CS157A:
Introduction to Database Management Systems

Chapter 1: Introduction

Chapter 2: The Relational Model of Data

Suneuy Kim

I. Database Management System (DBMS)

- Database is a collection of data that is managed by a DBMS.
- DBMS is specially designed software applications that interact with the user, other applications, and the database itself to capture and analyze data.
- Features of DBMS
  - Data definition – defining schema for the database, removing schema from the database, and altering an existing schema
  - Data modification – inserting, deleting, and updating data
  - Data retrieval – obtaining information from the database for user queries
  - Administration – Registering and monitoring users, enforcing data security, monitoring performance, maintaining data integrity, dealing with concurrency control, and recovering information in case of failures

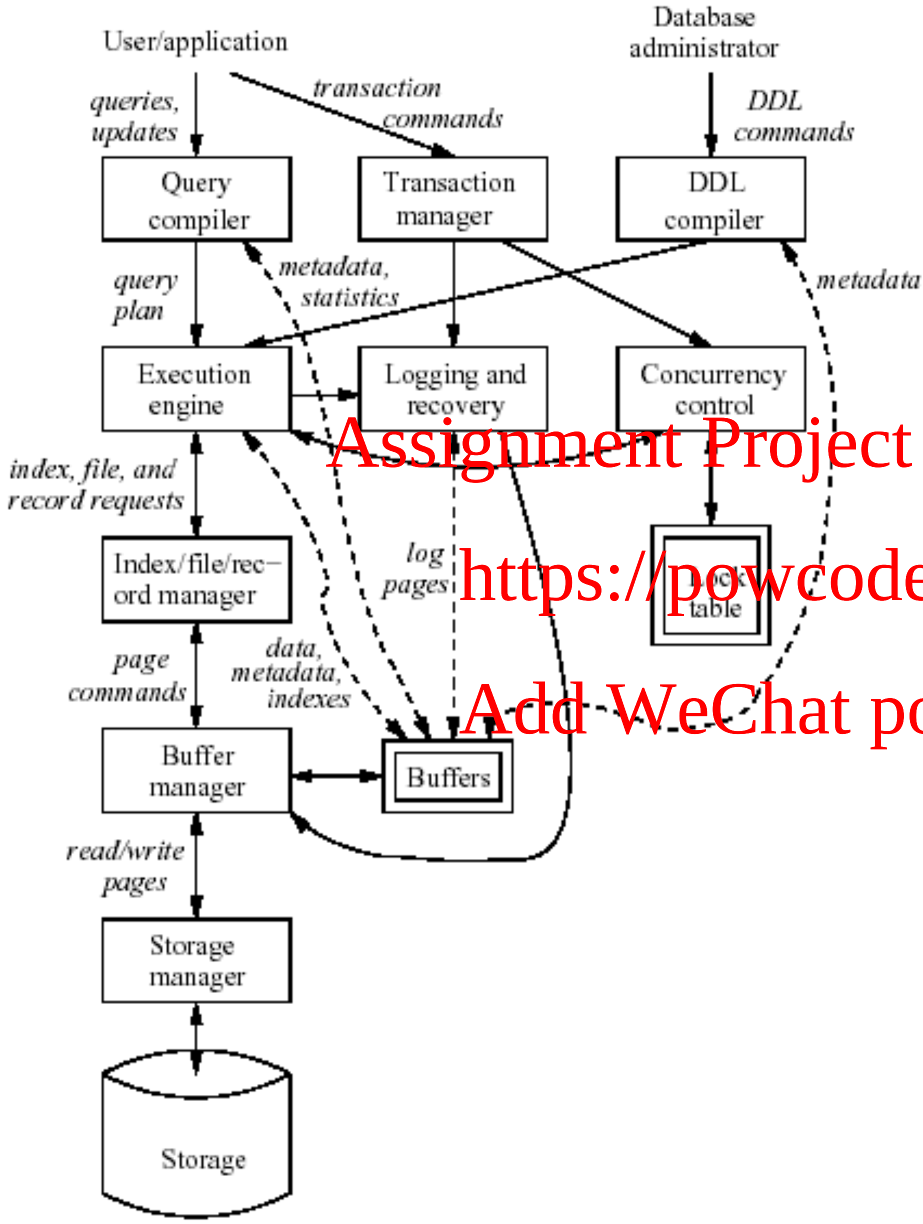
Terminology

- A **database schema** of a database system is its structure described in a formal language supported by the DBMS and refers to the organization of data as a blueprint of how a database is constructed - Wikipedia
- SQL (Structured Query Language) - RDBMS
  - Data Definition Language (DDL) for declaring database schemas

    e.g.) CREATE, DROP, ALTER
  - Data Manipulation Language (DML) for querying and for modifying databases

    e.g.) SELECT, INSERT, DELETE, UPDATE
  - Data Control Language (DCL) for controlling access to data stored in a database

    e.g.) GRANT, REVOKE



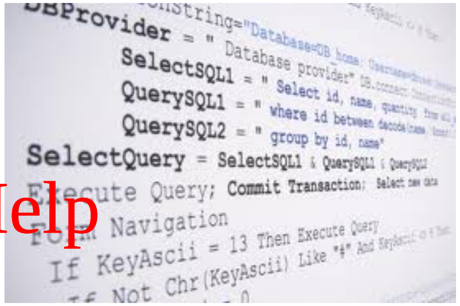
DBMS components

Database People

Database Designer

defines schema

Database Application Programmer



queries/modifies data

Database Administrator

- executes DDL
- loads data,
- monitors and maintains databases

builds system

DBMS Implementer

II. Data Model

- Data model consists of
  - Structure of the data
  - Operations on the data
  - Constraints on the data
- Representative data models
  - Relational data model
  - Semi-structured data model
  - NoSQL data model

Relational Data Model

- Structure: tables (relations)
- Operations – relational algebra, table-oriented

  e.g.) select, project, join, etc.

- Constraints

  e.g.) referential integrity constraints,

  key constraints

Example: Table Movies

title	year	length	genre
Gone with the Wind	1939	231	drama
Star Wars	1977	124	sciFi
Wayne’s World	1992	95	comedy

Semi-structured Data Model

- Structure: trees or graphs

  e.g.) XML data

- Operations involve following paths in the implied tree

  e.g.) /Movies/Movie/Version

- Constraints involve the data type of values associated with a tag

  e.g.) <xs:element name = "Movie" type = "movieType" minOccurs = "0" maxOccurs = "unbounded" />

Example: XML data

```
<?xml version="1.0" encoding="UTF-8"?>
<Movies>
  <Movie title = "King Kong">
    <Version year ="1933">
      <Star>Far Wray</Star>
    </Version>
    <Version year = "1976">
      <Star>Carrie Fisher</Star>
      <Star>Jessica Lange</Star>
    </Version>
   </Movie>
   <Movie title = "Footloose">
      <Version year = "1984">
        <Star> Kevin Bacon</Star>
        <Star>John Lithgow</Star>
        <Star>Sarah Jessica Parker</Star>
      </Version>
    </Movie>
</Movies>
```

Relational vs. Semi-structured data models

- Semi-structured models: flexible
- Relational models:
  - Used by major commercial database systems
  - Efficient access and modification of data
  - Easy of use
  - SQL allows us to program at high level

DB-engines Ranking

http://db-engines.com/en/ranking

Rank Aug 2020	Rank Jul 2020	Rank Aug 2019	DBMS	Database Model
1.	1.	1.	Oracle	Relational, Multi-model
2.	2.	2.	MySQL	Relational, Multi-model
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model
4.	4.	4.	PostgreSQL	Relational, Multi-model
5.	5.	5.	MongoDB	Document, Multi-model
6.	6.	6.	IBM Db2	Relational, Multi-model
7.	↑ 8.	↑ 8.	Redis	Key-value, Multi-model
8.	↓ 7.	↓ 7.	Elasticsearch	Search engine, Multi-model
9.	9.	↑ 11.	SQLite	Relational
10.	↑ 11.	↓ 9.	Microsoft Access	Relational

III. Relation Model

- Relation: two dimensional table to represent data
- Attributes: columns of relation
- Tuples: rows of a relation

- Domains: data type for each attribute

- Relation Schema: name of a relation and the set of attributes (attribute names and associated domains) for a relation

- Database schema - a set of schemas for the relations of a database.

Relational Model

- Instances: actual contents at given point in time

title	year	length	genre
Gone with the wind	1939	231	drama
Star Wars	1977	124	sciFi
Wayne's World	1992	95	comedy

- Key: attribute whose value is unique in each tuple (Or set of attributes whose combined values are unique)

e.g.) Movies(<u>title</u>, <u>year</u>, length, genre)

- Note:
  - The attributes in a relation schema are a set, not a list
  - Relations are sets of tuples, not lists of tuples

Example: Schema
Database Schema about Library

BOOK (
  <u>title</u>: string,
  author:string,
  copies:integer
)
USER (
  <u>uID</u>:integer,
  uNAME:string,
  age:integer,
  loaned:integer,
)

LOAN (
  <u>uID</u>:integer,
  <u>title</u>:string,
  <u>loanDate</u>:date,
  overdue: boolean
)

IV. Relational Algebra

- An algebra whose operands are relations or variables that represent relations.

Core relational operations

- Union, intersection, and difference.
  - *both operands have* the same number of attributes and the domains of the corresponding attributes are the same.
- Selection: selecting certain rows.
- Projection: projecting certain columns.
- Products and joins: combining two relations.
- Renaming of relations and attributes

Running Example

Book(title, author, copies)

User (uID, uName, age, loaned)

Loan (uID, title, loanDate, overdue)

Notes:

- copies means the number of copies left
- loaned means the number of books the user loaned.

Set operations

- R ∪ S

  Relation with tuples from R and S with duplicates removed.

- R ∩ S

  Relation with tuples that appear in both R and S.

- R– S

  Relation with tuples from R but not from S

  Difference operation is NOT commutative. That is, R-S is not equal S-R.

Example: Set operations

Book1

title	author	copies
Faraway Child	Amy Maida Wadsworth	3
Evening in the Ashes	Dorothy Love	20
The Sage and the Lace	James Dove	4

Book2

title	author	copies
Faraway Child	Amy Maida Wadsworth	3
Silent Wife	A.S.A. Harrison	10
Cloud of Unknown	Carl McColman	17

Books1 U Books2

title	author	copies
Evening in the Ashes	Dorothy Love	20
Faraway Child	Amy Maida Wadsworth	3
The Sage and the Lace	James Dove	4
Cloud of Unknown	Carl McColman	17
Silent Wife	A.S.A. Harrison	10

Book1 ∩ Book2

title	author	copies
Faraway Child	Amy Maida Wadsworth	3

Book1 – Book2

title	author	copies
Evening in the Ashes	Dorothy Love	20
The Sage and the Lace	James Dove	4

Select

- R1 := σ_C (R2)
  - C is a condition that involves attributes of R2.
  - R1 is all those tuples of R2 that satisfy C.

Example: Select

- Users who loaned more than 20 books.

  $\sigma_{\text{loaned} > 20}$ (User)

- Users who loaned more than 20 books with age > 10.

  $\sigma_{\text{loaned} > 20 \wedge \text{age} > 10}$ (User)

- Loans of book ‘Bambi’ being overdue

  $\sigma_{\text{title='Bambi'} \wedge \text{overdue=true}}$ (Loan)

Book

title	author	copies

User

uID	uName	age	loaned

Loan

uID	title	loanDate	overdue

Projection

R1 := π_L (R2)

- *L* is a list of attributes from the schema of R2.
- R1 is constructed by looking at each tuple of R2, extracting the attributes on list *L*, in the order specified, and creating from those components a tuple for R1.
- Eliminate duplicate tuples, if any

Example: Projection

- Ids and #of loaned books of all users

$\pi_{uID,loaned}(User)$

- Ids and names of users who loaned more than 20 books

$\pi_{uID,uName}(\sigma_{loaned > 20}(User))$

Book

title	author	copies

User

uID	uName	age	loaned

Loan

uID	title	loanDate	overdue

Different ways of handling duplicates

Titles and overdue information of all loans

$\pi_{\text{title,overdue}}(\text{Loan})$

Relational Algebra: Sets

title	overdue
Bambi	TRUE
Lion King	FALSE
Eye of Sierras	FALSE

SQL: Bags

title	overdue
Bambi	TRUE
Bambi	TRUE
Lion King	FALSE
Eye of Sierras	FALSE

Quiz

Are the following relational algebra expressions useful ?

- $\sigma_{loaned>20}(\sigma_{age>10} (User))$
- $\pi_{title}(\pi_{title,author}(Book))$

Book

title	author	copies

User

uID	uName	age	loaned

Loan

uID	title	loanDate	overdue

Extended Projection

- Using the same π_L operator, where the projection list L can have:

  - an expression x->y where x and y are attributes. x is renamed to y.

  - an expression E->z, where E involves operations and z is the name of the results of the expression

  e.g.) a+b-> x represents sum of the attributes a and b, renamed x

  - duplicate occurrences of the same attribute

Example: Extended Projection

R =

A	B
10	20
30	40

$\pi_{A+B \rightarrow C, A, A}(R)$ =

C	A1	A2
30	10	10
70	30	30

Cartesian Product
(= Cross Join)

R3 := R1 X R2

- Pair each tuple t1 of R1 with each tuple t2 of R2.

- Concatenation t1t2 is a tuple of R3.

- Schema of R3 is the attributes of R1 and then R2, in order.

- But beware attribute *A* of the same name in R1 and R2: use R1.*A* and R2.*A*.

Example: Cartesian Product

R1

A	B
1	2
3	4

X

R2

B	C	D
2	5	6
4	7	8
9	10	11

→

R3

A	R1.B	R2.B	C	D
1	2	2	5	6
1	2	4	7	8
1	2	9	10	11
3	4	2	5	6
3	4	4	7	8
3	4	9	10	11

Example: Cartesian Product

Ids and #of loaned books of users who loaned "Bambi" being overdue.

$\pi_{User.uID,loaned}(\sigma_{User.uID=Loan.uID \wedge title="Bambi" \wedge overdue = true}(User \times Loan))$

Book

title	author	copies

User

uID	uName	age	loaned

Loan

uID	title	loanDate	overdue

Theta-Join

- R3 := R1 $\bowtie_C$ R2
  - Take the product R1 X R2.
  - Then apply σ_C to the result, where C can be any boolean-valued condition.
- User names that happen to be the same as one of the book titles.

$$\pi_{\text{uName}}(\text{User} \bowtie_{\text{uName = title}} \text{Book})$$

Book

title	author	copies

User

uID	uName	age	loaned

Loan

uID	title	loanDate	overdue

Example: Theta Join

A	B
1	2
3	4

$\bowtie_{D\%A=0}$

B	C	D
2	5	6
4	7	8
9	10	11

→

A	B.1	B.2	C	D
1	2	2	5	6
1	2	4	7	8
1	2	9	10	11
3	4	2	5	6

Natural Joins

- R3 := R1 ⋈ R2.
  - Equating attributes of the same name, and
  - Projecting out one copy of each pair of equated attributes.

A	B
1	2
3	4

⋈

B	C	D
2	5	6
4	7	8
9	10	11

→

A	B	C	D
1	2	5	6
3	4	7	8

Example: Natural Join

Ids and #of loaned books of users who loaned "Bambi" being overdue.

$$\pi_{\text{uID,loaned}}(\sigma_{\text{title="Bambi"} \wedge \text{overdue = true}}(\text{User} \bowtie \text{Loan}))$$

Book

title	author	copies

User

uID	uName	age	loaned

Loan

uID	title	loanDate	overdue

Joins

- A *theta join* allows for arbitrary comparison relationships (such as ≥).
- An *equijoin* is a theta join using the equality operator.
- A *natural join* is an equijoin on attributes that have the same name in each relations . The resulting relation will contain only one column for each pair of the same named columns.

Renaming

- The ρ operator gives a new schema to a relation.

- $\rho_{S(A1,\ldots,An)}(R)$ makes S be a relation with attributes A1,…,A*n* and the same tuples as R.

Example: Renaming

R(A,B)

S(B,C,D)

(1) R x $\rho_{S(X,C,D)}$(S)

(2) $\rho_{RS(A,B,X,C,D)}$(RxS)

(1) and (2) are the same except for that resulting relation of (1) doesn't have any name while that of (2) has a name RS.

Division ($R \div S$)

The DIVISION operation is useful for a special kind of query involving **all**. For example,

- Retrieve the names of employees who work on **all** the projects that 'Jonh Smith' works on.
- Find all pizzerias that serve **every** pizza eaten by people over 30.

Division $T(Y) = R(Z) \div S(X)$

$T(Y) = R(Z) \div S(X)$, where

- the Z and X are schemas of R and S, respectively,
- the attributes of S are a subset of the attributes of R; that is $X \boldsymbol{\epsilon} Z$, and
- Y = Z-X (Y is the set of attributes of R that are not attributes of S; that is, Y = Z – X (and hence $Z = X \cup Y$)

$$T(Y) = R(Z) \div S(X)$$

- For a tuple t to appear in the result T of the Division, the values in t must appear in R in combination with every tuple in S.
- Example: b1 and b4 appear in R in combination with all three tuples in S.

R

A	B
a1	b1
a2	b1
a3	b1
a4	b1
a1	b2
a3	b2
a2	b3
a3	b3
a4	b3
a1	b4
a2	b4
a3	b4

÷

S

A
a1
a2
a3

=

T

B
b1
b4

Division using a sequence of π, $\times$, and – operators

- $T1 := \pi_Y(R)$
- $T2 := \pi_Y((S \times T1) - R)$
- $T := T1 - T2$

R

A	B
a1	b1
a2	b1
a1	b2

÷

S

A
a1
a2

=

T

B
b1

T1

B
b1
b2

S × T1

A	B
a1	b1
a1	b2
a2	b1
a2	b2

(S ×T1) - R

A	B
a2	b2

T2

B
b2

T

B
b1

Relationships among Operations

Independent operators

- $\cup$
- $-$
- σ (select)
- π (project)
- x (product)
- ρ (renaming)

Operators that can be expressed in terms of other R.A operators

- $R \cap S = R-(R-S)$
- $R \bowtie_C S = \sigma_C (RXS)$
- $R \bowtie S = \pi_L(\sigma_C (RXS))$
  - C is R.A1=S.A1^R.A2=S.A2 ... where A1, A2, ... are shared attributes by R and S.
  - L is list of attributes of R followed by attributes of S that are not also in R.

Expressing Complex Queries

- Relational algebra expressions
- Expression trees
- Linear Notations

R.A.

Movies(title, year, length, genre, studioName, producerC#)

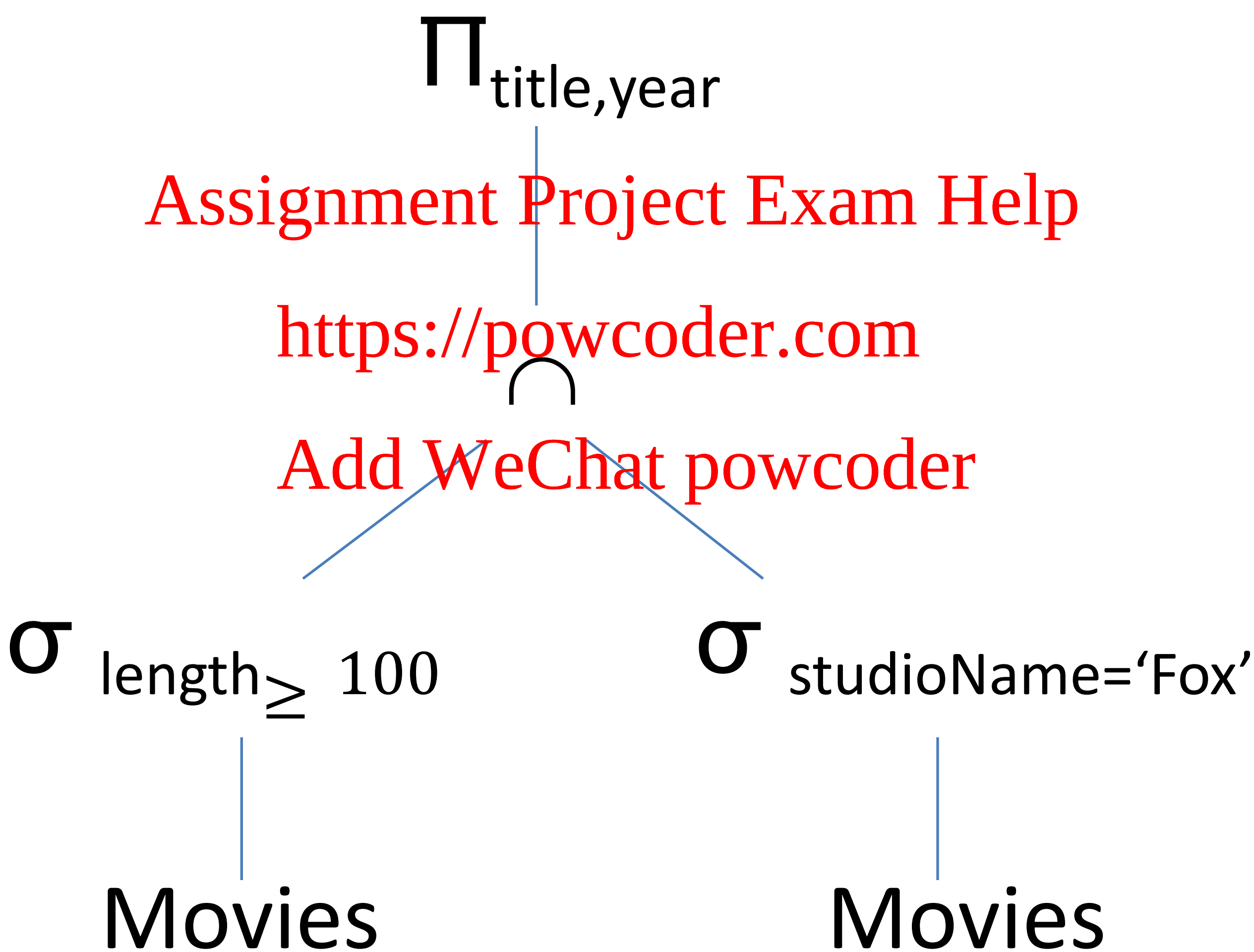
"What is the titles and years of movies made by Fox that are at least 100 minutes long ?"

R.A. Expression:

$\Pi_{title,year}(\sigma_{length \geq 100}(Movies) \cap \sigma_{studioName='Fox'}(Movies))$

$\Pi_{title,year}(\sigma_{length \geq 100 \text{ AND } studioName='Fox'}(Movies))$

Expression Trees



Linear Notations

- $R(t,y,l,g,s,p) := \sigma_{\text{length} \geq 100}(\text{Movies})$
- $S(t,y,l,g,s,p) := \sigma_{\text{studioName='Fox'}}(\text{Movies})$
- $T(t,y,l,g,s,p) := R \cap S$
- $\text{Answer(title, year)} := \Pi_{t,y}(T)$

or

- $\text{Answer(title, year)} := \Pi_{t,y}(R \cap S)$

Constraints on Relations

A <u>referential integrity constraint</u> asserts that a value appearing in one context will also appear in another related context.

<u>Example</u>


```
Movies (title, year, length, genre,
stuidoName, producerC#)
MovieExec(name, address,cert#,netWorth)
```


$$\Pi_{\text{producerC\#}}(\text{Movies}) \in \Pi_{\text{cert\#}}(\text{MovieExec})$$

Constraints on Relations

Key constraints

A key uniquely identifies each tuple in a relation.

Any two tuples in a relation must not have the same key.

Relational Algebra Exercise 2.4.1

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 ?

(b) Which manufacturers make laptops with a hard disk of at least 100GB ?

(c) Find the model number and price of all products (of any type) made by manufacturer B.

(d) Find the model numbers of all color laser printers.

(e) Find those manufacturers that sell Laptops, but not PCs.

(f) Find those hard disk sizes that occur in two or more PC's

(g) Find those pairs of PC models that have both the same speed and RAM. **A pair should be listed only once**; e.g., list(i,j) but not (j,i)

(h) Find the manufacturers of **at least two** different computers (PC's or laptops) with speeds of at least 2.80.

(i) Find the manufacturer(s) of the computer (PC or laptop) with **the highest available** speed.

(j) Find the manufacturers of PC's with **at least three** different speeds.

(k) Find the manufacturers who sell **exactly three** different models of PC.

(a) What PC models have a speed of at least 3.00 ?

$\pi_{\text{model}} (\sigma_{\text{speed} \geq 3.0} \text{PC})$

PC.model
1005
1006
1013

(b) Which manufacturers make laptops with a hard disk of at least 100GB ?

π_{maker} (σ $_{hd \geq 100}$ (Product $\bowtie$ Laptop))

Product.maker
'A'
'B'
'E'
'F'
'G'

(c)Find the model number and price of all products (of any type) made by manufacturer B.

$\pi_{\text{model,price}}$

(

$\sigma_{\text{maker = 'B'}}$

(Product $\bowtie$

$(\pi_{\text{model,price}}(\text{PC}) \cup \pi_{\text{model,price}}(\text{Laptop}) \cup \pi_{\text{model,price}}(\text{Printer}))$

)

)

Product.model	PC.price
1004	649
1005	630
1006	1049
2007	1429

(d) Find the model numbers of all color laser printers

$\pi_{\text{model}} (\sigma_{\text{color = true and type = 'laser'}} \text{Printer})$

Printer.model
3003
3007

(e) Find those manufacturers that sell Laptops, but not PCs.

$$\pi_{maker}(\sigma_{type='laptop'}\,Product) - \pi_{maker}(\sigma_{type='pc'}\,Product)$$

Product.maker
'F'
'G'

(f) Find those hard disk sizes that occur in two or more PC's

$\pi_{PC.hd}$

$(PC \bowtie_{PC.model \neq PC2.model \text{ and } PC.hd = PC2.hd} \rho_{PC2}\ PC)$

PC.hd
250
80
160

(g) Find those pairs of PC models that have both the same speed and RAM. **A pair should be listed only once**; e.g., list(i,j) but not (j,i)

$$\pi_{\text{PC.model, PC2.model}} (PC \bowtie_{\text{PC.model < PC2.model and PC.speed = PC2.speed and PC.ram = PC2.ram}} \rho_{PC2}\, PC)$$

PC.model	PC2.model
1004	1012

(h) Find the manufacturers of **at least two** different computers (PCs or laptops) with speeds of at least 2.80.

π_{maker}

$\sigma_{\text{model}\neq\text{model2}}$

(

$\pi_{\text{maker,model}}$ (Product $\bowtie$ ((π_{model} $\sigma_{\text{speed} \geq 2.80}$ PC)$\cup$(π_{model} $\sigma_{\text{speed} \geq 2.80}$ Laptop)))

$\bowtie$

$\rho_{\text{model2<-model}}$

$\pi_{\text{maker,model}}$ (Product $\bowtie$ (($\pi_{\text{model } \sigma \text{ speed} \geq 2.80}$ PC)$\cup$(π_{model} $\sigma_{\text{speed} \geq 2.80}$ Laptop)))

)

Product.maker
'B'
'E'

(i) Find the manufacturer(s) of the computer (PC or laptop) with **the highest available** speed.

π maker
(
Product $\bowtie$
(
π_{model}
$\sigma_{\text{type = 'pc' or type = 'laptop'}}$ (Product)
-
π_{model}
$((\pi_{\text{model,speed}}\ \text{PC} \cup \pi_{\text{model, speed}}\ \text{Laptop})$
$\bowtie_{\text{speed < speed2}}$
$\rho_{\text{model1<- model, speed2<-speed}}$
$(\pi_{\text{model,speed}}\ \text{PC} \cup \pi_{\text{model, speed}}\ \text{Laptop}))$
)
)

Product.maker
'B'

(j) Find the manufacturers of PCs with **at least three** different speeds.

π_{maker} ($\sigma_{\text{s1} \neq \text{s2 and s1} \neq \text{s3 and s2} \neq \text{s3}}$ (

($\rho_{\text{s1} \leftarrow \text{speed}}$ $\pi_{\text{maker,speed}}$ (Product $\bowtie$ PC)

$\bowtie$

$\rho_{\text{s2} \leftarrow \text{speed}}$ $\pi_{\text{maker,speed}}$ (Product $\bowtie$ PC))

$\bowtie$

$\rho_{\text{s3} \leftarrow \text{speed}}$ $\pi_{\text{maker, speed}}$ (Product $\bowtie$ PC))

)

Product.maker
'A'
'D'
'E'

(k) Find the manufacturers who sell **exactly three** different models of PC.

π_{maker}

$\sigma_{\text{m1} \neq \text{m2 and m2} \neq \text{m3 and m1} \neq \text{m3}}$

$((\rho_{\text{m1<-model}}\ \pi_{\text{maker,model}}\ \sigma_{\text{type='pc'}}\ \text{Product}) \bowtie$

$(\rho_{\text{m2<-model}}\ \pi_{\text{maker,model}}\ \sigma_{\text{type='pc'}}\ \text{Product}) \bowtie$

$(\rho_{\text{m3<-model}}\ \pi_{\text{maker,model}}\ \sigma_{\text{type='pc'}}\ \text{Product}))$

-

π maker

$(\sigma_{\text{m1} \neq \text{m2 and m1} \neq \text{m3 and m1} \neq \text{m4 and m2} \neq \text{m3 and m2} \neq \text{m4 and m3} \neq \text{m4}}$

$((\rho_{\text{m1<-model}}\ \pi_{\text{maker,model}}\ \sigma_{\text{type='pc'}}\ \text{Product}) \bowtie$

$(\rho_{\text{m2<-model}}\ \pi_{\text{maker,model}}\ \sigma_{\text{type='pc'}}\ \text{Product}) \bowtie$

$(\rho^{\text{m3<-model}}\ \pi_{\text{maker,model}}\ \sigma_{\text{type='pc'}}\ \text{Product}) \bowtie$

$(\rho^{\text{m4<-model}}\ \pi_{\text{maker,model}}\ \sigma_{\text{type='pc'}}\ \text{Product})$

))

Product.maker
'A'
'B'
'D'
'E'