer FROM Product, Printer WHERE Product.model =
Printer.model;
```

```
+--------------+
| manufacturer |
+--------------+
| Samsung      |
| Apple        |
| Dell         |
+--------------+
4 rows in set (0.00 sec)

mysql> SELECT model, ram, screen FROM Laptop WHERE price > 1500;
+-------+------+--------+
| model | ram  | screen |
+-------+------+--------+
|     1 |  4.5 |   20.5 |
|    10 |  3.5 |   22.5 |
+-------+------+--------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM Printer WHERE color = true;
+-------+-------+------+-------+
| model | color | type | price |
+-------+-------+------+-------+
|     1 |     1 | a    |   100 |
|   100 |     1 | c    |    90 |
+-------+-------+------+-------+
2 rows in set (0.00 sec)

mysql> SELECT model, hd FROM PC WHERE speed like 3.2 AND price < 2000;
+-------+------+
| model | hd   |
+-------+------+
|   100 |  3.5 |
+-------+------+
1 row in set (0.00 sec)
```