Database Exercises

Homework 1

2.9 Consider the bank database of Figure 2.15

branch(*branch_name*, *branch_city*, *assets*)
customer (*customer_name*, *customer_street*, *customer_city*)
loan (*loan_number*, *branch_name*, *amount*)
borrower (*customer_name*, *loan_number*)
account (*account_number*, *branch_name*, *balance*)
depositor (*customer_name*, *account_number*)

Figure 2.15

a. What are the appropriate primary keys?

branch(*branch name*, *branch city*, *assets*)
customer (*customer name*, *customer street*, *customer city*)
loan (*loan number*, *branch name*, *amount*)
borrower (*customer name*, *loan number*)
account (*account number*, *branch name*, *balance*)
depositor (*customer name*, *account number*)

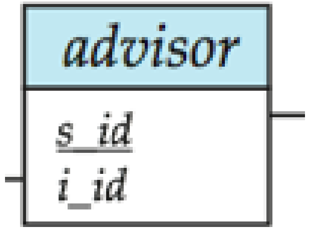
b. Given your choice of primary keys, identify appropriate foreign keys.

- For *loan*: *branch_name* referencing *branch*.
- For *borrower*: Attribute *customer_name* referencing *customer* and *loan_number* referencing *loan*
- For *account*: *branch_name* referencing *branch*.
- For *depositor*: Attribute *customer_name* referencing *customer* and *account_number* referencing *account*

Homework 1

2.10 Consider the advisor relation shown in Figure 2.8, with s_id as the primary key of advisor. Suppose a student can have more than one advisor. Then, would s_id still be a primary key of the advisor relation? If not, what should the primary key of advisor be?

No, *s_id* would not be a primary key, since there may be two (or more) tuples for a single student, corresponding to two (or more) advisors. The primary key should then be *s_id, i_id*.



Homework 2

2.12 Consider the relational database of Figure 2.14. Give an expression in the relational algebra to express each of the following queries:

a. **Find the names of all employees who work for "First Bank Corporation".**
b. **Find the names and cities of residence of all employees who work for "First Bank Corporation".**
c. **Find the names, street address, and cities of residence of all employees who work for "First Bank Corporation" and earn more than $10,000.**

employee (*person_name*, *street*, *city*)
works (*person_name*, *company_name*, *salary*)
company (*company_name*, *city*)

a. $\Pi_{person\_name}\ (\sigma_{company\_name\ =\ \text{"First Bank Corporation"}}\ (works))$

b. $\Pi_{person\_name,\ city}\ (employee\ \bowtie\ (\sigma_{company\_name\ =\ \text{"First Bank Corporation"}}\ (works)))$

c. $\Pi_{person\_name,\ street,\ city}\ (\sigma_{(company\_name\ =\ \text{"First Bank Corporation"}\ \wedge\ salary\ >\ 10000)}\ (works\ \bowtie\ employee))$

Homework 2

2.13 Consider the bank database of Figure 2.15. Give an expression in the relational algebra for each of the following queries:

branch(*branch_name*, *branch_city*, *assets*)
customer (*customer_name*, *customer_street*, *customer_city*)
loan (*loan_number*, *branch_name*, *amount*)
borrower (*customer_name*, *loan_number*)
account (*account_number*, *branch_name*, *balance*)
depositor (*customer_name*, *account_number*)

Figure 2.15

a. Find all loan numbers with a loan value greater than $10,000.
b. Find the names of all depositors who have an account with a value greater than $6,000.
c. Find the names of all depositors who have an account with a value greater than $6,000 at the "Uptown" branch.

Homework 1

a. Find all loan numbers with a loan value greater than $10,000.
b. Find the names of all depositors who have an account with a value greater than $6,000.
c. Find the names of all depositors who have an account with a value greater than $6,000 at the "Uptown" branch.

a. $\Pi_{loan\_number}\ (\sigma_{amount>\ 10000}(loan))$

b. $\Pi_{customer\_name}\ (\sigma_{balance>\ 6000}\ (depositor \bowtie account))$

c. $\Pi_{customer\_name}\ (\sigma_{balance>\ 6000 \wedge\ branch\_name=\text{“Uptown”}}\ (depositor \bowtie account))$

Homework 3

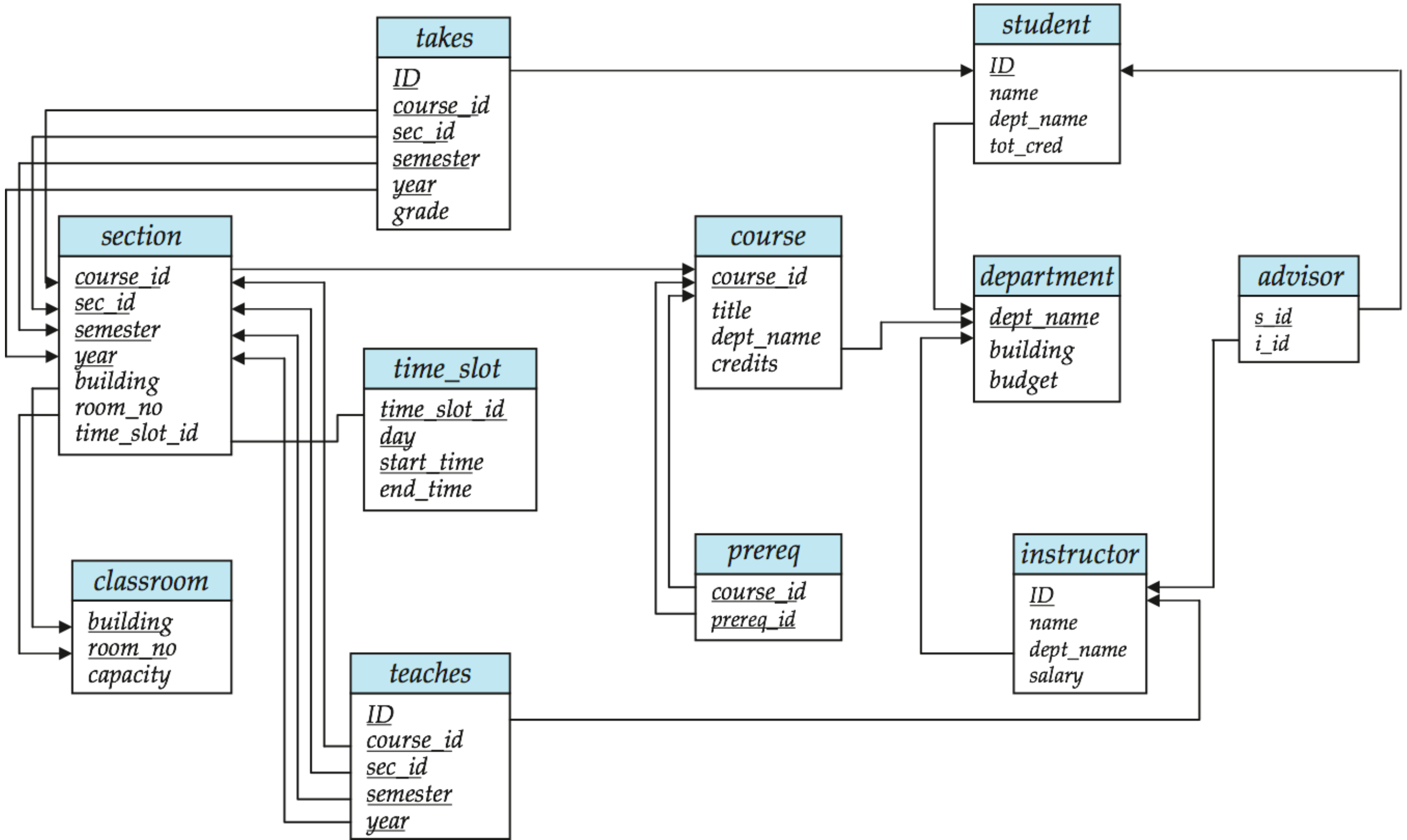
6.10 Write the following queries in relational algebra, using the university schema.

a. Find the names of all students who have taken at least one Comp. Sci. course.

b. Find the IDs and names of all students who have not taken any course offering before Spring 2009.

c. For each department, find the maximum salary of instructors in that department. You may assume that every department has at least one instructor.

d. Find the lowest, across all departments, of the per-department maximum salary computed by the preceding query



Homework 3

a. $\Pi_{name}\ (student \bowtie takes \bowtie \Pi_{course\_id}(\sigma_{dept\_name\ =\ 'Comp.Sci.'}(course)))$
Note that if we join *student*, *takes*, and *course*, only students from the Comp. Sci. department would be present in the result; students from other departments would be eliminated even if they had taken a Comp. Sci. course since the attribute *dept_name* appears in both *student* and *course*.

b. $\Pi_{ID,name}\ (student)\ -\ \Pi_{ID,name}\ (\sigma_{year<2009}(student \bowtie takes)$ Note that Spring is the first semester of the year, so we do not need to perform a comparison on *semester*.

c. ${}_{dept\_name}\mathcal{G}_{\mathbf{max}(salary)}(instructor)$

d. $\mathcal{G}_{\mathbf{min}(maxsal)}({}_{dept\_name}\mathcal{G}_{\mathbf{max}(salary)\ \mathbf{as}\ maxsal}(instructor))$

Homework 3

6.11 Consider the relational database of Figure 6.22, where the primary keys are underlined. Give an expression in the relational algebra to express each of the following queries:

employee (<u>*person_name*</u>, *street*, *city*)
works (<u>*person_name*</u>, <u>*company_name*</u>, *salary*)
company (<u>*company_name*</u>, *city*)
manages (<u>*person_name*</u>, *manager_name*)

Figure 6.22

a. **Find the names of all employees who work for "First Bank Corporation".**
b. **Find the names and cities of residence of all employees who work for "First Bank Corporation".**
c. **Find the names, street addresses, and cities of residence of all employees who work for "First Bank Corporation" and earn more than $10,000.**
d. **Find the names of all employees in this database who live in the same city as the company for which they work.**
e. **Assume the companies may be located in several cities. Find all companies located in every city in which "Small Bank Corporation" is located.**

Homework 3

a. $\Pi_{person\_name}\ (\sigma_{company\_name\ =\ \text{“First Bank Corporation”}}\ (works))$

b. $\Pi_{person\_name,\ city}\ (employee\ \bowtie$
$(\sigma_{company\_name\ =\ \text{“First Bank Corporation”}}\ (works)))$

c. $\Pi_{person\_name,\ street,\ city}$
$(\sigma_{(company\_name\ =\ \text{“First Bank Corporation”}\ \wedge\ salary\ >\ 10000)}$
$works \bowtie employee)$

d. $\Pi_{person\_name}\ (employee\ \bowtie\ works\ \bowtie\ company)$

e. Note: Small Bank Corporation will be included in each answer.
$\Pi_{company\_name}\ (company\ \div$
$(\Pi_{city}\ (\sigma_{company\_name\ =\ \text{“Small Bank Corporation”}}\ (company))))$

Homework 4

3.9 Consider the employee database of Figure 3.20, where the primary keys are underlined. Give an expression in SQL for each of the following queries.

employee (<u>*employee_name*</u>, *street*, *city*)
works (<u>*employee_name*</u>, *company name*, *salary*)
company (<u>*company_name*</u>, *city*)
manages (<u>*employee_name*</u>, *manager name*)

Figure 3.20. Employee database.

a. Find the names and cities of residence of all employees who work for First Bank Corporation.

select *e.employee_name, city*
from *employee e, works w*
where *w.company_name* = 'First Bank Corporation' **and**
w.employee_name = *e.employee_name*

Homework 4

b. Find the names, street address, and cities of residence of all employees who work for First Bank Corporation and earn more than $10,000.

employee (<u>*employee_name*</u>, *street*, *city*)
works (<u>*employee_name*</u>, *company_name*, *salary*)
company (<u>*company_name*</u>, *city*)
manages (<u>*employee_name*</u>, *manager_name*)

```
select *
from employee
where employee_name in
   (select employee_name
   from works
   where company_name = 'First Bank Corporation' and salary > 10000)
```

```
select employee_name, streat, city
from employee natural join works
where company_name = 'First Bank Corporation' and salary > 10000
```

Homework 4

c. Find all employees in the database who do not work for First Bank Corporation.

select *employee_name*
from *works*
where *company_name* ≠ 'First Bank Corporation'

employee (*employee_name*, *street*, *city*)
works (*employee_name*, *company_name*, *salary*)
company (*company_name*, *city*)
manages (*employee_name*, *manager_name*)

If one allows people to appear in the database (e.g. in *employee*) but not appear in *works*, or if people may have jobs with more than one company, the solution is slightly more complicated.

select *employee_name*
from *employee*
where *employee_name* **not in**
  (**select** *employee_name*
  from *works*
  where *company_name* = 'First Bank Corporation')

(**select** *employee_name*
from *employee*)
except
(**select** *employee_name*
from *works*
where *company_name* = 'First Bank Corporation')

Homework 4

d. Find all employees in the database who earn more than each employee of Small Bank Corporation.

employee (*employee_name*, *street*, *city*)
works (*employee_name*, *company_name*, *salary*)
company (*company_name*, *city*)
manages (*employee_name*, *manager_name*)

```
select employee name
from works
where salary > all
(select salary
from works
where company name = 'Small Bank Corporation')
```

```
select employee name
from works
where salary > (select max(salary)
                from works
                where company name = 'Small Bank Corporation');
```

Homework 4

e. Assume that the companies may be located in several cities. Find all companies located in every city in which Small Bank Corporation is located.

```
select S.company_name
from company S
where not exists ((select city
        from company
        where company_name
            = 'Small Bank Corporation')
    except
        (select city
        from company T
        where S.company_name = T.company_name))
```

employee (<u>*employee_name*</u>, *street*, *city*)
works (<u>*employee_name*</u>, *company_name*, *salary*)
company (<u>*company_name*</u>, <u>*city*</u>)
manages (<u>*employee_name*</u>, *manager_name*)

Homework 4

f. Find the company that has the most employees.

employee (*employee_name*, *street*, *city*)
works (*employee_name*, *company_name*, *salary*)
company (*company_name*, *city*)
manages (*employee_name*, *manager_name*)

```
with company_emp_num as
(select company_name, count(distinct employee_nmae) as num
from works
group by company_name)
select company_name
from company_emp_num
where num  =  (select max(num)
               from company_emp_num ) ;
```

Homework 4

g. Find those companies whose employees earn a higher salary, on average, than the average salary at First Bank Corporation.

employee (*employee_name*, *street*, *city*)
works (*employee_name*, *company_name*, *salary*)
company (*company_name*, *city*)
manages (*employee_name*, *manager_name*)

```
select company_name
from works
group by company_name
having avg (salary) > (select avg (salary)
                       from works
                       where company_name = 'First Bank Corporation')
```

```
with company_avg_salary as
(select company_name, avg(salary) as avg_salary
from works
group by company_name)
select company_name
from company_avg_salary
where avg_salary  >  (select avg(salary)
                      from works where company_name = "First Bank Corporation") ;
```

Homework 5

4.6 Complete the SQL DDL definition of the university database to include the relations student, takes, advisor, and prereq.

```
create table student
    (ID              varchar (5),
     name            varchar (20) not null,
     dept_name       varchar (20),
     tot_cred        numeric (3,0) check (tot_cred >= 0),
     primary key (ID),
     foreign key (dept_name) references department
                     on delete set null);
```

Homework 5

4.6 Complete the SQL DDL definition of the university database to include the relations student, takes, advisor, and prereq.

```
create table takes
    (ID           varchar (5),
     course_id    varchar (8),
     section_id   varchar (8),
     semester     varchar (6),
     year         numeric (4,0),
     grade        varchar (2),
     primary key (ID, course_id, section_id, semester, year),
     foreign key (course_id, section_id, semester, year) references section
                  on delete cascade,
     foreign key (ID) references student
                  on delete cascade);
```

Homework 5

4.6 Complete the SQL DDL definition of the university database to include the relations student, takes, advisor, and prereq.

```
create table advisor
    (i_id           varchar (5),
    s_id            varchar (5),
     primary key (s_ID),
     foreign key (i_ID) references instructor (ID)
                    on delete set null,
     foreign key (s_ID) references student (ID)
                    on delete cascade);
 create table prereq
    (course_id      varchar(8),
     prereq_id      varchar(8),
     primary key (course_id, prereq_id),
     foreign key (course_id) references course
                    on delete cascade,
     foreign key (prereq_id) references course);
```

Homework 5

4.8 As discussed in Section Section 4.4.7Complex Check Conditions and Assertion we expect the constraint “an instructor cannot teach sections in two different classrooms in a semester in the same time slot” to hold.

a. Write an SQL query that returns all (*instructor*, *section*) combinations that violate this constraint.

select ID, *name*, ~~*section_id*~~, *semester*, *year*, *time_slot_id*,
count(**distinct** *building*, *room_number*)
from *instructor* **natural join** *teaches* **natural join** *section*
group by (ID, *name*, ~~*section_id*~~, *semester*, *year*, *time_slot_id*)
having **count**(*building*, *room_number*) > 1

Homework 5

4.8 As discussed in Section Section 4.4.7Complex Check Conditions and Assertion we expect the constraint "an instructor cannot teach sections in two different classrooms in a semester in the same time slot" to hold.

b. Write an SQL assertion to enforce this constraint (as discussed in Section Section 4.4.7 Complex Check Conditions and Assertionssubsection.4.4.7, current generation database systems do not support such assertions, although they are part of the SQL standard).

```
create assertion check not exists
    ( select ID, name, section_id, semester, year, time_slot_id,
                count(distinct building, room_number)
      from      instructor natural join teaches natural join section
      group by (ID, name, section_id, semester, year, time_slot_id)
      having    count(building, room_number) > 1)
```

Homework 5

4.14 Show how to define a view tot_credits (year, tot_credits), giving the total number of credits taken by students in each year.

create view *tot_credits*(*year*, *tot_credits*)
as
  (**select** *year*, **sum**(*credits*)
  from *takes* **natural join** *course*
  group by *year*)

Homework 5

4.16 Referential-integrity constraints as defined in this chapter involve exactly two relations. Consider a database that includes the relations shown in Figure 4.12. Suppose that we wish to require that every name that appears in address appears in either salaried worker or hourly worker, but not necessarily in both.

salaried_worker (*name*, *office*, *phone*, *salary*)
hourly_worker (*name*, *hourly_wage*)
address (*name*, *street*, *city*)

Figure 4.12 Employee database

a. Propose a syntax for expressing such constraints.
b. Discuss the actions that the system must take to enforce a constraint of this form.

a. **foreign key** (*name*) **references** *salaried worker* **or** *hourly worker*

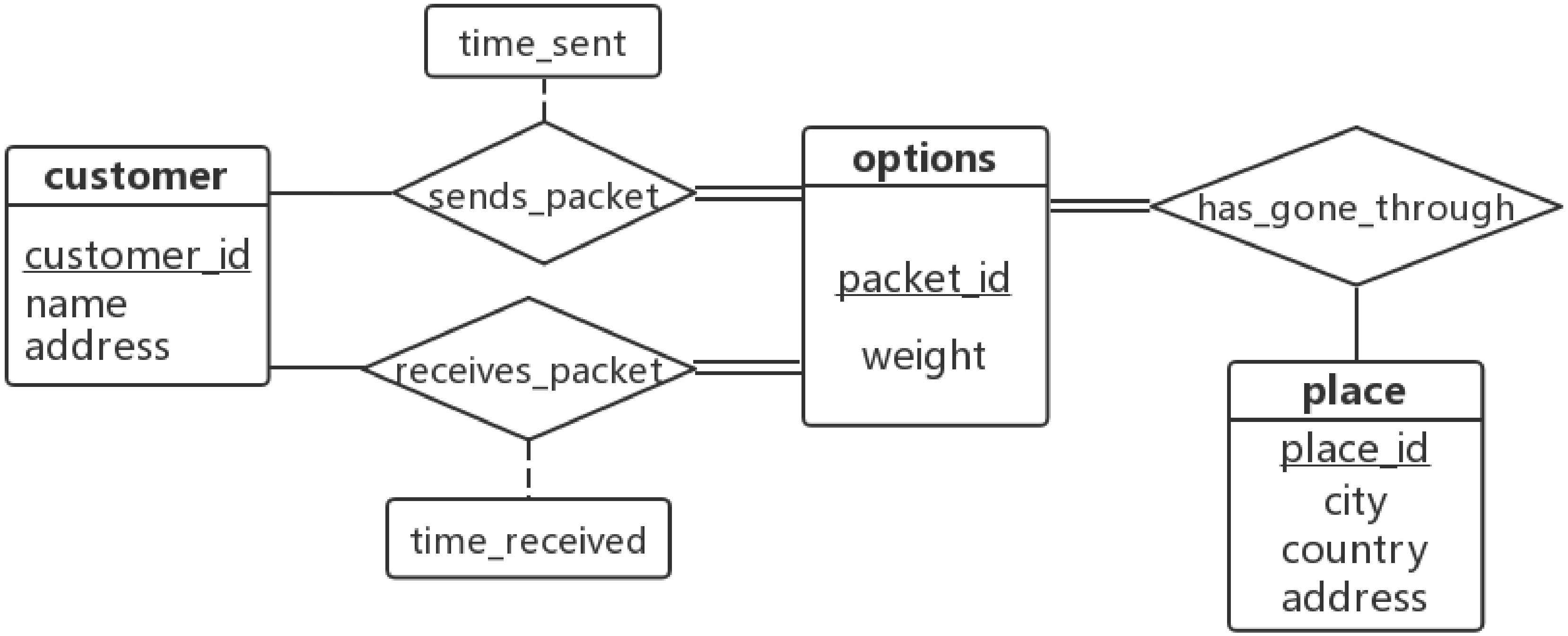
b. To enforce this constraint, whenever a tuple is inserted into the *address* relation, a lookup on the *name* value must be made on the *salaried worker* relation and (if that lookup failed) on the *hourly worker* relation (or vice-versa).

Homework 6

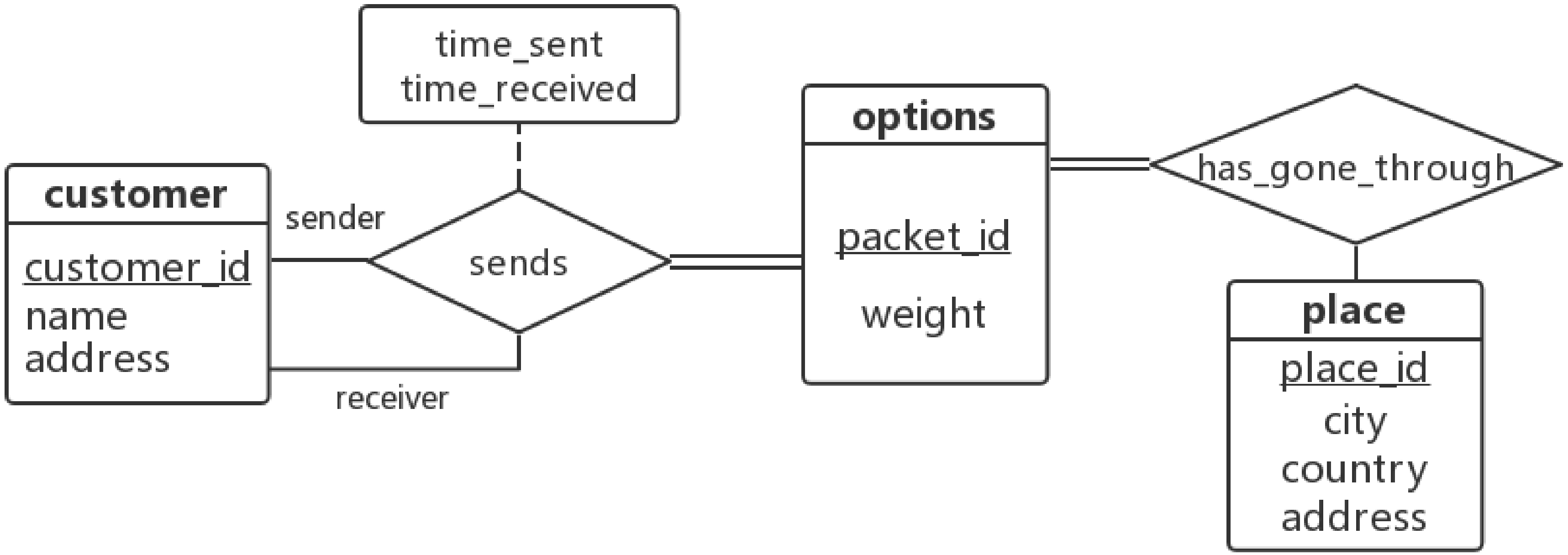
7.22 Design a database for a world-wide package delivery company (e.g., DHL or FedEX). The database must be able to keep track of customers (who ship items) and customers (who receive items); some customers may do both. Each package must be identifiable and trackable, so the database must be able to store the location of the package and its history of locations. Locations include trucks, planes, airports, and warehouses.

Your design should include an E-R diagram, a set of relational schemas, and a list of constraints, including primary-key and foreign-key constraints.

Homework 6



E-R diagram



Homework 6

customer(customer id,
name,
address)
packet(packet id,
weight)
place(place id,
city,
country,
address)

sends(sender id,
receiver id,
packet id,
time received,
time sent
foreign key *sender id* **references** *customer,*
foreign key *receiver id* **references** *customer,*
foreign key *packet id* **references** *packet)*
has gone through(
packet id,
place id
foreign key *packet id* **references** *packet,*
foreign key *place id* **references** *place)*

relational schemas

Homework 7

A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_1	c_1
a_2	b_1	c_3

8.2 List all functional dependencies satisfied by the relation of Figure 8.17

Answer:

- The nontrivial functional dependencies are: $A \rightarrow B$ and $C \rightarrow B$, and a dependency they logically imply: $AC \rightarrow B$.
- There are 19 trivial functional dependencies of the form $\alpha \rightarrow \beta$, where $\beta \subseteq \alpha$.
- C does not functionally determine A because the first and third tuples have the same C but different A values.
- The same tuples also show B does not functionally determine A.
- Likewise, A does not functionally determine C because the first two tuples have the same A value and different C values.
- The same tuples also show B does not functionally determine C.

Homework 7

8.6 Compute the closure of the following set F of functional dependencies for relation schema R = (A, B, C, D, E).

$$A \rightarrow BC$$
$$CD \rightarrow E$$
$$B \rightarrow D$$
$$E \rightarrow A$$

List the candidate keys for *R*.

Answer:

Starting with A → BC, we can conclude: A → B and A → C.	
Since A → B and B → D, A → D	(decomposition, transitive)
Since A → C D and C D → E, A → E	(union, decomposition, transitive)
Since A → A, we have	(reflexive)
A → ABC DE from the above steps	(union)
Since E → A, **E → ABC DE**	(transitive)
Since C D → E, **C D → ABC DE**	(transitive)
Since B → D and BC → C D, **BC →ABC DE**	(augmentative, transitive)

The candidate keys are A, BC, C D, and E.

Homework 7

8.7 Consider the following set F of functional dependencies on the relation schema r(A, B, C, D, E, F)

$$A \rightarrow BCD$$
$$BC \rightarrow DE$$
$$B \rightarrow D$$
$$D \rightarrow A$$

a. Compute B^+.
b. Prove (using Armstrong's axioms) that AF is a superkey.
c. Compute a canonical cover for the above set of functional dependencies F ; give each step of your derivation with an explanation.
d. Give a 3NF decomposition of r based on the canonical cover.
e. Give a BCNF decomposition of r using the original set of functional dependencies.
f. Can you get the same BCNF decomposition of r as above, using the canonical cover?

Homework 7

a. Compute B^+.

$B \rightarrow BD$	(third dependency)
$BD \rightarrow ABD$	(fourth dependency)
$ABD \rightarrow ABCD$	(first dependency)
$ABCD \rightarrow ABCDE$	(second dependency)

Thus, $B^+ = ABCDE$

b. Prove (using Armstrong's axioms) that AF is a superkey.

$A \rightarrow BCD$	(Given)
$A \rightarrow ABCD$	(Augmentation with A)
$BC \rightarrow DE$	(Given)
$ABCD \rightarrow ABCDE$	(Augmentation with ABCD)
$A \rightarrow ABCDE$	(Transitivity)
$AF \rightarrow ABCDEF$	(Augmentation with F)

Homework 7

c. Compute a canonical cover for the above set of functional dependencies F; give each step of your derivation with an explanation.

$$A \rightarrow BCD$$
$$BC \rightarrow DE$$
$$B \rightarrow D$$
$$D \rightarrow A$$

Answer:

- We see that D is extraneous in dep. 1 and 2, because of dep. 3. Removing these two, we get the new set of rules

$$A \rightarrow BC$$
$$BC \rightarrow E$$
$$B \rightarrow D$$
$$D \rightarrow A$$

- Now notice that B_+ is $ABCDE$, and in particular, the FD $B \rightarrow E$ can be determined from this set.
  Thus, the attribute C is extraneous in the third dependency.
  Removing this attribute, and combining with the third FD, we get the final canonical cover as :

$$A \rightarrow BC$$
$$B \rightarrow DE$$
$$D \rightarrow A$$

Homework 7

d. Give a 3NF decomposition of r based on the canonical cover.

Answer:

- We see that there is no FD in the canonical cover such that the set of attributes is a subset of any other FD in the canonical cover. Thus, each each FD gives rise to its own relation, giving

$$r_1(A, B, C)$$
$$r_2(B, D, E)$$
$$r_3(D, A)$$

Now the attribute F is not dependent on any attribute. Thus, it must be a part of every superkey. Also, none of the relations in the above schema have F, and hence, none of them have a superkey. Thus, we need to add a new relation with a superkey.

$$r_4(A, F)$$

e. Give a BCNF decomposition of r using the original set of functional dependencies.

Answer:

We start with

$r(A, B, C, D, E, F)$

We see that the relation is not in BCNF because of the first FD.

Hence, we decompose it accordingly to get

$r_1(A, B, C, D) \quad r_2(A, E, F)$

Now we notice that $A \rightarrow E$ is an FD in F^+, and causes r_2 to violate BCNF.

Once again, decomposing $r2$ gives

$r_1(A, B, C, D) \quad r_2(A, F) \quad r_3(A, E)$

This schema is now in BCNF.

Homework 7

f. Can you get the same BCNF decomposition of r as above, using the canonical cover?

Answer:

If we use the functional dependencies in the preceding canonical cover directly, we cannot get the above decomposition.

However, we can infer the original dependencies from the canonical cover, and if we use those for BCNF decomposition, we would be able to get the same decomposition.

Homework 8

10.9 Give an example of a relational-algebra expression and a query-processing strategy in each of the following situations:

- **a. MRU is preferable to LRU**
- **b. LRU is preferable to MRU**

Answer:

a. $R_1 \bowtie R_2$ is computed by using a **nested-loop** processing strategy where each tuple in R_2 must be compared to each block in R_1. After the first tuple of R_2 is processed, the next needed block is the first one in R_1. However, since it is the least recently used, the LRU buffer management strategy would replace that block if a new block was needed by the system.

b. $R_1 \bowtie R_2$ is computed by **sorting** the relations by join values and then comparing the values by proceeding through the relations. Due to duplicate join values, it may be necessary to "back-up" in one of the relations. This "backing-up" could cross a block boundary into the most recently used block, which would have been replaced by a system using MRU buffer management, if a new block was needed.

Homework 8

11.3 Construct a B⁺-tree for the following set of key values:

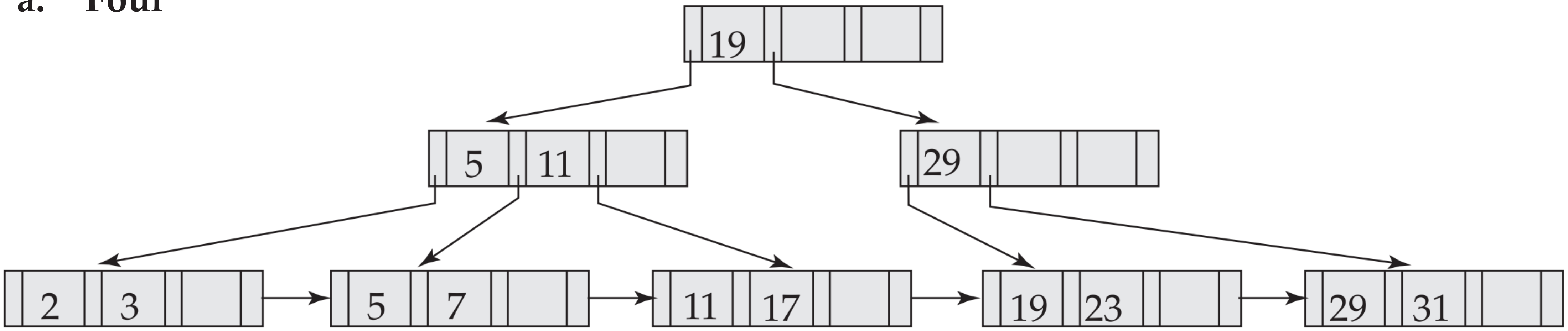
(2, 3, 5, 7, 11, 17, 19, 23, 29, 31)

Assume that the B⁺-tree is initially empty and values are added in ascending order. Construct for the cases where the number of pointers that will fit in on node is as follows:

a. Four
b. Six
c. Eight

Answer:

a. Four



Homework 8

11.3 Construct a B⁺-tree for the following set of key values:

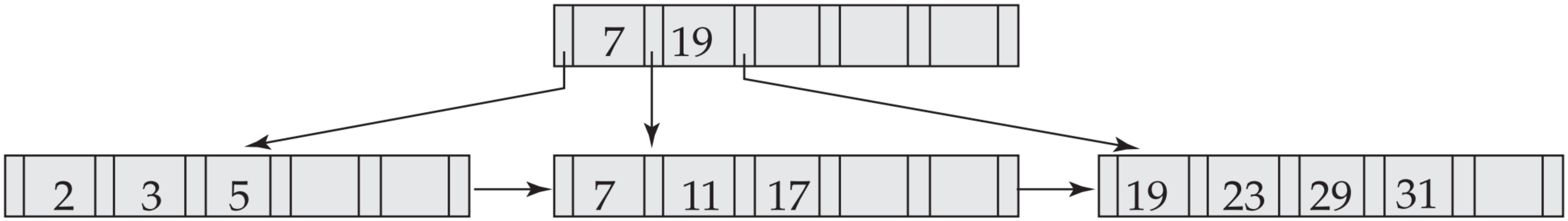
(2, 3, 5, 7, 11, 17, 19, 23, 29, 31)

Assume that the B⁺-tree is initially empty and values are added in ascending order.
Construct for the cases where the number of pointers that will fit in on node is as follows:

a. Four
b. Six
c. Eight

Answer:

b. Six



Homework 8

11.3 Construct a B⁺-tree for the following set of key values:

(2, 3, 5, 7, 11, 17, 19, 23, 29, 31)

Assume that the B⁺-tree is initially empty and values are added in ascending order. Construct for the cases where the number of pointers that will fit in on node is as follows:

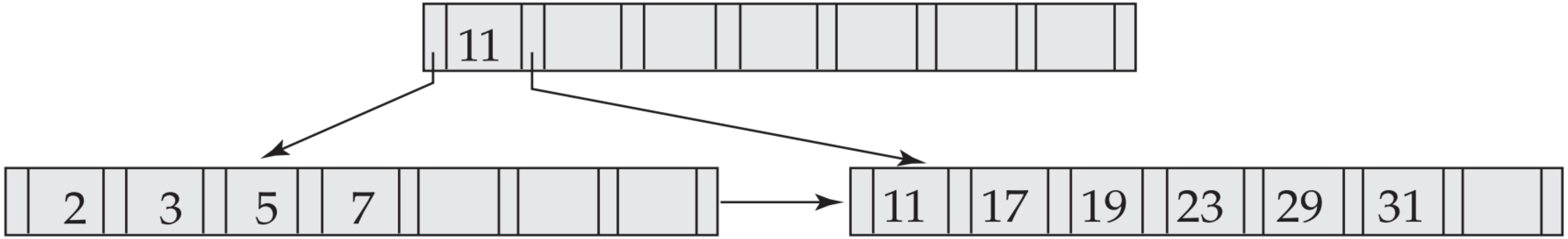
a. Four

b. Six

c. Eight

Answer:

c. Eight



Homework 9

12.3 Let relations $r_1(A, B, C)$ and $r_2(C, D, E)$ have the following properties: r_1 has 20,000 tuples, r_2 has 45,000 tuples, 25 tuples of r_1 fit on one block, and 30 tuples of r_2 fit on one block. Estimate the number of block transfers and seeks required, using each of the following join strategies for $r_1 \bowtie r_2$:

- **a. Nested-loop join.**
- **b. Block nested-loop join.**
- **c. Merge join.**
- **d. Hash join.**

Answer:

r_1 needs 800 blocks, and r_2 needs 1500 blocks. Let us assume M pages of memory. If $M > 800$, the join can easily be done in 1500 + 800 disk accesses, using even plain nested-loop join. So we consider only the worst case.

a. Nested-loop join:

Using r_1 as the outer relation we need $20000 * 1500 + 800 = 30000800$ block transfers, $20000 + 800 = 20800$ seeks

if r_2 is the outer relation we need $45000 * 800 + 1500 = 36001500$ block transfers, and 46500 seeks

Homework 9

Answer:

b. Block nested-loop join:

If r_1 is the outer relation, we need $800 * 1500 + 800$ block transfers, 800+800 seeks;
if r_2 is the outer relation we need $1500 * 800 + 1500$ block transfers. $1500 + 1500$ seeks

c. Merge-join:

Assuming that r_1 and r_2 are not initially sorted on the join key, the total sorting cost inclusive of the output is

$B_s = 1500(2\lceil log_{M-1}(1500/M)\rceil + 2) + 800(2\lceil log_{M-1}(800/M)\rceil + 2)$ disk accesses.

Assuming all tuples with the same value for the join attributes fit in memory, the total cost is $B_s + 1500 + 800$ disk accesses.

Homework 9

13.4 Consider the relations $r_1(A, B, C)$, $r_2(C, D, E)$, and $r_3(E, F)$, with primary keys A, C, and E, respectively. Assume that r_1 has 1000 tuples, r_2 has 1500 tuples, and r_3 has 750 tuples. Estimate the size of $r_1 \bowtie r_2 \bowtie r_3$, and give an efficient strategy for computing the join.

Answer:

The relation resulting from the join of r_1, r_2, and r_3 will be the same no matter which way we join them, due to the associative and commutative properties of joins. So we will consider the size based on the strategy of $((r_1 \bowtie r_2) \bowtie r_3)$. Joining r_1 with r_2 will yield a relation of at most 1000 tuples, since C is a key for r_2. Likewise, joining that result with r_3 will yield a relation of at most 1000 tuples because E is a key for r_3. Therefore the final relation will have at most 1000 tuples.

An efficient strategy for computing this join would be to create an index on attribute C for relation r_2 and on E for r_3. Then for each tuple in r_1, we do the following:

a. Use the index for r_2 to look up at most one tuple which matches the C value of r_1.
b. Use the created index on E to look up in r_3 at most one tuple which matches the unique value for E in r_2.

Homework 10

14.15 Consider the following two transactions:

$$
\begin{aligned}
T_{13}\text{: } & \mathsf{read}(A); \\
& \mathsf{read}(B); \\
& \textbf{if } A = 0 \textbf{ then } B := B + 1; \\
& \mathsf{write}(B). \\
T_{14}\text{: } & \mathsf{read}(B); \\
& \mathsf{read}(A); \\
& \textbf{if } B = 0 \textbf{ then } A := A + 1; \\
& \mathsf{write}(A).
\end{aligned}
$$

Let the consistency requirement be $A = 0 \vee B = 0$, with $A = B = 0$ the initial values.

a. Show that every serial execution involving these two transactions preserves the consistency of the database.

b. Show a concurrent execution of T_{13} and T_{14} that produces a non-serializable schedule.

c. Is there a concurrent execution of T_{13} and T_{14} that produces a serializable schedule?

Homework 10

T_{13}: read(A);
read(B);
if $A = 0$ **then** $B := B + 1$;
write(B).

T_{14}: read(B);
read(A);
if $B = 0$ **then** $A := A + 1$;
write(A).

Answer:

a. There are two possible executions: $T_{13}\,T_{14}$ and $T_{14}\,T_{13}$.

Case 1:

	A	B
initially	0	0
after T_{13}	0	1
after T_{14}	0	1

Consistency met: $A = 0 \vee B = 0 \equiv T \vee F = T$

Case 2:

	A	B
initially	0	0
after T_{14}	1	0
after T_{13}	1	0

Consistency met: $A = 0 \vee B = 0 \equiv F \vee T = T$

Homework 10

T_{13}: read(A);
read(B);
if $A = 0$ **then** $B := B + 1$;
write(B).

T_{14}: read(B);
read(A);
if $B = 0$ **then** $A := A + 1$;
write(A).

Answer:

b. Any interleaving of T_{13} and T_{14} results in a non-serializable schedule.

T_{13}	T_{14}
read(A)	
	read(B)
	read(A)
read(B)	
if $A = 0$ **then** $B = B + 1$	
	if $B = 0$ **then** $A = A + 1$
	write(A)
write(B)	

Homework 10

T_{13}: read(A);
read(B);
if $A = 0$ **then** $B := B + 1$;
write(B).
T_{14}: read(B);
read(A);
if $B = 0$ **then** $A := A + 1$;
write(A).

Answer:

c. Is there a concurrent execution of T_{13} and T_{14} that produces a serializable schedule?

There is no parallel execution resulting in a serializable schedule.

From part a. we know that a serializable schedule results in $A = 0 \vee B = 0$.

Suppose we start with T_{13} **read**(A). Then when the schedule ends, no matter when we run the steps of T_2, $B = 1$.

Now suppose we start executing T_{14} prior to completion of T_{13}.

Then T_2 **read**(B) will give B a value of 0. So when T_2 completes, $A= 1$.

Thus $B = 1 \wedge A = 1 \rightarrow \neg (A = 0 \vee B = 0)$.

Similarly for starting with T_{14} **read**(B).

Homework 10

15.2 Consider the following two transactions

T_{34}: read(A);
read(B);
if $A = 0$ **then** $B := B + 1$;
write(B).

T_{35}: read(B);
read(A);
if $B = 0$ **then** $A := A + 1$;
write(A).

Add lock and unlock instructions to transactions T_{31} and T_{32}, so that they observe the two-phase locking protocol. Can the execution of these transactions result in a deadlock?

Homework 10

T_{34}: read(A);
read(B);
if $A = 0$ **then** $B := B + 1$;
write(B).

T_{35}: read(B);
read(A);
if $B = 0$ **then** $A := A + 1$;
write(A).

Answer:

a. Lock and unlock instructions:

T_{34}:

lock-S(A)
read(A)
lock-X(B)
read(B)
if $A = 0$
then $B := B + 1$
write(B)
unlock(A)
unlock(B)

T_{35}:

lock-S(B)
read(B)
lock-X(A)
read(A)
if $B = 0$
then $A := A + 1$
write(A)
unlock(B)
unlock(A)

b. Execution of these transactions can result in deadlock. For example, consider the following partial schedule:

T_{31}	T_{32}
lock-S (A)	
	lock-S (B)
	read(B)
read(A)	
lock-X (B)	
	lock-X (A)

The transactions are now deadlocked.