Homework

1. Consider the LIBRARY relational schema shown in the following igure, which is used to keep track of books, borrowers, and book loans. Referential integrity constraints are shown as directed arcs. Write down relational algebra expressions for the following queries on the LIBRARY database:

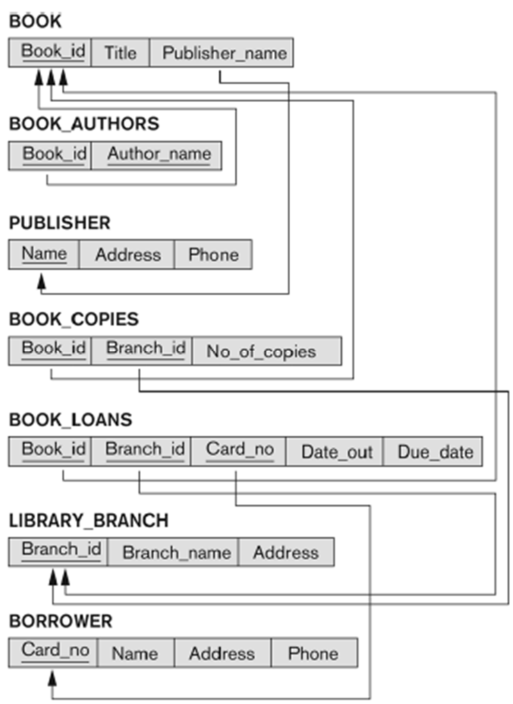

Figure 6.14
A relational database schema for a LIBRARY database.

(a) How many copies of the book titled The Lost Tribe are owned by the library branch whose name is "Sharpstown"?

A <-- BOOKCOPIES * LIBRARY-BRANCH * BOOK

RESULT <-- $\prod_{\text{No\_Of\_Copies}} (\sigma_{\text{BranchName='Sharpstown' and Title='The Lost Tribe'}}$

(b) How many copies of the book titled The Lost Tribe are owned by each library branch?

$\prod_{\text{BranchID,No\_Of\_Copies}} ((\sigma_{\text{Title='The Lost Tribe'}} (\text{BOOK})) * \text{BOOKCOPIES})$

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

NO_CHECKOUT_B <-- $\prod_{CardNo}$ (BORROWER) - $\prod_{CardNo}$ (BOOK_LOANS)

RESULT <-- $\prod_{Name}$ (BORROWER * NO_CHECKOUT_B)

(d) For each book that is loaned out from the "Sharpstown" branch and whose DueDate is today, retrieve the book title, the borrower's name, and the borrower's address.

S <-- $\prod$ BranchId ($\sigma_{BranchName='Sharpstown'}$ (LIBRARY-BRANCH))

B_FROM_S <-- $\prod$ BookId,CardNo (($\sigma_{DueDate='today'\ (BOOKLOANS)}$)) * S)

RESULT <-- $\prod_{Title,Name,Address}$ (BOOK * BORROWER * B_FROM_S)

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

R(BranchId,Total) <-- $_{BranchId}$ £ $_{COUNT}$(BookId,CardNo) (BOOK_LOANS)

RESULT <-- $\prod_{BranchName,Total}$ (R * LIBRARY_BRANCH)

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

B(CardNo,TotalCheckout) <-- $_{CardNo}$ £ $_{COUNT}$(BookId) (BOOK_LOANS)

B5 <-- $\sigma_{TotalCheckout > 5}$ (B)

RESULT <-- $\prod_{Name,Address,TotalCheckout}$ (B5 * BORROWER)

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

SK(BookId,Title) <-- ($\sigma_{AuthorName='Stephen\ King'}$ (BOOK_AUTHORS)) * BOOK

CENTRAL(BranchId) <-- $\sigma_{BranchName='Central'}$ (LIBRARY_BRANCH)

RESULT <-- $\prod_{Title,NoOfCopies}$ (SK * BOOKCOPIES * CENTRAL)

---

2. Download the script1 posted on the class website, run the script to create the three tables and design the following queries using PostgreSQL.

a). Find the name(s) of the supplier(s) that haven't supplied any part. .

```
SELECT DISTINCT sname
FROM        suppliers S
WHERE       S.sid NOT IN (SELECT C.sid
                          FROM   catalog C )
```

b). Find the snames of suppliers who supply all red parts.

```
SELECT S.sname
FROM Suppliers S
WHERE NOT EXISTS (( SELECT P.pid
                   FROM Parts P
                   WHERE  P.color = 'red')
                   EXCEPT
                  (SELECT C.pid
                   FROM Catalog C, Parts P
                   WHERE C.sid = S.sid AND
                   C.pid = P.pid AND P.color = 'red' ))
```

c). For each part, find the name of the supplier that changes the least for that part.

```
SELECT      P.pid, S.sname
FROM        Parts P, Suppliers S, Catalog C
WHERE       C.pid = P.pid  AND  C.sid = S.sid
           AND   C.cost =  (SELECT MIN(C1.cost)
                            FROM Catalog C1
                            WHERE C1.pid = P.pid)
```

d). Find the name of the supplier who supplies the most number of parts.

```
SELECT DISTINCT    S.sname, count(*)
FROM          suppliers S INNER JOIN catalog C on S.sid = C.sid
GROUP BY S.sname
HAVING   count(*) >= (SELECT count(*)
                          FROM suppliers S1 INNER JOIN catalog C1 on S1.sid =
C1.sid
                          GROUP BY  S1.sname)
```

e). Find the names of the parts that have not been supplied by any supplier.

```
SELECT          P.pname
FROM            Parts P
WHERE           P.pid  NOT IN (SELECT pid)
                              FROM Catalog)
```

f). Find the sids of suppliers who supply a red part or a green part.

```
SELECT    DISTINCT S.sname
FROM      Suppliers S, Catalog C, Parts P
WHERE     S.sid = C.sid AND C.pid = P.pid AND P.color = 'Red'
UNION
SELECT    DISTINCT S.sname
FROM      Suppliers S, Catalog C, Parts P
WHERE     S.sid = C.sid AND C.pid = P.pid AND P.color = 'Green'
```

---

3. Design the following functions and triggers based on the same database in 1).

   a. Given a supplier id, returns the part name of the most expensive part provided by the supplier.

```
create or replace function expensive_part(integer) returns table(part_name varchar) as $$
declare
        supplier_id alias for $1;
begin
        return query
        select pname
        from parts natural join catalog
        where cost =
        (select max(cost) from catalog where sid = supplier_id);
end
$$ language 'plpgsql';
```

   b. Given a part id, output the name of all suppliers that provide the part and the corresponding cost .

```
create or replace function part_supplier(integer) returns table(supplier varchar, cost float)
as $$
declare
        part_id alias for $1;
begin
        return query
        select sname, catalog.cost
        from suppliers natural join catalog
        where pid = part_id;
end
```

c. Create a trigger to ensure that maximum number of parts provided by any supplier is 4.

```
create or replace function part_restrict() returns trigger as $$
declare
        supplier_id integer;
        part_count smallint;
begin
        for supplier_id in select sid from suppliers
                loop
                        part_count := count(pid) from catalog group by sid having sid =
supplier_id;
                        if part_count > 4 then
                                raise exception 'too many parts being supplied by supplier';
                        end if;
                end loop;
        return new;
end
$$ language 'plpgsql';

create trigger part_restrict
after insert or update on catalog
for each row execute procedure part_restrict();
```

---

4 Consider a relation with schema R(A, B, C, D) and a set of functional dependencies F: { AB --> D, BC --> A, CD-->B, AD-->C}, answer the following questions:

a) Compute $(CD)^+$ and $(BD)^+$.

$(CD)^+$={C,D}

={B,C,D} CD-->B

={A,B,C,D} BC-->A

$(BD)^+$={B,D}

b) Find all keys of R.

AB, AD, BC, CD

c) Find all super keys for R that are not keys.

ABC, ABD,ACD, BCD, ABCD

5. For each of the following relation schemas and sets of FD's:

a: R is (A, B, C, D) with FD's D --> C, C --> B, B --> A, A -->D.
b: R is (A, B, C, D) with FD's C --> A and C --> B.

1) Identify candidate keys for R
2) Indicate BCNF violations and decompose if necessary.
3) Indicate 3NF violations and decompose if necessary.

a) A^{+}= {ABCD}, B^{+}={ABCD}, C^{+}={ABCD}, D^{+}={ABCD}

Thus, candidate keys are A,B,C,D

Since the left side of all FDs are candidate keys, there is NO BCNF, 3NF violations. No decomposition necessary.

b) 1) C^{+}={ABC},
CD->ABCD. Thus, CD is candidate key.

2) C->A, C->B both violate BCNF, since the closure of C is not a candidate key.

C^{+} = {A,B,C}, so we decompose R into R1(ABC) and R2(CD)

3) C->A, C→B both violate 3NF , since C is not a superkey, and A,B are not part of a candidate key

Merge the FDs with the same left side: C->AB

Since there is only one FD, F'=F

create a relation according to C->AB R1(ABC)

because R1 doesn't contain the key CD, we need to create another relation R2(CD)

The result: R1(ABC), R2(CD)

Another way to decompose it is R1(AC), R2(BC), R3(CD)

---

6. Consider a relation R: (A,B,C,D,E,F) and the FDs: BC-->F, DF --> E, F-->DE,. Consider the following decomposition of R: R1(A,B,C,D). R2(B,C,E,F), Check if this decomposition is lossless-join.

Let R be a relation and F be a set of FDs that hold over R. The decomposition of R into relations and attribute sets R1 and R2 is lossless if and only if F+ contains either the FD $R1 \cap R2 \rightarrow R1$ or the FD $R1 \cap R2 \rightarrow R2$

Thus, R1(A,B,C,D) R2(B,C,E,F) is a lossless join decomposition since, $R1 \cap R2$ is {BC}. and we can prove that BC->BCEF.

Proof:

BC->F

and F->DE or (decomposition) F->E

and BC->F and F->E => BC->E

BC->E and BC->F => AB->EF or AB->ABEF

---

7. Show that AB --> C is in the closure of {AB -->D, DE --> C, B-->E}

fd1 fd2 fd3

1) ABE-->DE (fd1 and augmentation) fd4
2) ABE-->C(fd2, fd4 and transitivity)
3) AB-->C (fd2 and fd3)