Chapter 5

Different Types of queries

Nested Queries

exists vs unique – exist if the result of the nested query contains at least 1 tuple. Unique

basic structure

select <attribute and function list>
from <table list>
[where <condition>]
[group by <grouping attribute(s)>]
[having <group condition>]
[order by <attribute list>];

command formats

drop schema company cascade;
drop table dependent cascade;
alter table company.employee add column Job VARCHAR(12);
alter table company.employee drop column Address Cascade;
alter table company.department alter column mgr_ssn drop default;
alter table company.department alter column mgr_ssn set default '333445555';
alter tale company.employee drop constraint empsuperfk cascade;

Exercises 5.5, 5.6, 5.7

5.5

a.

select dname, count(*)
from department, employee
where dno = dnumber
group by dname
having avg(salary) > 30000;

b.

select dname, count(*)
from department, employee
where dno = dnumber and upper(gender) = 'M'
group by dname
having avg(salary) > 30000;

another interruption of problem

select dname, count(*)
from department, employee
where dno = dnumber and upper(gender) = 'M' and salary > 30000
group by dname;

5.6

a.

```
select e.ename, e.fname, e.major
from employee e,
where not exists (select *
                  from grade_report g
                  where e.student_number = g.student_number and upper(grade) != 'A');
```

b.

```
select e.ename, e.fname, e.major
from employee e,
where not exists (select *
                  from grade_report g
                  where e.student_number = g.student_number and upper(grade) = 'A');
```

5.7

a.

```
select e.fname, e.lname
from employee e
where e.dno in (select ee.dno
                from employee ee
                where ee.salary = (select max(eee.salary)
                                   from employee eee));
```

b.

```
select e.fname, e.lname
from employee e
where e.superssn in (select ee.ssn
                     from employee ee
                     where ee.superssn = '888665555');
```

c.

```
select e.fname, e.lname
from employee e
where e.salary >= 10000 + (select min(ee.salary)
                           from employee ee);
```

<u>Chapter 6 – 6.1 to 6.5</u>

Relational Algebra Symbols

Operation	Symbol
Project	π
select	σ
rename	ρ
union	$\cup$
intersection	$\cap$
difference	-
assignment	<--
cartesian product	x
join	$\bowtie$
natural join	*
Division	$\div$
aggregate	$\mathfrak{F}$

Select Format:	$\sigma_{dno = 4}$ (Employee)
Project Format:	$\pi_{lname, fname, salary}$(Employee)
Rename:	Result1 <-- $\sigma_{dno = 4}$ (Employee)
Rename V2:	Result1 <-- $\rho_{(dname, dnum, mgr\_ssn, mgr\_start\_date)}$ (Employee)
Cartesian Product:	Employee x Result1
Join:	Department $\bowtie_{Mgr\_ssn = ssn}$ Employee
Natural Join:	Department * Employee
Difference:	Department – Employee
Aggregate:	dno $\mathfrak{F}_{Count\ ssn,\ Average\ salary}$(Employee)
Division:	Department $\div$ Employee

Select – returns a tuple based on selection condition specified
Project – returns a tuple
Rename – renames the resulting tuple, or renames individual attributes
Cartesian Product – resulting join is a complete cross. Everything on left is crossed with everything on right. 5 tuples on left and 3 on right creates 15 tuples
Join - joins to relations based on a specific attribute in each relation specified. The resulting relation has the number of tuples where they match.
Natural Join – Joins two relations where they have matching attributes. The resulting relation has the number of tuples where they match.
Difference – The resulting relation contains tuples that existed in the left side, but do not exist on the right side. The two relations have the same attributes by name and in same order.
Aggregate – Used for max, min, sum, avg, count, etc.
Division – identifies the attribute values from a relation that are found to be paired with all values from another relation

Exercises 6.16, 6.18

6.16

a.
Retrieve the names of all employees in department 5 who more than 10 hours per week on ProductX project.

Proj <-- $\sigma_{\text{pname = 'ProjectX'}}$ (Project)
Proj2 <-- Proj $\bowtie_{\text{pnumber = pno}}$ (Works_On)
Proj3 <-- $\sigma_{\text{hours > 10}}$ (Proj2)
Proj4 <-- $\sigma_{\text{dno = 5}}$ (Employee)
Result <-- $\pi_{\text{lname, fname}}$ (Proj4 $\bowtie_{\text{ssn = essn}}$ (Proj3))

b.
List the names of all employees who have a dependent with the same first name as themselves.

Result <-- $\pi_{\text{lname, fname}}$ (Employee $\bowtie_{\text{ssn = essn and fname = dependent\_name}}$ (Dependent))

c.
List the names of all employees who are directly supervised by 'Franklin Wong'.

R1 <-- π_{ssn} ($\sigma_{\text{fname = 'Franklin and lname = 'Wong'}}$ (Employee))
Result <-- $\pi_{\text{lname, fname}}$ (Employee $\bowtie_{\text{ssn = super\_ssn}}$ (R1))

d.
For each project, list the project name and the total hours per week (by all employees) spent on that project.

R1(pno, sum_hours) <-- pno $\mathfrak{F}_{\text{Sum Hours}}$ (Works_On)
Result <-- $\pi_{\text{pname, sum\_hours}}$ (Project $\bowtie_{\text{pnumber = pno}}$ (R1))

e.
Retrieve the names of all employees who work on every project.

R1 (pno) <-- π_{pnumber} (Project)
R2 <-- $\pi_{\text{essn, pno}}$ (Works_On)
R3 <-- R2 ÷ R1
Result <-- $\pi_{\text{lname, fname}}$ (Employee $\bowtie_{\text{ssn = essn}}$ (R3))

f.
Retrieve the names of all employees who do not work on any project.

R1 <-- π_{ssn} (Employee)
R2 (ssn) <-- π_{essn} (Works_On)
R3 <-- R1 - R2
Result <-- $\pi_{\text{fname, lname}}$(R3 * Employee)

g.

For each department, retrieve the department name and the average salary of all employees working in that department.

R1(dno, avg_salary) <-- dno $\Im$ $_{\text{Avg salary}}$ (Employee)

Result <-- $\pi_{\text{dname, avg\_salary}}$ (Department $\bowtie_{\text{dnumber = dno}}$ (R1))

h.
Retrieve the average salary of all female employees.

R1 <-- π_{salary} ($\sigma_{\text{gender = 'F'}}$ (Employee))
Result <-- $\Im$ $_{\text{Avg salary}}$ (R1)

i.
Find the names and addresses of all employees who work on at least one project located in Houston but whose department has no location in Houston.

All employees work on one projectin houston

all employees whose department not in houston

join

R1 <-- π_{pnumber} ($\sigma_{\text{plocation = 'Houston'}}$ (Project))
R2 <-- Works_On $\bowtie_{\text{pno= pnumber}}$ (R1)
R3 <-- Employee $\bowtie_{\text{ssn= essn}}$ (R2)) // all employees who work on a project in houston

R4 <-- $\sigma_{\text{dlocation = 'Houston'}}$ (Dept_Locations))
R5 <-- Employee $\bowtie_{\text{dno = dnumber}}$ (R4))
R6 <-- Employee – R5 // all employees who's department is not in houston
Result <-- $\pi_{\text{fname, lname, address}}$ (R3 * R6) // join would be all employees who are in both

j.
List the last names of all department managers who have no dependents

R1 (ssn) <-- $\pi_{\text{mgr\_ssn}}$ (Department)

R2 (ssn) <-- π_{essn} (Dependent)
R3 <-- R1 – R2

Result <-- $\pi_{\text{fname, lname}}$ (R3 * Employee)

6.18

a. How many copies of the book titled ‘The Lost Tribe’ are owned by the library branch whose name is ‘Sharpton’?

b.
How many copies of the book 'The Lost Tribe' are owned by each library branch?

c.
Retrieve the names of all borrowers who do not have any books checked out.

d.
For each book that is loaned out from the Sharpstown branch and whose due date is today, retrieve the book tile, the borrower's name and the borrower's address.

e.
For each library branch, retrieve the branch name and the total number of books loaned out from that branch.

f.
Retrieve the names, addresses, and number of books checked out for all borrowers who have more than five books checked out.

g.
For each book authored (or coauthored) by Stephen King, retrieve the title and the number of copies owned by the library branch whose name is Central

Chapter 15

2nf Formal Definition - every non prime attribute is not partially dependent on any candidate key.
3nf Formal Definition - whenever a nontrivial dependency hold x -> a in R, either X is a superkey of R, or A is a prime attribute of R.
BCNF – whenever a nontrivial functional dependency x -> a holds in R, then X is a superkey of R

x
Exercises 15.30, 15.31, 15.35, 15.36

15.30

Car_Sale(Car#, Date_sold, Salesperson#, Commission%, Discount_amt)

Date_sold -> discount_amt
Salesperson# -> commission#

PK = Car#, Salesperson#

1NF:

The relation is in 1NF because there are no multivalue or composite attributes in Car_Sale

2NF:

Not in 2NF because there exists non prime attributes that are not dependent on a full candidate key. In this case, commission# is only dependent on part of the key, salesperson#.

To Normalize:

Move the attribute that is causing the problem, commission#, and put it in it's own relation. Use as the primary key, salesperson#, but keep salesperson# in the original relation.

The result after putting in 2NF is:

Car_Sale(Car#, Date_sold, Salesperson#, Discount_amt)
Commission(Salesperson#, commission#)

3NF: We are not in 3NF because of the data_sold -> discount_amt FD. Data_sold is not a superkey and discount_amt is not a prime attribute so we are in violation of 3NF. To solve this, we move the problem attribute, discount_amt, to a new relation and use as it's primary key, dale_sold, which stays in the orignla relation. The result after putting in 3NF is:

Car_Sale(Car#, Date_sold, Salesperson#)
Date(Date_sold, discount_amt)
Commission(Salesperson#, commission#)

15.31

Book(Book_title, author_name, book_type,list_price, author_afil, publisher)

FD1 book_title -> publisher, book_type
FD2 book_type -> list_price
FD3 author_name -> author_afil

Determine primary key

L - book_title, author_name
M - book_type
R - publisher, list_price, author_afil

Closures

book_title+ = book_title, publisher, book_type, list_price
author_name+ = authro_name, author_afil

Book_title, author_name+ = book_title, publisher, book_type, list_price, author_name, author_a primary key = book_title, author_name

1NF: We are in 1NF because there are no multivalued or composite attributes

2NF: We are in violation of 2NF because a non prime attribute, publisher, is determined by only part of the only key. Since publisher is only determined by book, and not author_name as well, we are in violation of 2NF. Publisher is not the only offender. Book_type and author_afil are also offenders. To fix this, we take the offender and move them into their own relations, using the other half of the dependency that causes the problem as the key. The result is:

Book, Author_Name

Book, publisher, book_type, list_price

Author_name, author_afil

3NF: We are in violation of 3NF because in the second relation, a non prime attribute, list_price, is determined by another non prime attribute, book_type. To fix this, we move the offending attribute, list_price into a new relation, with primary key being book_type. We keep book_type in the original relation. The result is:

<u>Book_title, Author_Name</u>

<u>Book_title</u>, publisher, book_type

<u>book_type</u>, list_price

<u>Author_name</u>, author_afil

15.35

Book(Book_name,author, Edition, Year)

a.

possible candidate keys

book_name,author, edition

b.

book ->-> author
book ->-> edition,year

These two above are MVD's because author is a multivalued attribute given a particular book name. Same goes for edition, year, which is also a multivalued set of attributes for a given book. Also, since book ->-> author, we automatically get book ->-> edition, year c.

The decomposition would be to take the two multivalued items and move them into their own relations.

Book(book, author)
Book2(book, edition, year)

15.36

Trip(trip_id,start_date,cities_visited,cards_used)

a.

FD trip_id -> start_date
MVD trip_id ->-> cities_visited
MVD trip_id ->-> cards_used

b.

To normalize, move the two MVD's into their own relations, leaving trip_id and start_date in the original Trip relation.

Trip(<u>trip_id</u>,start_date)
Cities(<u>trip_id,cities_visited</u>)
Cards(<u>trip_id, cards_used</u>)

<u>Chapter 16</u>

Inference Rules

IR1 (reflexive rule): attributes determine self and subset of self
IR2 (augmentaiton rule): {x -> y} |= xz -> yz
IR3 (transitive): {x -> y, y -> z} |= xz -> yz
IR4 (decomposition): {x -> yz} |= x -> y and x -> z
IR5 (union or additive): {x -> y, x -> z} |= x -> yz
IR6 (pseudotransitive): {x -> y, wy -> z} |= wx -> z

<u>Nonadditive jon test for binary decompositions</u>

The FD (($R1 \cap R2$) -> ($R1 - R2$)) is in F+ or
The FD (($R1 \cap R2$) -> ($R2 - R1$)) is in F+

<u>Nonadditive jon test for n-ary tests</u>

R = ssn, ename, pnumber, pname, plocation, hours
R1 = EMP = ssn, ename
R2 = Proj = pnumber, pname, plocation
R3 = Works_on = ssn, pnumber, hours

ssn -> ename
pnumber -> name, plocation
ssn,pnumber -> hours

	SSN	**Ename**	**Pnumber**	**Pname**	**Plocation**	**hours**
R1	a1	a2	B13	B14	B15	b16
R2	b21	b22	A3	A4	A5	b26
R3	a1	~~B32~~ a2	a3	~~B34~~ a4	~~B35~~ a5	a6

Exercises 16.32

Refrig(Model#, year, price, manuf_plant, color)
Refrig(M, y, p, mp, c)

FD1: m -> mp
FD2: m,y -> p
FD3: MP -> c

a.

m+ = m, mp, c NO
m,y+ = m,y,mp,c,p YES
m,c+ = m,c,mp NO

b.

Not in 3NF because it's not in 2nf because mp is determined by part of the only candidate key, there fore can't be in 3nf. Also, fd3 has a transitive dependecy because the left side is not a super key and right is not prime

Not in BCNF because not in 2nf and because on FD1, m is not a super key of Refrig.

c.

R1 = m,y,p
R2 = m,mp,c

	M	Y	P	Mp	C
R1	a1	a2	a3	~~B14~~ a4	~~B15~~ a5
R2	a1	b22	b23	a4	a5