

**EXAMINATIONS — 2005
MID-YEAR****COMP 302
Database Systems****Time allowed:** 3 Hours

Instructions: Answer all questions.
Make sure that your answers are clear and to the point.
Write your answers in the spaces provided.
Calculators and printed foreign language dictionaries are allowed.
No reference material is allowed.
There are 180 marks on the exam.

CONTENTS:

Question 1.	Relational Data Model	[17 marks]
Question 2.	SQL and Relational Algebra	[31 marks]
Question 3.	Enhanced Entity Relationship Data Model	[30 marks]
Question 4.	Mapping ER to Relational Data Model	[25 marks]
Question 5.	Functional Dependencies and Normalization	[35 marks]
Question 6.	Query Optimisation	[26 marks]
Question 7.	Concurrency Control and Recovery	[16 marks]

Appendices:

- A. Formulae for Computing Query Cost Estimate**
- B. SQL Reference**

Question 1. Relational Data Model

[17 marks]

a) [3 marks] Define the structure of a relational database schema.

ANSWER

b) [3 marks] What is the difference between a relation and a relation schema, and what is their relationship?

ANSWER

c) [3 marks] What is a relation schema key? List the properties of a relation schema key.

ANSWER

d) [5 marks] What is a foreign key? List the properties of a foreign key.

ANSWER

- e) [3 marks] Suppose r_1 and r_2 are relations over relation schemas $N_1(\{A, B\}, \{A\})$ and $N_2(\{B, C\}, \{B\})$, respectively, where A is the primary key of N_1 , and B is the primary key of N_2 . Using the relational algebra project operator (π), express the referential integrity constraint $N_1[B] \subseteq N_2[B]$, and explain its meaning.

ANSWER

Question 2. SQL and Relational Algebra**[31 marks]**

The relational database schema below is a part of a University database schema, where each student has at most one tutor and each course has at most one class representative. The schema is defined using SQL/1999 syntax.

```
CREATE DOMAIN StudIdDomain AS int CHECK (VALUE >= 30000000 AND VALUE =< 300099999);
```

```
CREATE DOMAIN CharDomain AS char(15);
```

```
CREATE DOMAIN PointDomain AS int CHECK (VALUE BETWEEN 0 AND 1000);
```

```
CREATE TABLE Student (  
  StudentId StudIdDomain PRIMARY KEY,  
  Name CharDomain NOT NULL,  
  NoOfPts PointDomain NOT NULL,  
  Tutor StudIdDomain REFERENCES Student(StudentId)  
);
```

```
CREATE TABLE Course (  
  CourseId CharDomain PRIMARY KEY,  
  CourName CharDomain,  
  ClassRep StudIdDomain REFERENCES Student(StudentId)  
);
```

```
CREATE TABLE Enrolled (  
  StudentId StudIdDomain NOT NULL REFERENCES Student,  
  CourseId CharDomain NOT NULL REFERENCES Course,  
  InTermPts PointDomain NOT NULL,  
  PRIMARY KEY (StudentID, CourseId)  
);
```

a) Write the following SQL queries on the university database given above:

- i. **[3 marks]** Retrieve the number of students that have a tutor.

ANSWER

- ii. [4 marks] Retrieve `StudentId`, `Name` and `NoOfPts` of all students whose `Name` ends with 'a', and who have `NoOfPts` greater than 100 and less than 300.

ANSWER

- iii. [5 marks] Retrieve `StudentId` and `Name` of all students who are not class representatives.

ANSWER

- iv. [6 marks] Retrieve `StudentId` and `NoOfPts` of all students who have more points than the student with `StudentId` = 007. Sort the result in descending order of `NoOfPts`.

ANSWER

b) Use relational algebra to write the queries below.

- i. [5 marks] Retrieve `StudentId`, `Name`, and `InTermPts` of all students who enrolled the course with `CourseId = 'COMP302'`.

ANSWER

- ii. [8 marks] Retrieve `StudentId` and `NoOfPts` of all students who have more points than the student with `StudentId = 007`.

Note: In SQL, if a nested query returns a single attribute and a single tuple, the query result will be treated as a single scalar value. Unlike SQL, Relational Algebra expressions always produce sets of tuples as outputs.

Hint: You may find it useful to do this query in a stepwise way

ANSWER

(Spare Page for extra working)

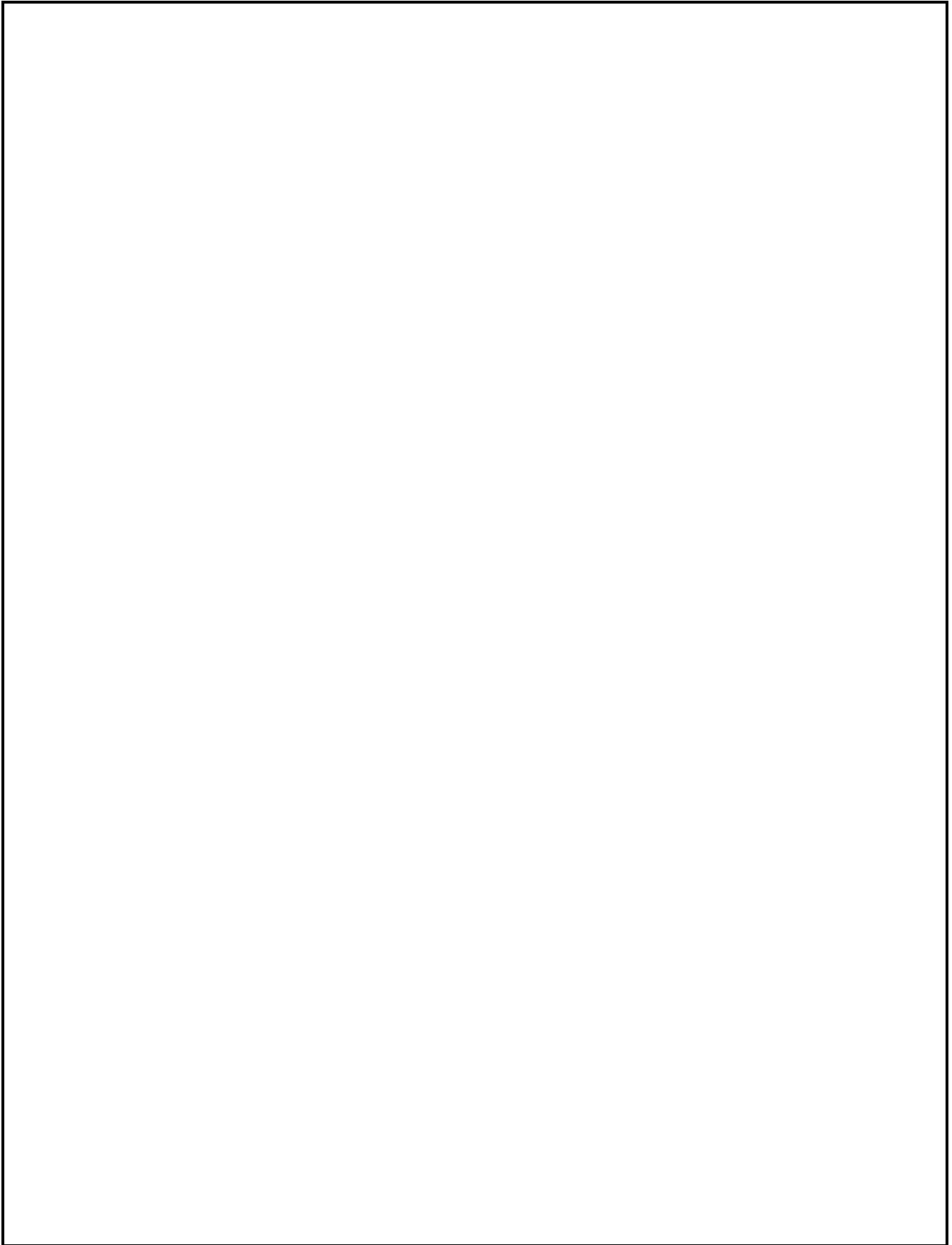
Question 3. Enhanced Entity Relationship Data Model**[30 marks]**

In this question you will be asked to draw EER diagrams of two external schemas for a university database. Recall that external schemas express views of different users of the possibly overlapping parts of the same mini world.

Use EER notation as introduced in lectures. If you use a different notation, define it clearly in your answer.

- a) **[12 marks]** Map the part of the University relational database schema defined in Question 2 of this examination to the entity relationship data model by drawing an EER diagram. State any assumption you made.

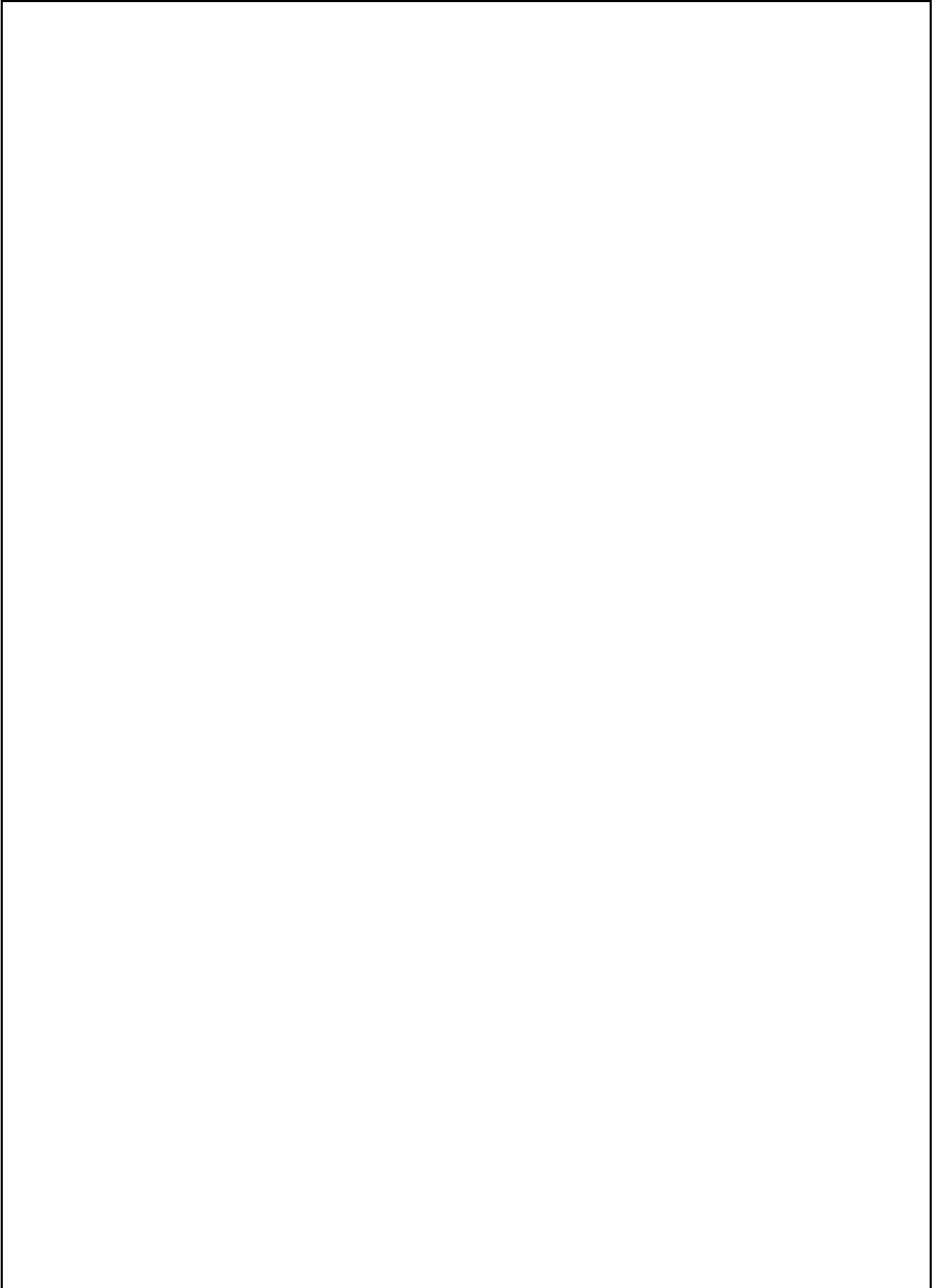
ANSWER

A large, empty rectangular box with a black border, intended for the student to write their answer to the question.

b) [18 marks] Draw an EER diagram of the conceptual schema for another part of a University database, described as follows:

- Academic staff, general staff and students are the only persons at the university.
- Each person is either an academic staff, or a general staff, or a student.
- A person is uniquely identified by a `PerId` (person's ID), and has a `Name`, and an `Address`. An `Address` is composed of `HouseNo`, `Street`, and `City`.
- A characteristic property of a student is that she/he has at least one `Major` and one `NoOfPts` (number of points) for each major.
- An academic staff has a `Position` and an `AcQual` (academic qualification).
- A general staff has a `GenPos` (general position).
- An academic staff teaches at most one course, whereas a student takes at least one course.
- A course is uniquely identified by a `CourId` (course ID), and has a `CourName` (course name).
- Each course is taught by at least one academic staff, and can be taken by many students, but there may be courses that are not taken by any students.
- Each course can use more than one textbook, but there may be courses with no textbook.
- A textbook is uniquely identified by the course which uses the book, and by an `OrdNo`. The attribute `OrdNo` is the ordinal number of the book in the list of the textbooks of a particular course. A book also has a `Title`.

ANSWER



Question 4. Mapping EER to Relational Data Model**[25 marks]**

The EER diagram on the facing page describes a part of a *Bank* database. Note that the entity type attributes are given separately, in the form

$$\textit{Entity_Type_Name} (\textit{Attribute}_1, \dots, \textit{Attribute}_n),$$

and that entity type keys are underlined.

Each bank can have multiple accounts and loans. The accounts and loans belong to customers.

a) [9 marks] Map conceptual schema *Bank* to a set of relation schemas *S*.

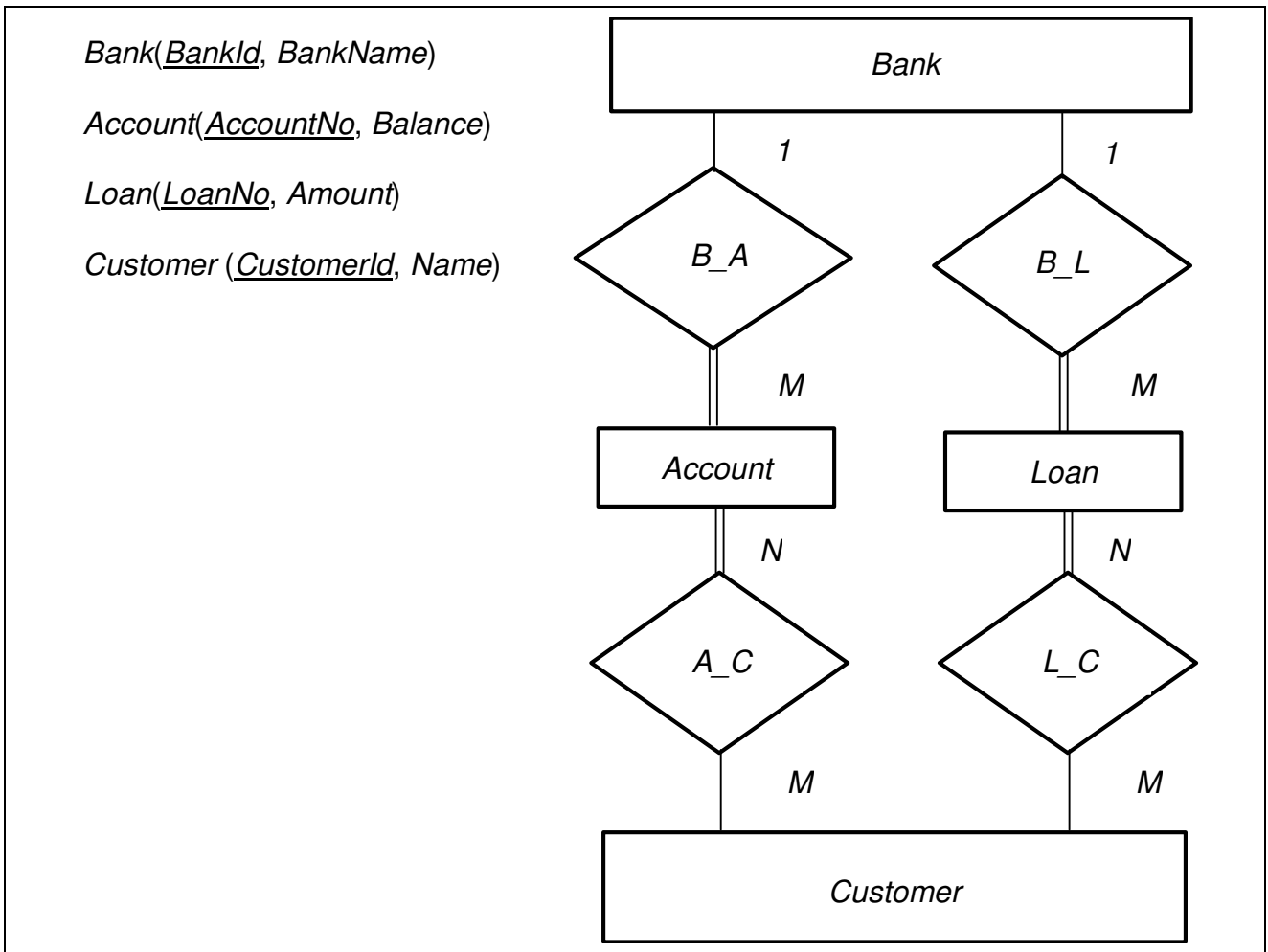
ANSWER

b) [6 marks] Define the set of functional dependencies that are implied by the relation schema keys.

ANSWER

c) [6 marks] Determine whether the set of relation schemas you produced in question 4.a) is a lossless join decomposition and explain your conclusion. If it is not a lossless join decomposition, transform it into a lossless join decomposition.

ANSWER



d) [4 marks] State the referential integrity constraints that have a NOT NULL foreign key.

ANSWER

Question 5. Functional Dependencies and Normalization**[35 marks]**

a) Express the following real world facts using functional dependencies:

- i. [3 marks] A lecturer, identified by the value of the attribute *LecturerId*, has a name (*Name*), an office (*Office*), and a phone extension number (*ExtensionNo*).

ANSWER

- ii. [3 marks] The number of students (*NoOfStud*) enrolled in a course, which is identified by the value of the attribute *CourseId*, depends on the term when the course is offered (*Term*) and the year (*Year*).

ANSWER

- iii. [2 marks] Each office (*Office*) has only one phone extension number (*ExtensionNo*) and each phone extension number belongs to at most one office.

ANSWER

b) [4 marks] Consider the following set of functional dependencies:

$$F_1 = \{AB \rightarrow C, DEL \rightarrow H, A \rightarrow C, DE \rightarrow H\}.$$

Transform the set F_1 into an equivalent set of fd's F_2 , in which each fd is left reduced (*i.e.* has no redundant attributes on its left hand side).

ANSWER

c) [6 marks] Consider the following set of left reduced functional dependencies:

$$F_3 = \{AB \rightarrow C, C \rightarrow C, AC \rightarrow D, AB \rightarrow D\}.$$

Transform the set F_3 into an equivalent, non redundant set of fd's F_4 .

ANSWER

d) Consider the following third normal form relation schema:

$$\text{Stud_Course_Lec}(\{StudentId, CourseId, LecturerId, Grade\}, \\ \{StudentId + CourseId \rightarrow Grade, StudentId + CourseId \rightarrow LecturerId, LecturerId \rightarrow CourseId\}).$$

i. [3 marks] What is the set of keys K of the relation schema *Stud_Course_Lec*?

ANSWER

ii. [2 marks] Which functional dependency violates BCNF?

ANSWER

- iii. [5 marks] If you use the functional dependency that violates BCNF to decompose the relation schema *Stud_Course_Lec* into two new BCNF relation schemes, you will lose two functional dependencies and generate a pseudotransitive one. Identify the two functional dependencies that would be lost, and the pseudotransitive functional dependency that would be generated.

ANSWER

- iv. [7 marks] Use a two step procedure to decompose the relation schema *Stud_Course_Lec* into three BCNF relation schemas, so that you lose only one of the functional dependencies, and produce no pseudotransitive ones. In the first step take one of the functional dependencies to decompose relation schema *Stud_Course_Lec* into two relation schemas (one still not BCNF) without loss of any functional dependencies. Then decompose into BCNF.

ANSWER

(Spare Page for extra working)

Question 6. Query Optimisation**[26 marks]**

The same part of the University database schema as in the question 2, is given below.

```
CREATE DOMAIN StudIdDomain AS int CHECK (VALUE >= 30000000 AND
VALUE =< 300099999);
```

```
CREATE DOMAIN CharDomain AS char(15);
```

```
CREATE DOMAIN PointDomain AS int CHECK (VALUE BETWEEN 0 AND 1000);
```

```
CREATE TABLE Student (
StudentId StudIdDomain PRIMARY KEY,
Name CharDomain NOT NULL,
NoOfPts PointDomain NOT NULL,
Tutor StudIdDomain REFERENCES Student(StudentId)
);
```

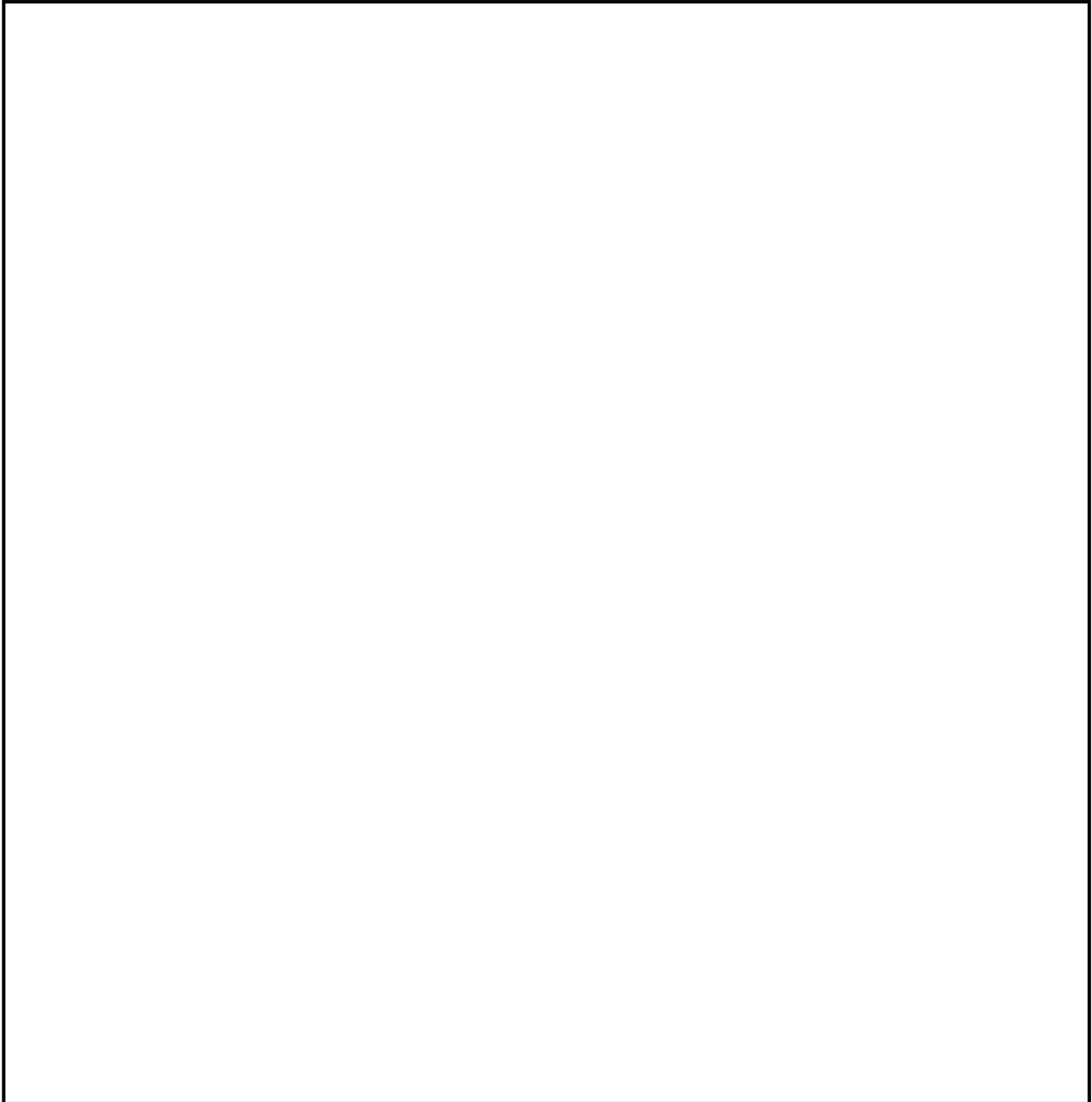
```
CREATE TABLE Course (
CourseId CharDomain PRIMARY KEY,
CourName CharDomain,
ClassRep StudIdDomain REFERENCES Student(StudentId)
);
```

```
CREATE TABLE Enrolled (
StudentId StudIdDomain NOT NULL REFERENCES Student,
CourseId CharDomain NOT NULL REFERENCES Course,
InTermPts PointDomain NOT NULL,
PRIMARY KEY (StudentID, CourseId)
);
```

a) [12 marks] Draw a heuristic optimization tree that corresponds to the SQL query:

```
SELECT StudentId, Name, InTermPts, CourName
FROM (Student s NATURAL JOIN Enrolled NATURAL JOIN Course)
WHERE InTermPts > 300 AND CourseId = "COMP302";
```

ANSWER

A large, empty rectangular box with a black border, intended for the student to write their answer to the question.

b) [14 marks] Assume:

- The `Student` relation contains data about $r = 20000$ students,
- The size of a `Student` tuple is $L = 23$ bytes,
- The block size is $B = 500$ bytes,
- The intermediate results of the query evaluation are materialized,
- The final result of the query is materialized,
- The size of each intermediate or final result block should not exceed 500 bytes,
- There is a buffer pool of 4000 bytes provided for query processing in main memory.

Draw the heuristic optimization tree and calculate the lowest execution cost of the query

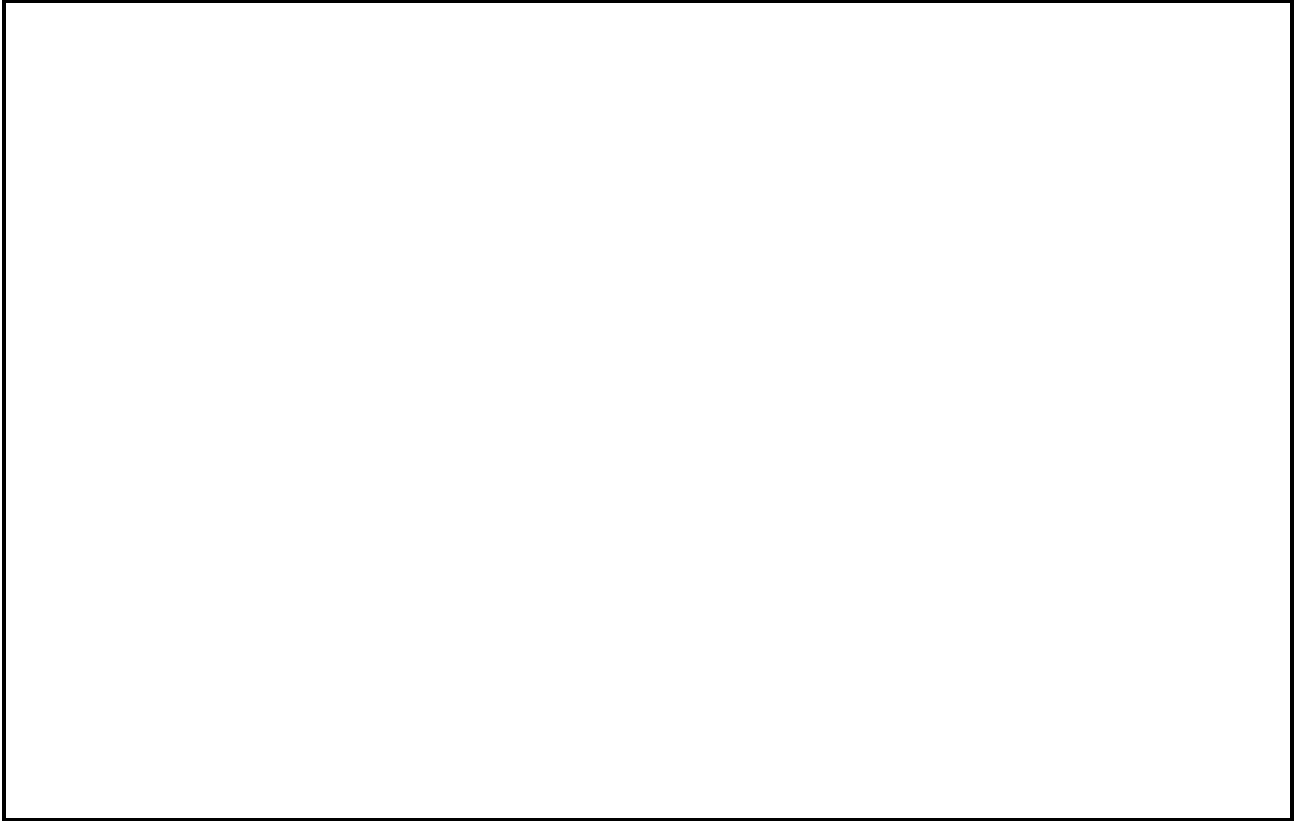
```
SELECT max(NoOfPts) FROM Student;
```

NOTE:

Some of the formulae you may need when computing the estimated query costs are given at the end of this exam paper.

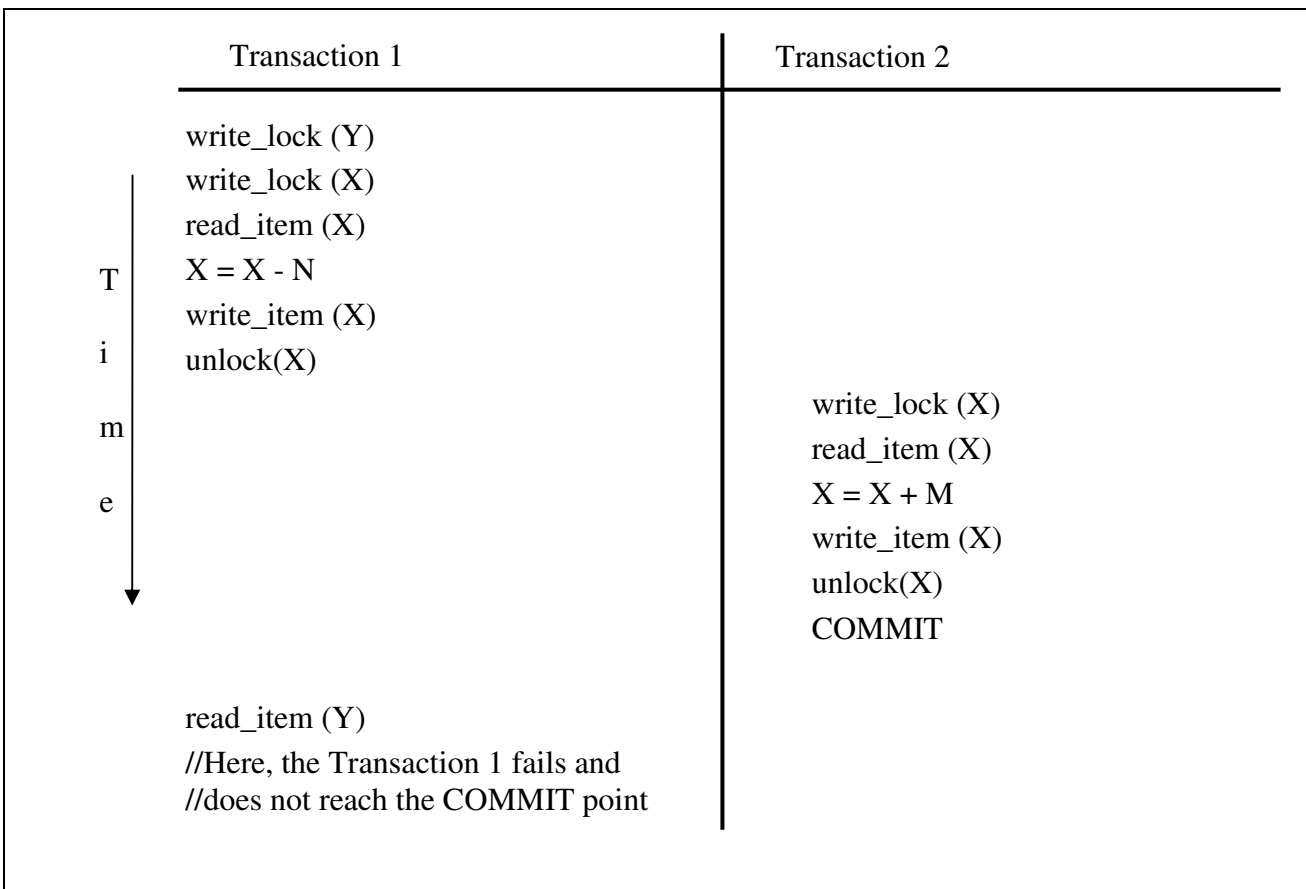
ANSWER

ANSWER

A large, empty rectangular box with a black border, intended for the student to write their answer to the question.

Question 7. Concurrency Control and Recovery**[16 marks]**

a) Consider the concurrent execution of two transactions in Figure below.

**Figure 7.1**

(i) [2 marks] What is the name of the transaction anomaly that occurred?

ANSWER

(ii) [5 marks] Why does the transaction anomaly in Figure 7.1 happen, and what is its consequence?

ANSWER

(iii) [3 marks] List the JDBC instructions we use to avoid transaction anomalies like one in Figure 7.1. In your answer, use the proper order of these instructions.

ANSWER

b) Consider the diagram in Figure 7.2, and suppose the DBMS applies the immediate update recovery technique. The immediate update algorithm used here allows a transaction to commit before all of its updates are written in the database. In Figure 7.2, T_i stands for transaction i , where $i = 1, 2, \dots, 5$.

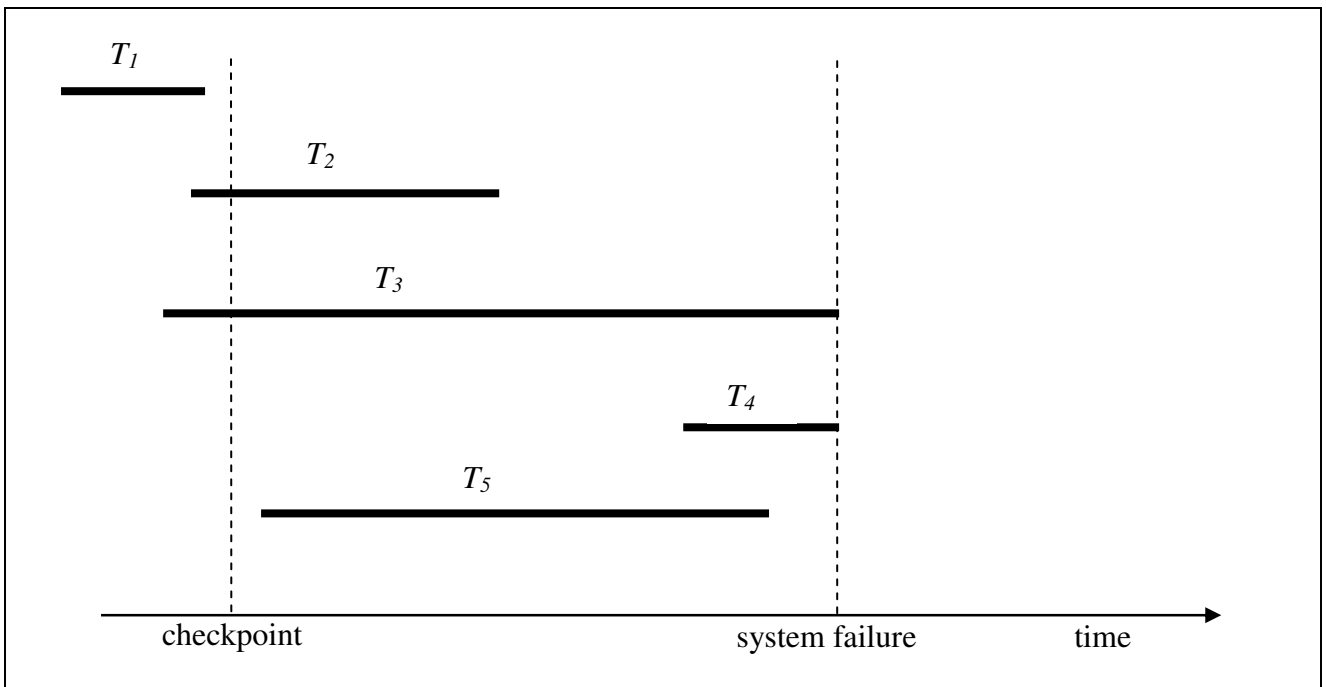


Figure 7.2

i. [3 marks] Which transactions will the DBMS have to REDO after the system failure?

ANSWER

ii. [3 marks] Which transactions will the DBMS have to UNDO after the system failure?

ANSWER

Formulae for Computing A Query Cost Estimate

Blocking factor: $f = \lfloor B / L \rfloor$

Number of blocks: $b = \lceil r / f \rceil$

Selection cardinality of the attribute A : $s(A) = r / d(A)$, where $d(A)$ is the number of different A values

Number of buffers $n = \lfloor K / B \rfloor$, where K is the size of the buffer pool

$C(\text{project}) = b_1 + b_2$

$C(\text{project_distinct}) = b_1 + b_2 + 2b_2(1 + \lceil \log_m b_2 - 1 \rceil) + b_2 + b_3$, $m = n - 1$

$C(\text{select_linear}) = b_1 + \lceil s(Y) / f \rceil$, $s(Y)$ is the selection cardinality of the search argument Y

$C(\text{select_index}) = x + s(Y) + \lceil s(Y) / f \rceil$, $s(Y)$ is the selection cardinality of the indexed set of attributes Y

Costs of join algorithms

(o stands for the outer loop relation, and i stands for the inner loop relation)

$C(\text{nested_join}) = b_o + b_i \lceil b_o / (n - 2) \rceil + \lceil js * r_o * r_i / f \rceil$

$C(\text{single_join}) = b_o + r_o * f(\text{index}_i) + \lceil js * r_o * r_i / f \rceil$ (minimum of 4 buffers required)

$C(\text{sort_join}) = b_1(3 + 2 \lceil \log_m b_1 - 1 \rceil) + b_2(3 + 2 \lceil \log_m b_2 - 1 \rceil) + \lceil js * r_1 * r_2 / f \rceil$, $m = n - 1$

$C(\text{partition_join}) = 3(b_1 + b_2) + \lceil js * r_1 * r_2 / f \rceil$, $(n - 1)(n - 2) \geq b_1$, $b_1 < b_2$

$C(\text{sort}) = 2b(1 + \lceil \log_m b - 1 \rceil)$

$f(\text{index})$:

primary index: $f(\text{index}) = x + 1$

secondary index: $f(\text{index}) = x + s(Y)$

Approximate formulae for choosing the most efficient join algorithm

Mandatory condition: $b_o \ll b_i$

(o stands for the outer loop relation, and i stands for the inner loop relation)

$C(\text{single}) < C(\text{nested}) \Leftrightarrow r_o * f(\text{index}_i) < b_i \lceil b_o / (n - 2) \rceil$ // Providing $r_o * f(\text{index}_i) \gg b_o$

$C(\text{single}) < C(\text{sort}) \Leftrightarrow r_o * f(\text{index}_i) < b_i(3 + 2 \lceil \log_m b_i - 1 \rceil)$ // Providing $r_o * f(\text{index}_i) \gg b_o$

$C(\text{single}) < C(\text{partition-hash}) \Leftrightarrow r_o * f(\text{index}_i) < 3b_i$ // Providing $r_o * f(\text{index}_i) \gg b_o$

$C(\text{nested}) < C(\text{sort-merge}) \Leftrightarrow b_o < (n - 2)(3 + 2 \lceil \log_m b_i - 1 \rceil)$

$C(\text{nested}) < C(\text{partition-hash}) \Leftrightarrow \lceil b_o / (n - 2) \rceil \leq 3$

$C(\text{sort-merge}) < C(\text{partition-hash}) \Leftrightarrow \perp$

Simplified PostgreSQL documentation:

CREATE TABLE

```
CREATE TABLE table_name (  
    { column_name data_type [ DEFAULT default_expr ] [ column_constraint [, ... ] ]  
    | table_constraint } [, ... ]  
)
```

where *column_constraint* is:

```
[ CONSTRAINT constraint_name ]  
{ NOT NULL | NULL | UNIQUE | PRIMARY KEY | CHECK (expression) |  
  REFERENCES reftable [ ( refcolumn ) ] [ ON DELETE action ] [ ON UPDATE action ] }
```

table_constraint is:

```
[ CONSTRAINT constraint_name ]  
{ UNIQUE ( column_name [, ... ] ) |  
  PRIMARY KEY ( column_name [, ... ] ) |  
  CHECK ( expression ) |  
  FOREIGN KEY ( column_name [, ... ] ) REFERENCES reftable [ ( refcolumn [, ... ] ) ]  
  [ ON DELETE action ] [ ON UPDATE action ] }
```

and *action* is one of RESTRICT, CASCADE, SET NULL, or SET DEFAULT

SELECT

```
SELECT [ ALL | DISTINCT ]  
    * | expression [ AS output_name ] [, ... ]  
    [ FROM from_item [, ... ] ]  
    [ WHERE condition ]  
    [ GROUP BY expression [, ... ] ]  
    [ HAVING condition [, ... ] ]  
    [ { UNION | INTERSECT | EXCEPT } [ ALL ] select ]  
    [ ORDER BY expression [ ASC | DESC | USING operator ] [, ... ] ]  
    [ FOR UPDATE [ OF tablename [, ... ] ] ]
```

where *from_item* can be:

```
[ ONLY ] table_name [ * ] [ [ AS ] alias [ ( column_alias_list ) ] ] |  
( select ) [ AS ] alias [ ( column_alias_list ) ] |  
from_item [ NATURAL ] [ join_type ] JOIN from_item [ ON join_condition | USING ( join_column_list ) ]
```

and *join_type* can be:

```
INNER |  
LEFT [ OUTER ] |  
RIGHT [ OUTER ] |  
FULL [ OUTER ] |  
CROSS
```

For INNER (the default) and OUTER join types, exactly one of NATURAL, ON *join_condition*, or USING (*join_column_list*) must appear. For CROSS JOIN, none of these items may appear.

CREATE VIEW

```
CREATE VIEW view [ ( column name list ) ] AS SELECT query
```

Some Data Types

integer, int, smallint
character[*n*], char[*n*], character varying[*n*], varchar[*n*], varchar
numeric, numeric[*precision*], numeric[*precision*, *scale*], real, double
boolean, date,

Note: [*xxx*] means *xxx* is optional, { *xxx* | *yyy* } means *xxx* or *yyy*.