Exercise 2.2.1a

For relation Accounts, the attributes are:

acctNo, type, balance

For relation Customers, the attributes are:

firstName, lastName, idNo, account

Exercise 2.2.1b

For relation Accounts, the tuples are:

(12345, savings, 12000),
(23456, checking, 1000),
(34567, savings, 25)

For relation Customers, the tuples are:

(Robbie, Banks, 901-222, 12345),
(Lena, Hand, 805-333, 12345),
(Lena, Hand, 805-333, 23456)

Exercise 2.2.1c

For relation Accounts and the first tuple, the components are:

123456 → acctNo
savings → type
12000 → balance

For relation Customers and the first tuple, the components are:

Robbie → firstName
Banks → lastName
901-222 → idNo
12345 → account

Exercise 2.2.1d

For relation Accounts, a relation schema is:

Accounts(acctNo, type, balance)

For relation Customers, a relation schema is:

Customers(firstName, lastName, idNo, account)

Exercise 2.2.1e

An example database schema is:

Accounts (
acctNo,
type,
balance
)
Customers (
firstName,
lastName,
idNo,
account
)

Exercise 2.2.1f

A suitable domain for each attribute:

acctNo → Integer
type → String
balance → Integer
firstName → String
lastName → String
idNo → String (because there is a hyphen we cannot use Integer)
account → Integer

Exercise 2.2.1g

Another equivalent way to present the Account relation:

acctNo	balance	type
34567	25	savings
23456	1000	checking
12345	12000	savings

Another equivalent way to present the Customers relation:

idNo	firstName	lastName	account
805-333	Lena	Hand	23456
805-333	Lena	Hand	12345
901-222	Robbie	Banks	12345

Exercise 2.2.2

Examples of attributes that are created for primarily serving as keys in a relation:

Universal Product Code (UPC) used widely in United States and Canada to track products in stores.

Serial Numbers on a wide variety of products to allow the manufacturer to individually track each product.

Vehicle Identification Numbers (VIN), a unique serial number used by the automotive industry to identify vehicles.

Exercise 2.2.3a

We can order the three tuples in any of 3! = 6 ways. Also, the columns can be ordered in any of 3! = 6 ways. Thus, the number of presentations is 6*6 = 36.

Exercise 2.2.3b

We can order the three tuples in any of 5! = 120 ways. Also, the columns can be ordered in any of 4! = 24 ways. Thus, the number of presentations is 120*24 = 2880

Exercise 2.2.3c

We can order the three tuples in any of m! ways. Also, the columns can be ordered in any of n! ways. Thus, the number of presentations is n!m!

Exercise 2.3.1a

```
CREATE TABLE Product (
        maker  CHAR(30),
        model  CHAR(10) PRIMARY KEY,
        type CHAR(15)
);
```

Exercise 2.3.1b

```
CREATE TABLE PC (
```

```
        model CHAR(30),
        speed DECIMAL(4,2),
        ram INTEGER,
        hd INTEGER,
        price DECIMAL(7,2)
);
```

Exercise 2.3.1c

```
CREATE TABLE Laptop (
        model CHAR(30),
        speed DECIMAL(4,2),
        ram INTEGER,
        hd INTEGER,
        screen DECIMAL(3,1),
        price DECIMAL(7,2)
);
```

Exercise 2.3.1d

```
CREATE TABLE Printer (
        model CHAR(30),
        color BOOLEAN,
        type CHAR (10),
        price DECIMAL(7,2)
);
```

Exercise 2.3.1e

```
ALTER TABLE Printer DROP color;
```

Exercise 2.3.1f

```
ALTER TABLE Laptop ADD od CHAR (10) DEFAULT ‘none’;
```

Exercise 2.3.2a

```
CREATE TABLE Classes (
        class CHAR(20),
        type CHAR(5),
        country CHAR(20),
        numGuns INTEGER,
        bore DECIMAL(3,1),
        displacement INTEGER
);
```

Exercise 2.3.2b

```
CREATE TABLE Ships (
        name CHAR(30),
        class CHAR(20),
        launched INTEGER
);
```

Exercise 2.3.2c

```
CREATE TABLE Battles (
        name CHAR(30),
        date DATE
);
```

Exercise 2.3.2d

```
CREATE TABLE Outcomes (
        ship CHAR(30),
        battle CHAR(30),
        result CHAR(10)
);
```

Exercise 2.3.2e

```
ALTER TABLE Classes DROP bore;
```

Exercise 2.3.2f

```
ALTER TABLE Ships ADD yard CHAR(30);
```

Exercise 2.4.1a

R1 := $\sigma_{speed \geq 3.00}$ (PC)
R2 := π_{model}(R1)

model
1005
1006
1013

Exercise 2.4.1b

R1 := $\sigma_{hd \geq 100}$ (Laptop)
R2 := Product $\bowtie$ (R1)
R3 := π_{maker} (R2)

maker
E
A
B
F
G

Exercise 2.4.1c

R1 := $\sigma_{maker=B}$ (Product ⋈ PC)
R2 := $\sigma_{maker=B}$ (Product ⋈ Laptop)
R3 := $\sigma_{maker=B}$ (Product ⋈ Printer)
R4 := $\pi_{model,price}$ (R1)
R5 := $\pi_{model,price}$ (R2)
R6: = $\pi_{model,price}$ (R3)
R7 := R4 ∪ R5 ∪ R6

model	price
1004	649
1005	630
1006	1049
2007	1429

Exercise 2.4.1d

R1 := $\sigma_{color = true\ AND\ type = laser}$ (Printer)
R2 := π_{model} (R1)

model
3003
3007

Exercise 2.4.1e

R1 := $\sigma_{type=laptop}$ (Product)
R2 := $\sigma_{type=PC}$(Product)
R3 := π_{maker}(R1)
R4 := π_{maker}(R2)
R5 := R3 – R4

maker
F
G

Exercise 2.4.1f

$R1 := \rho_{PC1}(PC)$
$R2 := \rho_{PC2}(PC)$
$R3 := R1 \bowtie_{(PC1.hd = PC2.hd \text{ AND } PC1.model <> PC2.model)} R2$
$R4 := \pi_{hd}(R3)$

hd
250
80
160

Exercise 2.4.1g

$R1 := \rho_{PC1}(PC)$
$R2 := \rho_{PC2}(PC)$
$R3 := R1 \bowtie_{(PC1.speed = PC2.speed \text{ AND } PC1.ram = PC2.ram \text{ AND } PC1.model < PC2.model)} R2$
$R4 := \pi_{PC1.model,PC2.model}(R3)$

PC1.model	PC2.model
1004	1012

Exercise 2.4.1h

$R1 := \pi_{model}(\sigma_{speed \geq 2.80}(PC)) \cup \pi_{model}(\sigma_{speed \geq 2.80}(Laptop))$
$R2 := \pi_{maker,model}(R1 \bowtie Product)$
$R3 := \rho_{R3(maker2,model2)}(R2)$
$R4 := R2 \bowtie_{(maker = maker2 \text{ AND } model <> model2)} R3$
$R5 := \pi_{maker}(R4)$

maker
B
E

Exercise 2.4.1i

$R1 := \pi_{model,speed}(PC)$
$R2 := \pi_{model,speed}(Laptop)$
$R3 := R1 \cup R2$
$R4 := \rho_{R4(model2,speed2)}(R3)$
$R5 := \pi_{model,speed}(R3 \bowtie_{(speed < speed2)} R4)$
$R6 := R3 - R5$
$R7 := \pi_{maker}(R6 \bowtie Product)$

maker
B

Exercise 2.4.1j

$R1 := \pi_{maker,speed}(Product \bowtie PC)$
$R2 := \rho_{R2(maker2,speed2)}(R1)$
$R3 := \rho_{R3(maker3,speed3)}(R1)$
$R4 := R1 \bowtie_{(maker = maker2 \text{ AND } speed <> speed2)} R2$
$R5 := R4 \bowtie_{(maker3 = maker \text{ AND } speed3 <> speed2 \text{ AND } speed3 <> speed)} R3$
$R6 := \pi_{maker}(R5)$

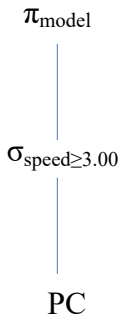
maker
A
D
E

Exercise 2.4.1k

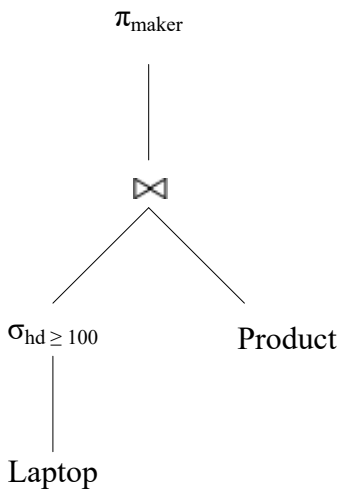
$R1 := \pi_{maker,model}(Product \bowtie PC)$
$R2 := \rho_{R2(maker2,model2)}(R1)$
$R3 := \rho_{R3(maker3,model3)}(R1)$
$R4 := \rho_{R4(maker4,model4)}(R1)$
$R5 := R1 \bowtie_{(maker = maker2 \text{ AND } model <> model2)} R2$
$R6 := R3 \bowtie_{(maker3 = maker \text{ AND } model3 <> model2 \text{ AND } model3 <> model)} R5$
$R7 := R4 \bowtie_{(maker4 = maker \text{ AND } (model4=model \text{ OR } model4=model2 \text{ OR } model4=model3))} R6$
$R8 := \pi_{maker}(R7)$

maker
A
B
D
E

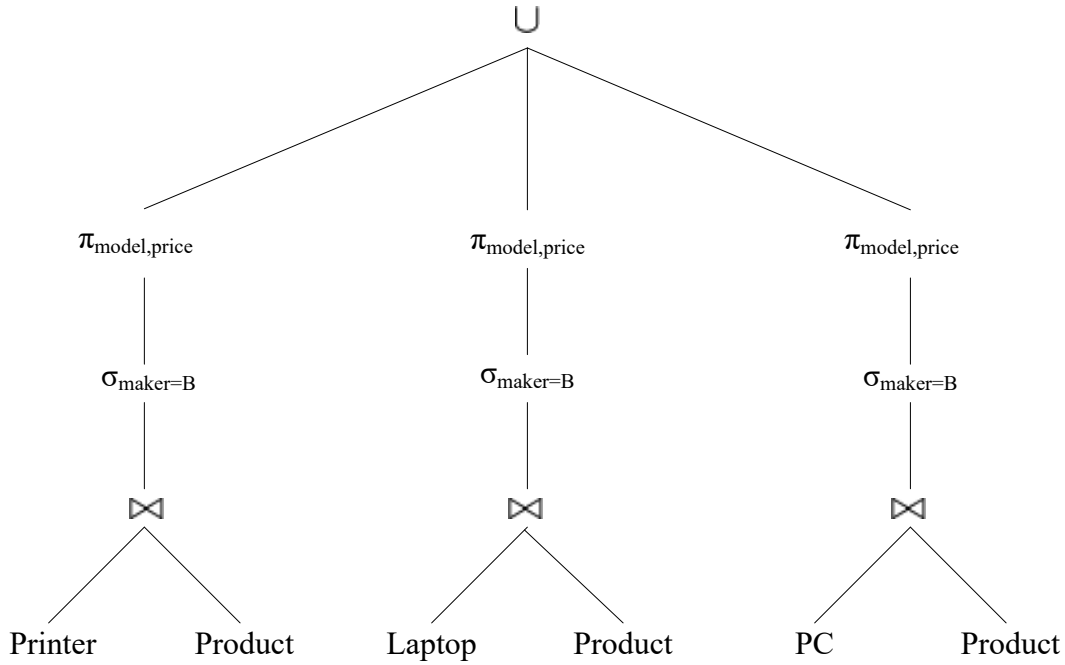
Exercise 2.4.2a



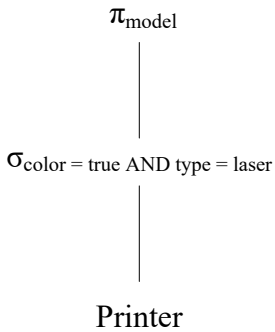
Exercise 2.4.2b



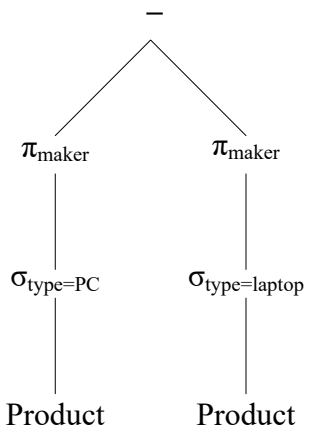
Exercise 2.4.2c



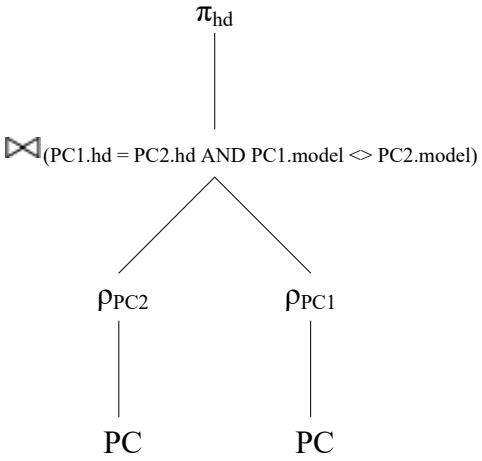
Exercise 2.4.2d



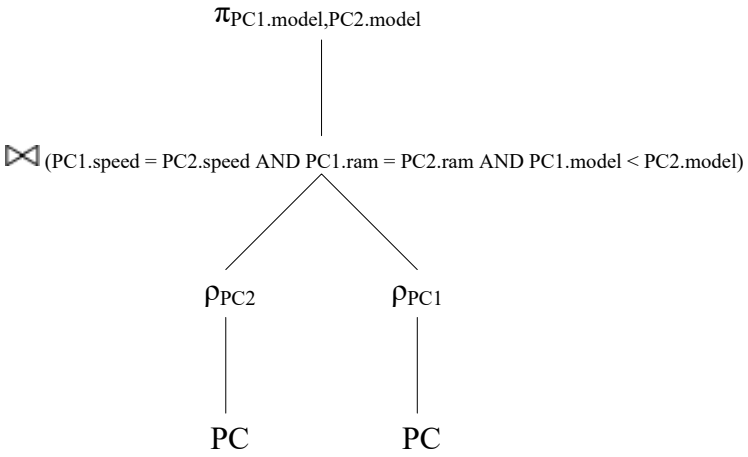
Exercise 2.4.2e



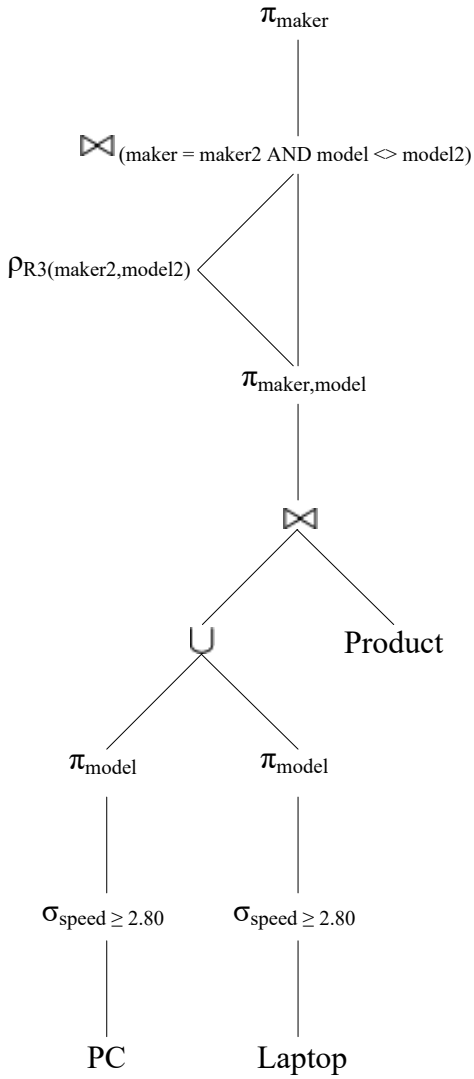
Exercise 2.4.2f



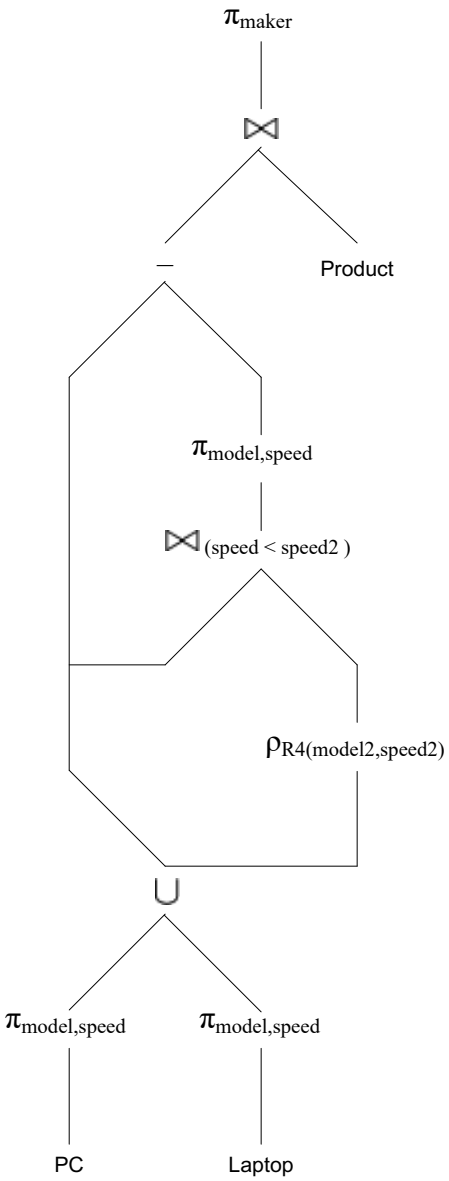
Exercise 2.4.2g



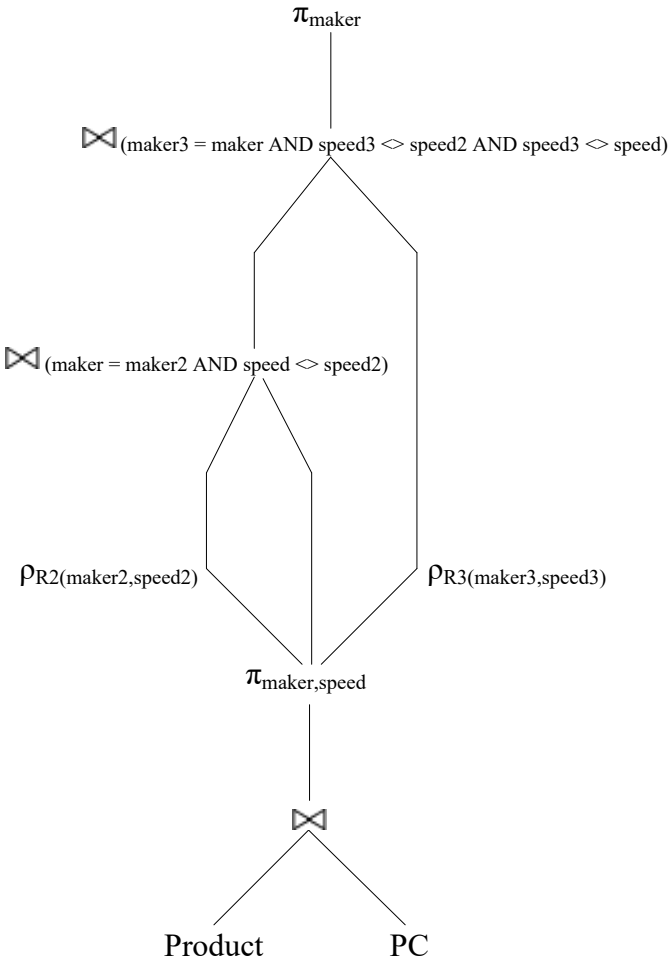
Exercise 2.4.2h



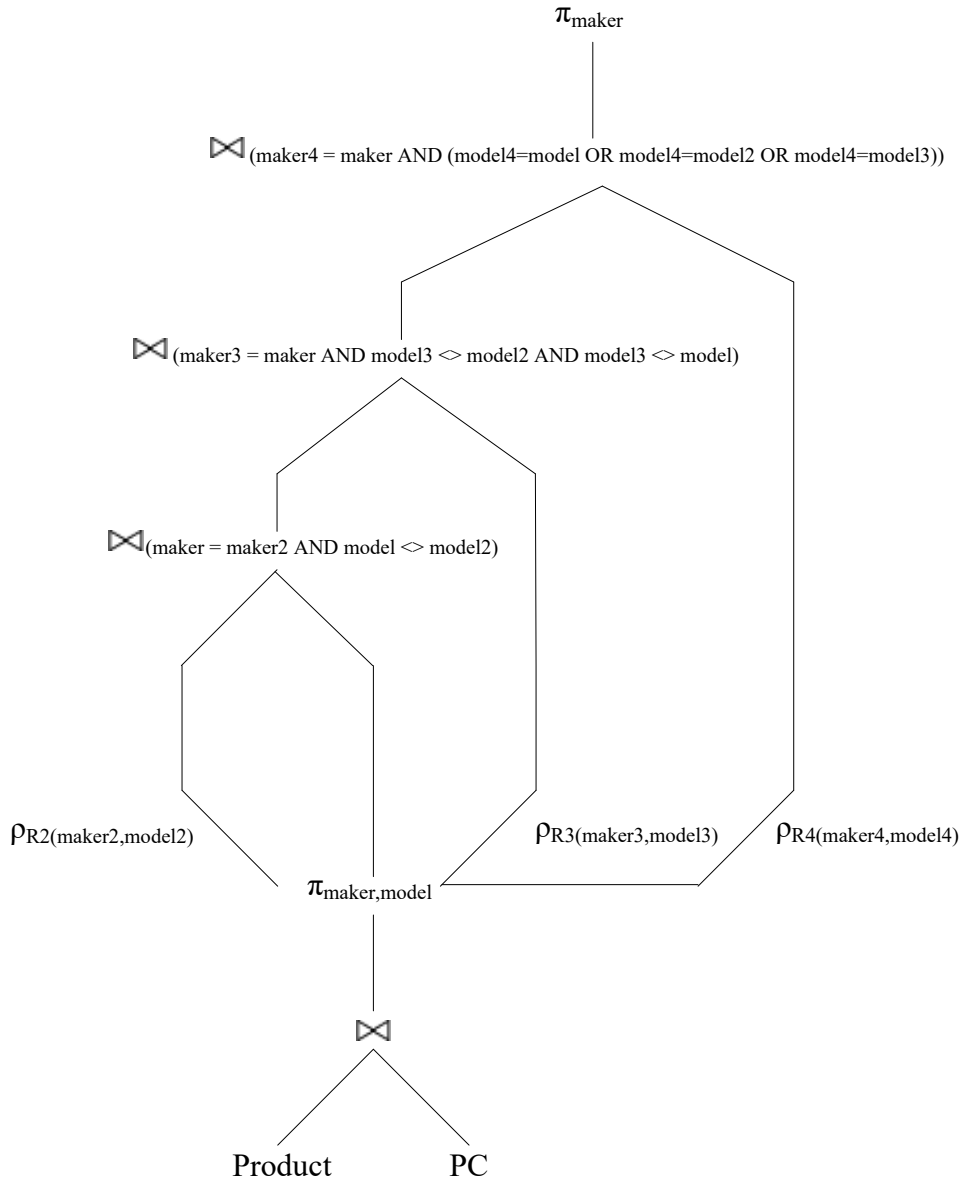
Exercise 2.4.2i



Exercise 2.4.2j



Exercise 2.4.2k



Exercise 2.4.3a

R1 := $\sigma_{bore \geq 16}$ (Classes)
R2 := $\pi_{class,country}$ (R1)

class	country
Iowa	USA
North Carolina	USA
Yamato	Japan

Exercise 2.4.3b

R1 := $\sigma_{launched < 1921}$ (Ships)
R2 := π_{name} (R1)

name
Haruna
Hiei
Kirishima
Kongo
Ramillies
Renown
Repulse
Resolution
Revenge
Royal Oak
Royal Sovereign
Tennessee

Exercise 2.4.3c

R1 := $\sigma_{battle=Denmark\ Strait\ AND\ result=sunk}$(Outcomes)
R2 := π_{ship} (R1)

ship
Bismarck
Hood

Exercise 2.4.3d

R1 := Classes ⋈ Ships
R2 := $\sigma_{launched > 1921\ AND\ displacement > 35000}$ (R1)
R3 := π_{name} (R2)

name
Iowa
Missouri
Musashi
New Jersey
North Carolina
Washington
Wisconsin
Yamato

Exercise 2.4.3e

R1 := $\sigma_{battle=Guadalcanal}$(Outcomes)

R2 := Ships $\bowtie_{(ship=name)}$ R1
R3 := Classes $\bowtie$ R2
R4 := $\pi_{name,displacement,numGuns}$(R3)

name	displacement	numGuns
Kirishima	32000	8
Washington	37000	9

Exercise 2.4.3f

R1 := π_{name}(Ships)
R2 := π_{ship}(Outcomes)
R3 := $\rho_{R3(name)}$(R2)
R4 := R1 $\cup$ R3

name
California
Haruna
Hiei
Iowa
Kirishima
Kongo
Missouri
Musashi
New Jersey
North Carolina
Ramillies
Renown
Repulse
Resolution
Revenge
Royal Oak
Royal Sovereign
Tennessee
Washington
Wisconsin
Yamato
Arizona
Bismarck
Duke of York
Fuso
Hood
King George V
Prince of Wales
Rodney
Scharnhorst

South Dakota
West Virginia
Yamashiro

Exercise 2.4.3g

From 2.3.2, assuming that every class has one ship named after the class.

R1 := π_{class}(Classes)
R2 := π_{class}($\sigma_{name <> class}$(Ships))
R3 := R1 – R2

class
Bismarck

Exercise 2.4.3h

R1 := $\pi_{country}$($\sigma_{type=bb}$(Classes))
R2 := $\pi_{country}$($\sigma_{type=bc}$(Classes))
R3 := R1 ∩ R2

country
Japan
Gt. Britain

Exercise 2.4.3i

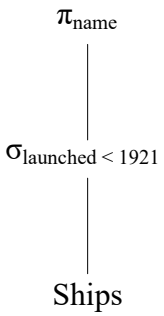
R1 := $\pi_{ship,result,date}$(Battles $\bowtie_{(battle=name)}$ Outcomes)
R2 := $\rho_{R2(ship2,result2,date2)}$(R1)
R3 := R1 $\bowtie_{(ship=ship2\ AND\ result=damaged\ AND\ date < date2)}$ R2
R4 := π_{ship}(R3)

No results from sample data.

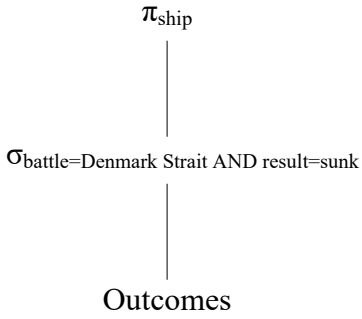
Exercise 2.4.4a

$\pi_{class,country}$
|
$\sigma_{bore \geq 16}$
|
Classes

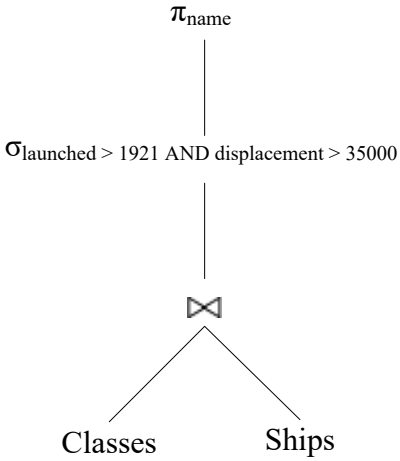
Exercise 2.4.4b



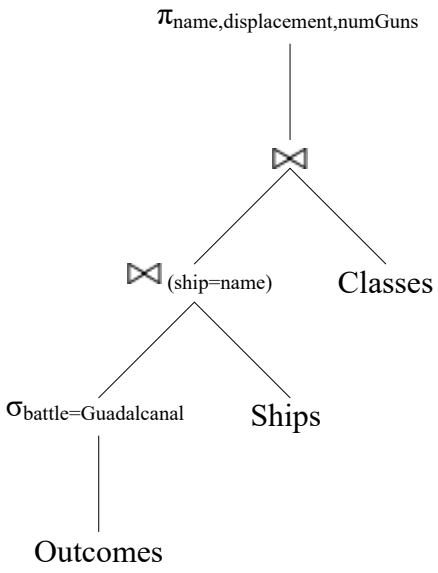
Exercise 2.4.4c



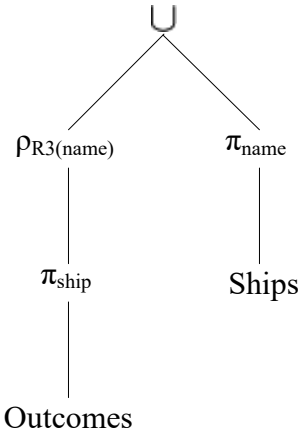
Exercise 2.4.4d



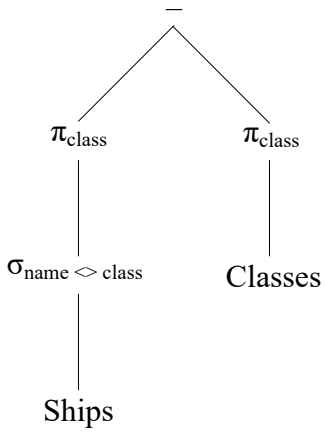
Exercise 2.4.4e



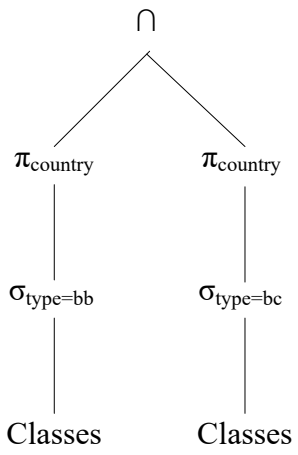
Exercise 2.4.4f



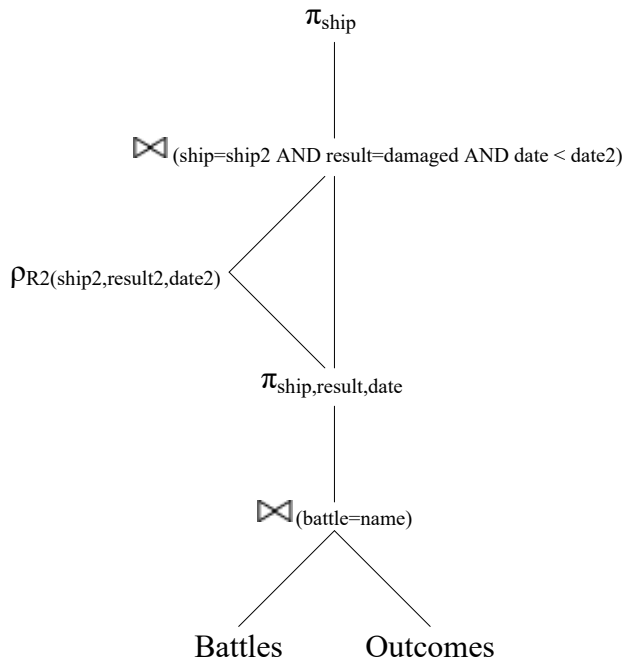
Exercise 2.4.4g



Exercise 2.4.4h



Exercise 2.4.4i



Exercise 2.4.5

The result of the natural join has only one attribute from each pair of equated attributes. On the other hand, the result of the theta-join has both columns of the attributes and their values are identical.

Exercise 2.4.6

Union

If we add a tuple to the arguments of the union operator, we will get all of the tuples of the original result and maybe the added tuple. If the added tuple is a duplicate tuple, then the set behavior will eliminate that tuple. Thus the union operator is monotone.

Intersection

If we add a tuple to the arguments of the intersection operator, we will get all of the tuples of the original result and maybe the added tuple. If the added tuple does not exist in the relation that it is added but does exist in the other relation, then the result set will include the added tuple. Thus the intersection operator is monotone.

Difference

If we add a tuple to the arguments of the difference operator, we may not get all of the tuples of the original result. Suppose we have relations R and S and we are computing R – S. Suppose also that tuple *t* is in R but not in S. The result of R – S would include tuple *t*. However, if we add tuple *t* to S, then the new result will not have tuple *t*. Thus the difference operator is not monotone.

Projection

If we add a tuple to the arguments of the projection operator, we will get all of the tuples of the original result and the projection of the added tuple. The projection operator only selects columns from the relation and does not affect the rows that are selected. Thus the projection operator is monotone.

Selection

If we add a tuple to the arguments of the selection operator, we will get all of the tuples of the original result and maybe the added tuple. If the added tuple satisfies the select condition, then it will be added to the new result. The original tuples are included in the new result because they still satisfy the select condition. Thus the selection operator is monotone.

Cartesian Product

If we add a tuple to the arguments of the Cartesian product operator, we will get all of the tuples of the original result and possibly additional tuples. The Cartesian product pairs the tuples of one relation with the tuples of another relation. Suppose that we are calculating R x S where R has *m* tuples and S has *n* tuples. If we add a tuple to R that is not already in R, then we expect the result of R x S to have $(m + 1) * n$ tuples. Thus the Cartesian product operator is monotone.

Natural Joins

If we add a tuple to the arguments of a natural join operator, we will get all of the tuples of the original result and possibly additional tuples. The new tuple can only create additional successful joins, not less. If, however, the added tuple cannot successfully join with any of the existing tuples, then we will have zero additional successful joins. Thus the natural join operator is monotone.

Theta Joins

If we add a tuple to the arguments of a theta join operator, we will get all of the tuples of the original result and possibly additional tuples. The theta join can be modeled by a Cartesian product followed by a selection on some condition. The new tuple can only create additional tuples in the result, not less. If, however, the added tuple does not satisfy the select condition, then no additional tuples will be added to the result. Thus the theta join operator is monotone.

Renaming

If we add a tuple to the arguments of a renaming operator, we will get all of the tuples of the original result and the added tuple. The renaming operator does not have any effect on whether a tuple is selected or not. In fact, the renaming operator will always return as many tuples as its argument. Thus the renaming operator is monotone.

Exercise 2.4.7a

If all the tuples of R and S are different, then the union has $n + m$ tuples, and this number is the maximum possible.

The minimum number of tuples that can appear in the result occurs if every tuple of one relation also appears in the other. Then the union has $\max(m , n)$ tuples.

Exercise 2.4.7b

If all the tuples in one relation can pair successfully with all the tuples in the other relation, then the natural join has $n * m$ tuples. This number would be the maximum possible.

The minimum number of tuples that can appear in the result occurs if none of the tuples of one relation can pair successfully with all the tuples in the other relation. Then the natural join has zero tuples.

Exercise 2.4.7c

If the condition C brings back all the tuples of R, then the cross product will contain $n * m$ tuples. This number would be the maximum possible.

The minimum number of tuples that can appear in the result occurs if the condition C brings back none of the tuples of R. Then the cross product has zero tuples.

Exercise 2.4.7d

Assuming that the list of attributes L makes the resulting relation $\pi_L(R)$ and relation S schema compatible, then the maximum possible tuples is n. This happens when all of the tuples of $\pi_L(R)$ are not in S.

The minimum number of tuples that can appear in the result occurs when all of the tuples in $\pi_L(R)$ appear in S. Then the difference has max($n - m$, 0) tuples.

Exercise 2.4.8

Defining r as the schema of R and s as the schema of S:

1. $\pi_r(R \bowtie S)$
2. $R \bowtie \delta(\pi_{r \cap s}(S))$ where δ is the duplicate-elimination operator in Section 5.2 pg. 213
3. $R - (R - \pi_r(R \bowtie S))$

Exercise 2.4.9

Defining r as the schema of R

1. $R - \pi_r(R \bowtie S)$

Exercise 2.4.10

$\pi_{A1,A2 \ldots An}(R \bowtie S)$

Exercise 2.5.1a

$\sigma_{speed < 2.00 \text{ AND } price > 500}(PC) = ø$

Model 1011 violates this constraint.

Exercise 2.5.1b

$\sigma_{screen < 15.4 \text{ AND } hd < 100 \text{ AND } price \geq 1000}(Laptop) = ø$

Model 2004 violates the constraint.

Exercise 2.5.1c

$\pi_{maker}(\sigma_{type = laptop}(Product)) \cap \pi_{maker}(\sigma_{type = pc}(Product)) = ø$

Manufacturers A,B,E violate the constraint.

Exercise 2.5.1d

This complex expression is best seen as a sequence of steps in which we define temporary relations R1 through R4 that stand for nodes of expression trees. Here is the sequence:

R1(maker, model, speed) := $\pi_{maker,model,speed}(Product \bowtie PC)$

R2(maker, speed) := $\pi_{maker,speed}$(Product ⋈ Laptop)
R3(model) := π_{model}(R1 $\bowtie_{R1.maker = R2.maker \text{ AND } R1.speed \leq R2.speed}$ R2)
R4(model) := π_{model}(PC)

The constraint is R4 ⊆ R3
Manufacturers B,C,D violate the constraint.

Exercise 2.5.1e

$\pi_{model}(\sigma_{Laptop.ram > PC.ram \text{ AND } Laptop.price \leq PC.price}(PC \times Laptop)) = ø$

Models 2002,2006,2008 violate the constraint.

Exercise 2.5.2a

$\pi_{class}(\sigma_{bore > 16}(Classes)) = ø$

The Yamato class violates the constraint.

Exercise 2.5.2b

$\pi_{class}(\sigma_{numGuns > 9 \text{ AND } bore > 14}(Classes)) = ø$

No violations to the constraint.

Exercise 2.5.2c

This complex expression is best seen as a sequence of steps in which we define temporary relations R1 through R5 that stand for nodes of expression trees. Here is the sequence:

R1(class,name) := $\pi_{class,name}$(Classes ⋈ Ships)
R2(class2,name2) := $\rho_{R2(class2,name2)}$(R1)
R3(class3,name3) := $\rho_{R3(class3,name3)}$(R1)
R4(class,name,class2,name2) := R1 $\bowtie_{(class = class2 \text{ AND } name <> name2)}$ R2
R5(class,name,class2,name2,class3,name3) := R4 $\bowtie_{(class=class3 \text{ AND } name <> name3 \text{ AND } name2 <> name3)}$ R3

The constraint is R5 = ø
The Kongo, Iowa and Revenge classes violate the constraint.

Exercise 2.5.2d

$\pi_{country}(\sigma_{type = bb}(Classes)) \cap \pi_{country}(\sigma_{type = bc}(Classes)) = ø$

Japan and Gt. Britain violate the constraint.

Exercise 2.5.2e

This complex expression is best seen as a sequence of steps in which we define temporary relations R1 through R5 that stand for nodes of expression trees. Here is the sequence:

$R1(ship,battle,result,class) := \pi_{ship,battle,result,class}(Outcomes \bowtie_{(ship = name)} Ships)$
$R2(ship,battle,result,numGuns) := \pi_{ship,battle,result,numGuns}(R1 \bowtie Classes)$
$R3(ship,battle) := \pi_{ship,battle}(\sigma_{numGuns < 9 \text{ AND } result = sunk}(R2))$
$R4(ship2,battle2) := \rho_{R4(ship2,battle2)}(\pi_{ship,battle}(\sigma_{numGuns > 9}(R2)))$
$R5(ship2) := \pi_{ship2}(R3 \bowtie_{(battle = battle2)} R4)$

The constraint is R5 = ø
No violations to the constraint. Since there are some ships in the Outcomes table that are not in the Ships table, we are unable to determine the number of guns on that ship.

Exercise 2.5.3

Defining r as the schema $A_1,A_2,\ldots,A_n$ and s as the schema $B_1,B_2,\ldots,B_n$:

$\pi_r(R) \triangleright \pi_s(S) = ø$ where $\triangleright$ is the antisemijoin

Exercise 2.5.4

The form of a constraint as $E_1 = E_2$ can be expressed as the other two constraints.
Using the "equating an expression to the empty set" method, we can simply say:

$E_1 - E_2 = ø$

As a containment, we can simply say:

$E_1 \subseteq E_2$ AND $E_2 \subseteq E_1$

Thus, the form $E_1 = E_2$ of a constraint cannot express more than the two other forms discussed in this section.