QUESTION BANK FOR INSTRUCTORS

VERSION DATED: February 20, 2013

TO ACCOMPANY

Database System Concepts

Sixth Edition

Abraham Silberschatz
Yale University

Henry F. Korth
Lehigh University

S. Sudarshan
Indian Institute of Technology, Bombay

Contents

Preface

This question bank provides a set of questions for different chapters of the book Database System Concepts 6th Ed. The goal of this question bank is to provide instructors with a large set of questions that could help them in creating examinations.

Each question has associated with it a number that is a rough estimate of the number of minutes that a good student should take to answer the question (for some of the tougher questions, the time taken may be significantly higher than the associated number). To take average students into account, the numbers should be inflated by a factor of around 1.5 or 2. (Obviously the time depends greatly on the specific student population, so your mileage may vary.)

Most of these questions appeared in examinations of courses taught by us. We welcome contributions of questions, to make this question bank a larger and more valuable resource.

Our current policy is to not publish answers along with the questions, to limit the damage done if any part of the material gets accidentally published on the Web by anybody. This may be revised based on feedback from instructors.

NOTE*: The material in this manual may not be further distributed in any form, except by way of usage in exams/problem sets in courses for which the Database System Concepts book is the recommended textbook.*

Avi Silberschatz, Hank Korth and S. Sudarshan.

CHAPTER 1

Introduction and Relational Model (Chapters 1 and 2)

We have not provided any questions for Chapter 1 (Introduction) and Chapter 2 (Relational Model) currently.

C H A P T E R 3

SQL (Chapters 3 and 4)

Questions on SQL covering Chapters 3 and 4 are provided here.

Exercises

3.1 Given the schema

```
item(itemid, name, category, price)
itemsale(transid, itemid, qty)
transaction(transid, custid, date)
customer(custid, name, street-addr, city)
```

where primary keys are underlined, write the following queries in SQL:

- **a.** Find the name and price of the most expensive item (if more than one item is the most expensive, print them all). . . .5
- **b.** Print the total sales (in terms of units and total price) of every item category in every customer-city. . . .5
- **c.** Find items with no sales at all to customers in Mumbai . . .5
- **d.** Find customers who bought the same quantity of the same item on subsequent dates. . . .5
- **e.** Find all customers who did not buy any item in category "Electronics". . . .5

3.2 Suppose you are given a relation $r(R, M)$, where R indicates roll number and M the marks scored.

- **a.** Write an SQL query to compute the top 5 distinct marks. . . .3
- **b.** Write an SQL query to find the "dense rank" of each student. That is, all students with the top mark get a rank of 1, those with the next highest mark get a rank of 2, and so on. Hint: split the task into parts, using the **with** clause. . . .10

3.3 Consider a system to store the marks of students on various exams, for several courses.

a. Define a view `totalmarks(course, rollno, marks)` to get total marks for students of each course, given a table: `weights(course, exam, weightage)`. Here `weightage` is a factor by which you multiply the marks for the corresponding exam. . . .4

b. Define a view `grades(course, rollno, grade)` to get the grades for students, given a relation `cutoff(course, grade, cutoff-mark)`. Assume that marks below the lowest cutoff mark get an FF grade. . . .6

3.4 I want to design a system to automate project groups signing up for project demo slots.

(Note: formal coverage of schema design is in Chapter 8, but this question can still be asked of smart students)

a. Design a relational schema for this task. You should record group members for groups. Enforce the following constraints: every student is in at most one group, each slot has at most one group, each group has at most one slot. . . .4

b. Write SQL queries to list students who are not in any group, and groups that have not yet signed up for a slot. . . .3+ 3

c. Write an SQL query to find all slots for which there is no group signed up. . . .3

3.5 Given a relation $income(name, value)$ write an SQL query to find the first 10 tuples of $income$, sorted by $value$. The query should only output these 10 tuples; assume for simplicity that the income values of different people are guaranteed to be different. (Don't worry about efficiency.) . . .7

3.6 Write an SQL query to list the name of each student and the total number of courses he/she is registered for. Assume relations $student(rollno, name)$ and $registered(rollno, course)$ are given. *Make sure that your query lists students who aren't registered for any course, with a count of 0.* Hint: use a nested subquery. . . .5

3.7 Write SQL expressions to do the following. Assume you are given two relations, student(name, rollno) and marks(rollno, exam, mark)

a. Show names of all students who have got marks in at least two exams. . . .2

b. Find the names of the students with highest total marks (summed across all exams for each student). . . .3

3.8 Given a table r with a foreign key referencing s, what is the effect of adding the clause **on delete cascade** to the foreign key declaration? . . .2

3.9 Give an example of a pair of relations with integrity constraints, and inserts into both of them such that, regardless of whichever insert is first, the integrity constraints are violated by the first insert, although they are satisfied after the second insert. (To handle such situations (among other reasons) integrity constraints are only checked at the end of transaction.) . . .5

CHAPTER 5

Advanced SQL

Exercises

5.1 The following example shows how difficult it is to do some simple things in SQL. Suppose you have a relation $transaction(accountno, seqno, amount)$ where the amount can be positive (deposit) or negative (withdrawal), and the sequence number (seqno) is unique within each account number.

Write a query to output every accountno which has a negative balance (the balance is the cumulative total from the earliest sequence number for the account). . . .8

Now write the same query using SQL with OLAP extensions. . . .3

5.2 Give a (small) example of a crosstab with summaries, and show how to represent the same example data, including summaries, as a normal relation. . . .2+ 2

5.3 Write a query in basic SQL equivalent to the following query in SQL with OLAP extensions (which generates a "data cube")

```
select r.A, r.B, sum(r.C)
from r
group by cube(r.A, r.B)
```

. . .3

5.4 SQL/OLAP

a. Suppose I have a table *grade(rollno, course, exam, marks)* giving the marks of students in different exams. Assume the only exams are named `quiz`, `midsem` and `final`. Assume each student has taken each exam for every course they are registered for. Write an SQL query to generate a crosstab with course and rollnumbers identifying rows and exam names as columns,

with each cell containing the marks for that course/student for that exam. Sort the output by rollno. You need not output summary results. . . .7

b. Extend the above query to output the crosstab with scaled marks, where the marks for each course/exam combination are multiplied by 50/avg-mark, where avg-mark is the average across all students for that course/exam combination. This scales the average mark for each exam to 50. Hint: use the with clause. . . .7

5.5 Suppose you are given a relation $parent(P, C)$, where P is the name of a parent and C the name of a child of P. Assume that names are unique, that is, no two people have the same name.

a. Write a recursive SQL query to compute all descendants of each person, and record with each descendant the level of descent (1 for child, 2 for grandchild, etc.). The program should define a view $descendant(A, D, L)$. Here, A is the name of a person D is the name of a descendant of A, and L the level of the descendant. . . .6

b. Write a JDBC program using non-recursive SQL to create a relation $descendant$ defined as above.

NOTE 1: your program must use only constant space. For example, storing a potentially large set of names in a Java hash table is not legal.

NOTE 2: Don't bother to give code for setting up or deleting connections, statement objects etc

. . .9

5.6 **a.** Give table definitions of $r(A, B)$ and $s(A, B)$, with all attributes of type integer, along with a check statement to ensure the constraint $s.B \subseteq r.A$. . . .3

b. Ensure the same constraint by specifying SQL triggers on r and s instead of check statements. Do not perform unnecessary checks. . . .7

5.7 Given the relation `person(name, sex, father, mother)`, with name as the primary key, and sex taking on values 'M' and 'F', write recursive views in SQL to find

a. $brother(x, y)$: all x, y such that y is a brother of x (x may be female). . . .3

b. $anc(x, y, h)$: all x, y, h such that y is an ancestor of x at level h. E.g. parents are at level 1, grandparents at level 2, and so on. . . .4

5.8 Suppose you are given a relation $parent(X, Y)$ indicating that X is a parent of Y. Write a recursive SQL query to find a relation $sg(X, Y, D)$ containing all pairs of people X, Y who are in the same generation, along with their degree of separation D (brothers/sisters=1, first-cousin=2, etc). . . .8

5.9 Suppose you are given a relation $connection(X, Y, Dep, Arr)$, where a tuple in the relation indicates that there is a train from X to Y, departing at time Dep and arriving at time Arr.

a. Write a recursive SQL program to find all cities reachable from Mumbai, possibly via a sequence of connections. Don't worry about departure and arrival times. . . .4

b. Modify the program to find cities reachable using a sequence of connections such that the stopover at any intermediate city is not more than 3 hours and not less than 1 hour. Don't worry about time wrapping around midnight. . . .3

5.10 Explain, using a simple example, the motivation for roles in SQL security. . . .3

5.11 Explain why you need a references privilege to create a foreign key. . . .3

5.12 What are authentication, authorization, non-repudiation, and non-duplicatability and why are they important in secure electronic transactions? How are these properties achieved when paper cheques are used? . . .3+ 1

5.13 Give a trigger definition that will result in a infinite work. . . .3

5.14 Why is a "references" privilege required to declare an attribute of a relation as a foreign key referencing another relation? . . .2

5.15 Suppose a relation r has an attribute `user-id`. Show how to use a before trigger on a relation r to ensure that the users userid (assume a global variable $uid) is either the special user id `admin` or matches the `user-id` attribute of the tuple, before allowing access to the tuple. . . .3

5.16 Give a simple example of authorization using SQL statements, which involves *roles* as well as the use of the *grant option*. . . .3

5.17 Give an example of an application requirement of authorization which cannot be specified using SQL. (This will explain why authorization is done using application code, instead of using SQL, in most applications, including your lab projects). . . .2

5.18 Suppose relations r and s have only inserts on them (no deletes/updates). Write triggers on r and s to maintain a materialized view $r \bowtie s$. Also outline what will have to be done to handle deletes (no need to give actual triggers for this case). . . .5 + 2

C H A P T E R 6

Formal Relational Query Languages

Note: Exercises mentioning QBE and Datalog may be omitted, unless you are covering QBE and Datalog using the online appendices as part of your course.

Exercises

6.1 Give a relational algebra expression to find the maximum value in relation $r(A)$, *without using aggregation operations*. . . .3

6.2 Given relation $r(A, B)$ give an expression to find all tuples with the maximum A value. . . .2

6.3 Show how to express $A \supset\!\bowtie B$ using the basic relational operations, plus natural join. . . .2

6.4 Draw the truth table for the AND operation on three-valued logic. . . .3

6.5 Give an example of a simple selection view (in relational algebra) and an update on it that cannot be executed. . . .3

6.6 Consider the following relational schema

FACULTY(empno,name,office,age)
BOOKS(isbn,title,authors,publisher)
LOAN(empno,isbn,date)

The attributes empno and isbn are the candidate keys for FACULTY and BOOKS respectively.

Write the following queries in Relational Algebra.

a. Print names of faculty members who have borrowed a book published by Addison-Wesley.

b. Print names of faculty members who have borrowed all books published by Addison-Wesley.
c. For each publisher, print the names of faculty who have borrowed more than five books of that publisher. ...8

6.7 Write extended relational algebra expressions to do the following. Assume you are given two relations, student(name, rollno) and marks(rollno, exam, mark)

a. Show names of all students who have got marks in at least two exams. ...2
b. Find the names of the students with highest total marks (summed across all exams for each student). ...3

6.8 Consider a relation marks(rollno, course, examname, marks).

a. Suppose the course DBIS has two quizzes, and you want to find the maximum of the two quiz marks for each student of the course. Assume that each student has got a mark for both the quizzes.

Write a relational algebra query **using** a join and a function "bigger(m1,m2)" to find the maximum of the two marks for each DBIS student.
b. Now answer the above query under the assumption that some students may have missed one of the quizzes (don't worry about both being missed). **NOTE**: don't use a join in this case, use an alternative technique.

...12

6.9 Let r and s be relations with the schema $R(a1, a2)$ and $S(a1, a3)$ respectively. The semijoin operation $r \ltimes s$ selects tuples from r that match with some tuple in s. The semijoin operation can be defined as $\Pi_{a1,a2}(r \bowtie s)$. (Here, $\bowtie$ is the natural join where common attributes appear only once.)

Consider the relational algebra with duplicates.

a. Define (in words) the multiset versions of Π, σ and $\bowtie$. ...5
b. Define the multiset version of $\ltimes$ as containing the same *set* of tuples in $r \ltimes s$, but each tuple has the same multiplicity as it has in r. For example if $r = \{(1, 2), (1, 2), (2, 3)\}$ and $s = \{(1, 4), (1, 5)\}$, $r \ltimes s = \{(1, 2), (1, 2)\}$, whereas $r \bowtie s = (1, 2, 4), (1, 2, 4), (1, 2, 5), (1, 2, 5)$. Define the multiset version of $\ltimes$ in terms of the multiset versions of $\Pi, \sigma, \bowtie$, and δ, where δ is the duplicate elimination operation. ...10

6.10 Given relations $r(A, B)$, and $s(A, C)$:

a. Give an expression in SQL that is equivalent to ${}_B\mathcal{G}_{sum(C)}(\Pi_{BC}(r \bowtie s))$. Don't forget that Π eliminates duplicates. ...4
b. Give a relational algebra expression equivalent to

```
select * from r where exists
     (select * from s where s.A = r.A)
```

...4

6.11 Given the schema

```
item(itemid, name, category, price)
itemsale(transid, itemid, qty)
transaction(transid, custid, date)
customer(custid, name, street-addr, city)
```

where primary keys are underlined, write the following queries in relational algebra:

- **a.** Find the name and price of the most expensive item (if more than one item is the most expensive, print them all). . . .5
- **b.** Print the total sales (in terms of units and total price) of every item category in every customer-city. . . .5
- **c.** Find items with no sales at all to customers in Mumbai . . .5
- **d.** Find customers who bought the same quantity of the same item on subsequent dates. . . .5
- **e.** Find all customers who did not buy any item in category "Electronics". . . .5

6.12 Using the schema *account(account-number, branch-name, balance)*, *depositor(name, account-number)*, and *branch(branch-name, branch-city)* write a query in tuple relational calculus to find all customers who have an account at every branch in Mumbai. . . .5

6.13 Given the schema

```
item(itemid, name, category, price)
itemsale(transid, itemid, qty)
transaction(transid, custid, date)
customer(custid, name, street-addr, city)
```

where primary keys are underlined, write the following queries in tuple and domain relational calculus

- **a.** Find items with no sales at all to customers in Mumbai . . .5
- **b.** Find customers who bought the same quantity of the same item on subsequent dates. . . .5
- **c.** Find all customers who did not buy any item in category "Electronics". . . .5

6.14 Write tuple and domain relational calculus expressions to do the following. Assume you are given two relations, student(name, rollno) and marks(rollno, exam, mark)

- **a.** Show names of all students who have got marks in at least two exams. . . .2
- **b.** Find the names of the students with highest total marks (summed across all exams for each student). . . .3

6.15 Suppose you are given relations *account(number, balance)* and *owner(name, number)*.

- **a.** Write a query to find balances of all accounts owned by Bala using
  - i. tuple relational calculus . . .2
  - ii. domain relational calculus . . .2
  - iii. QBE . . .2
- **b.** Write a QBE query to print the names of customers, with the account name and and balances of each account owned by the customer. . . .3

6.16 Suppose you are given a relation $r(R, M)$, where R indicates roll number and M the marks scored. Write queries to compute the top 5 distinct marks using

- **a.** QBE . . .3

b. Tuple relational calculus . . .3

6.17 Give a QBE expression equivalent to $\{< a > \mid \exists\, c\, (< a, c > \in\ s\ \wedge\ \exists\, b_1, b_2\, (< a, b_1 > \in\ r \wedge\ < c, b_2 > \ \in\ r\ \wedge\ b_1\ >\ b_2))\}$ . . .4

6.18 For each of the following give an example Datalog program:

a. with non-stratified negation. . . .2

b. which uses arithmetic predicates and generates infinite answers, although each iteration of the fixpoint procedure produces only 1 answer. . . .4

CHAPTER 7

Entity-Relationship Model

Exercises

7.1 Suppose we examine an instance of a relation R(A,B,C) and observe that there are no repeated A values among the tuples. Can we assume that A is a key for the relation.

7.2 Consider an entity set with candidate key K1 and a super key K2 where K2 is not a candidate key. Is it possible that K1 has more attributes than K2 ? Why or why not ?

7.3 What is wrong with an ER diagram which has an entity *department* with primary key *deptid*, and an entity *employee*, one of whose attributes is *deptid*. . . .3

7.4 Consider a system to store the marks of students on various exams, for several courses.

- **a.** Draw an ER diagram to repesent this information. Don't worry about student information other than roll number and name, and course information other than course number and name. . . .3
- **b.** Convert the above ER diagram to a set of SQL tables with appropriate primary and foreign key constraints. . . .3

7.5 Show how to express a ternary relationship R between entity sets A, B and C as 3 binary relationships, including appropriate constraints (assume there are no constraints on R). . . .4

7.6 What is the difference between disjoint and overlapping generalization? . . .2

7.7 Give a meaningful example of the need for role indicators in ER diagrams. (Don't give a diagram, just an example in words.) . . .2

7.8 Describe how to represent a ternary relationship between entity sets A, B and C using binary relationships with a new strong entity set E. Give constraints on the ER diagram to ensure that every relationship represented by the binary relationships corresponds to exactly one (not less or more) ternary relationship. . . .4

7.9 Draw an (incorrect) example ER diagram where a relationship is implicit whereas it should be explicit. . . .3

7.10 Give examples of (a) a recursive relationship and (b) total participation. (They can be separate examples and need not be realistic.) . . .3

7.11 Draw an ER diagram for a inter-collegiate contest with multiple events, recording events(name, location, date, time), student(id, name, college), and which students participate in which events. Each student in the student relation must participate in some event. . . .2

7.12 A weak entity, like course-offering, can always be converted to a strong entity by adding the primary key of its identifying entity. Why then does it make sense to use weak entities in ER diagrams. . . .2

7.13 Considering the increasing number of accidents on campus, the security section wishes to automate its accident recording section.

- **a.** Draw an ER diagram for the accident recording system. The system should track vehicles involved in accidents (license no, type and make), drivers of the vehicles, location, and all injured persons (drivers or otherwise). . . .6
- **b.** Convert the ER Diagram to SQL table definitions including applicable foreign key constraints. . . .4

7.14 We wish to develop a tool for drawing and storing ER diagrams. The information in the ER diagrams has to be stored in a database. Multiple ER diagrams may have to be stored.

- **a.** Give an ER diagram for storing information about (multiple) ER diagrams. You can assume that the stored ER diagrams have only strong entities with only atomic attributes, only binary relationships with no attributes, and the relationships may have only simple cardinality constraints (many-one, one-one, one-many). . . .8
- **b.** Give tables corresponding to the above ER diagram. . . .4

7.15 Draw an ER diagram for recording cricket players statistics. Specifically, we want to track: players, matches, matches that a player played in, his performance in each innings: batting (scores, balls, how out) and bowling (O,M,R,W). . . .7

7.16 You need to help the IITB film society manage its movie voting system.

- **a.** Draw an ER diagram for the movie voting system. The system should track (i) members (id and name) (ii) movies (including title, description, when shown (null if not shown yet)), (iii) reviews for the movie (including author

of review), (iv) movies proposed to be shown next month, and (v) votes cast by members for each proposed movie. . . .5

b. Convert the ER Diagram to SQL table definitions including applicable constraints. . . .4

c. Give constraints in SQL to ensure that (i) each member votes at most once for each movie, and (ii) each member votes for at most five movies. . . .2 + 3

7.17 Illustrate two ways of converting a generalization/specialization hierarchy into a set of tables, and mention (a) under what conditions each can be used, and (b) the advantages of each of the schemes. . . .5

7.18 You are the CEO/CTO/Sole-Programmer-Analyst of ERDiagramsRUS, a leading ER diagramming company. ShadyNDangerous Tobacco Co has asked you to come up with an ER model for their organization, including

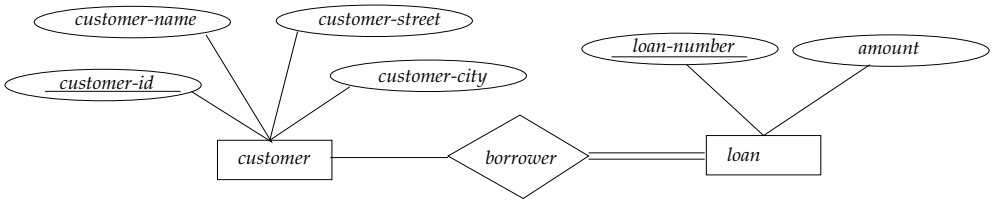
a. ProductType information (ProductTypeID, name, ...)
b. Product information (ProductID, ProductTypeID, weight, ..)
c. Distributor information
d. Who distributes what products
e. Sales for each distributor for each product for each month of each year. (Model year/month as an entity)

Give an ER diagram for the above scenario. Clearly document any assumptions you have made while coming up with the diagram.

Give the tables that would be generated from the above ER diagram (no need for type information etc, and don't write it in SQL).

. . .13

7.19 Translate the following ER diagram into relational tables, by giving appropriate **create table** clauses, along with all appropriate constraints.



Note: a foreign key is specified in SQL as part of **create table**, in the form **foreign key** (A1,A2) **references** R. A **check** clause can have an expression with a subquery, which is useful for representing one of the constraints. . . .6

CHAPTER 8

Relational-Database Design

Exercises

8.1 What is a lossy join decomposition? Give a small example including a relation instance and its decomposition. . . .4

8.2 Give both dependency preserving and non-dependency preserving BCNF decompositions of $R(N, S, G, L, K)$ under the dependencies $S, G \rightarrow L$ and $N \rightarrow S, G$. . . .4

8.3 Given a set of dependencies $A \rightarrow B$, $B \rightarrow C$ and $C \rightarrow D$, and a relation $r(A, B, C, D)$, what is the best order for the BCNF decomposition to consider the dependencies, and why? . . .3

8.4 Give the definition of when a relation is in BCNF. . . .2

8.5 Give a necessary and sufficient condition for a decomposition of R into $R1$ and $R2$ to be lossless join. Very briefly explain why it necessary and sufficient. . . .2

8.6 Given the relation $r(A, B, C)$ and the functional dependencies $A \rightarrow B$ and $B \rightarrow C$, give a lossless join dependency preserving decomposition of r into BCNF. . . .2

8.7 For the same relation as above, give a lossless join but non-dependency preserving decomposition of r into BCNF. . . .2

8.8 What, from a normalization perspective, is wrong with a system of student-id numbers that encodes year, major and degree program into the student-id (e.g. 2000CSEBS034). Give an example of the problems it causes. . . .3

8.9 Given $R = (A, B, C, D, E)$ and $F = \{AB \rightarrow C, D \rightarrow E\}$, give all candidate keys for R. . . .2

8.10 Show that all binary relations are in BCNF. . . .3

8.11 Does $A \twoheadrightarrow BC$ logically imply $A \twoheadrightarrow B$ and $A \twoheadrightarrow C$? If yes prove it, else give a counter example. . . .3

8.12 Consider the following functional dependencies for relation schema $R = (A, B, C, D, E)$: $A \rightarrow BC$, $CD \rightarrow E$, $B \rightarrow D$, $E \rightarrow A$.
Compute A^+. . . .2

8.13 Give an example of a relation, with one or more multivalued dependencies, which is not in 4NF, and show how to bring your example to 4NF. . . .2+ 2

8.14 Dependency theory

a. Give an example of functional dependencies on relation $r(A, B, C)$ such that r does not satisfy BCNF, and give an example relation showing the resultant redundancy. . . .2 + 2

b. Use the definition of functional dependency to argue that the augmentation and transitivity axioms are sound. . . .6

c. Given the following functional dependencies

$A \rightarrow BCD$
$CD \rightarrow E$
$B \rightarrow D$
$E \rightarrow A$
$AD \rightarrow E$

i. Find a canonical cover of the above set of dependencies (you must explain how you arrived at the answer) . . .6

ii. Normalize the relation to 3NF (again, you must explain how you arrived at the answer). . . .4

8.15 Show using the schema $r(A, B, C, D, E)$ with functional dependencies $A \rightarrow B$ and $BC \rightarrow D$, that using the original set of dependencies is not enough to test if a decomposed relation is in BCNF. . . .5

8.16 Dependency theory

a. Prove, from first principles, the transitivity axiom for multivalued dependencies ($\alpha \twoheadrightarrow \beta$ and $\beta \twoheadrightarrow \gamma \Rightarrow \alpha \twoheadrightarrow \gamma - \beta$). . . .4

b. Prove, using Armstrong's axioms, the union rule for functional dependencies ($\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma \Rightarrow \alpha \rightarrow \beta\gamma$). . . .3

c. Does $A \twoheadrightarrow BC$ imply $A \twoheadrightarrow B$ and $A \twoheadrightarrow C$. Prove or give a counter-example. . . .3

d. Given a schema $R(A, B, C, D)$, with an MVD $A \twoheadrightarrow BC$, what MVDs, if any, hold on a decomposition of R into $R1(A, B, D)$ and $R2(A, C, D)$? Prove your answer from first principles. . . .5

8.17 Normalization

a. Give an example of a relation with two columns and two tuples that satisfies all possible functional dependencies. . . .2

b. Consider a relation schema $R = (A, B, C, D, E)$ with the functional dependencies

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

Give a lossless join BCNF decomposition of the above schema. . . .4

c. Using first principles, prove the multivalued augmentation rule: If $\alpha \twoheadrightarrow \beta$ holds, and $\gamma \subseteq R$ and $\delta \subseteq \gamma$, then $\gamma\alpha \twoheadrightarrow \delta\beta$ holds. . . .4

d. Compute the canonical cover F_c of R from part (b). Show the individual steps in computing the canonical cover. . . .7

8.18 List all non-trivial functional dependencies satisfied by the following relation instance. . . .5

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

8.19 Give a canonical cover of the following set of functional dependencies.

$$AB \rightarrow CD, \;\; A \rightarrow B, \;\; B \rightarrow C, \;\; D \rightarrow A$$

You must show each step in arriving at the canonical cover, and outline why the step is correct. . . .6

8.20 **a.** Given a relation R, it is sufficient to use the given set of dependencies F instead of F^+ to test for BCNF violation. Explain why this is so (an informal argument is fine, no need for a formal proof). . . .6

b. Give an example to show that when you have decomposed R it is not sufficient to use F to test a decomposed relation. . . .4

c. Suggest an efficient test which can be used in this case, without computing all of F^+. . . .4

8.21 **a.** Give an example of a relation with no dependency preserving decomposition into BCNF, but which is in 3NF. . . .2

b. What is the redundancy in the above example? . . .2

c. Show that the 3NF decomposition algorithm ensures the lossless join property. . . .4

d. State if each of the following is in P or in NP: (a) 3NF testing and (b) 3NF decomposition . . .2

8.22 Normalization

a. Give the definition of BCNF. . . .2

b. Can a relation that is in BCNF not be in 3NF? Why or why not? . . .2

c. Give the definition of 4NF. . . .2

8.23 Suppose you have a relation $transaction(accountno, seqno, amount)$ where the amount can be positive (deposit) or negative (withdrawal), and the sequence number (seqno) is unique within each account number.

Write a query to output every accountno which has a negative balance (the balance is the cumulative total from the earliest sequence number for the account). . . .8

8.24 Questions on advanced normalization (Appendix C)

a. When inferring MVDs on a decomposition of R into R_1 and R_2, what do we need to do after computing $D+$, the closure of the given set of MVDs? . . .3

b. If a schema is in PJNF, list all the other normal forms it is in. . . .2

c. Consider the relations

`registration(rollno, course)` and `student(rollno, name, year)`,

with the following kind of dependency (constraint): students in year 1 should register for exactly 2 courses, students in year 2 should register for exactly 1 course. (No other year is permissible.)

Suggest how to decompose the two relations in such a way that the union (NOT join) of the decomposed relations gives the original relations back, and such that superkey, foreign key and domain constraints on the decomposed relations (without any other constraints) are sufficient to ensure that the constraint above is satisfied.

Give the decomposition and the super/foreign key declarations in pseudocode (no need for SQL). (NOTE: the solution may not be great if the number of courses is large, that's OK.) . . .7

d. Give an example to show that join dependencies are more general than multivalued dependencies; to do so, give a relation that does not satisfy any non-trivial multivalued dependency, but satisfies a non-trivial join dependency. . . .5

CHAPTER 9

Application Design and Development

Currently we have just the one question below.

Exercises

9.1 Consider the following snippet of a program using JDBC:

```
String q = "select * from user where userid = '" +
                    request.getAttribute("uid") + "'";
PreparedStatement pstmt = conn.prepareStatement(q);
ResultSet rs = pstmt.executeQuery();
```

Is there a security risk with the above snippet? If no, explain why, and if yes, explain why and how to fix it. . . .4

CHAPTER 10

Storage and File Structure

Exercises

10.1 If you have data that should not be lost on disk failure, and the data is write intensive, how would you store the data? . . .3

10.2 If the slotted page architecture is used for storing tuples, can indices store pointers to actual tuples? If not, why not, and what do they store instead? . . .2

10.3 Disk capacities have been growing very fast in the last few years, and cost per byte has been falling equally fast. However, access times (latency/seek) have decreased at a much slower rate. Which of RAID 1 or RAID 5 does this trend favor, and why? . . .2

10.4 Describe one benefit of each of RAID 5 and RAID 1 (mirroring) relative to each other. . . .2

10.5 In disks in the past, the number of sectors per track was the same across all tracks. These days, disks have more sectors per track on outer tracks, and fewer sectors per track on inner tracks (since they are shorter in length). What is the impact of such a change on each of the three main indicators of disk speed? . . .2

10.6 Consider a RAID system with block striping (e.g. RAID 5) and a workload with many small I/Os and one long sequential read running concurrently.

a. How would the transfer rate for the sequential read on a RAID system with 8 disks compare with reading the data from a single dedicated disk, if one block is read at a time? . . .2

b. Suggest how to perform the read to improve the transfer rate from the RAID system. . . .2

10.7 In a slotted page architecture, objects (tuples) are accessed indirectly via a header table at the beginning of the page. That is, a tuple id gives a page-id and an offset in the header table. Each entry in the table consists of a pointer (offset in the page) to an object. Describe the main advantage of this architecture over one where tuple-ids has a page-id and a direct offset to the tuple. . . .4

With the above scheme, the table at the beginning of the page can have "holes" corresponding to deleted tuples. Describe a small modification to the scheme, so that the table does not have to store any holes. . . .6

10.8 (Buffer management)

a. Standard buffer(cache) managers assume each page is of the same size and costs the same to read. Consider a cache manager that, instead of LRU, uses the rate-of-reference to objects, i.e., how often an object has been accessed in the last n minutes. Suppose you wanted to cache objects of varying sizes, and varying read costs (e.g., web pages, whose read cost depends on the site you are fetching from). Suggest how one may decide which page to evict. . . .3

CHAPTER 11

Indexing and Hashing

Exercises

11.1 Why are B-trees, which have a very large fanout (i.e., number of children), preferred over binary trees (having at most two children) in the context of disk-based databases? . . .3

11.2 What is the maximum height of a B^+-tree with N keys and a maximum fanout of k. . . .1

11.3 Suppose you want to build a B^+-tree index on a CD-ROM, where the transfer rates are comparable to disk transfer rates, but seek times are much higher. Should the block size be the same as on disk, higher or lower? Explain intuitively (in one or two sentences) why. . . .3

11.4 Name two factors that affect the decision of the page size of a B^+-tree (e.g., should the page be 1KB or 16 KB). . . .2

11.5 Give three alternative ways of evaluating a selection of the form $\sigma_{A=\$1 \wedge B=\$2}(r)$ if secondary indices are available on A and on B. . . .3

11.6 Suppose I store two dimensional points as the location attribute of a relation, and queries ask for tuples whose location is at exactly a given point. What sort of index is best for this case? . . .2

11.7 Can a secondary (unclustered) index be worse than a file scan for a select? Describe a rough scenario. . . .4

11.8 Suggest an easy way (other than using buckets of key values) to implement B^+-trees on non-unique search keys. Briefly outline how to find all records with a given search key value using such an index. . . .3

11.9 The textbook description of static hashing assumes that a large contiguous stretch of disk blocks can be allocated to a static hash table. Suppose you can only allocate C contiguous blocks. Briefly sketch how to implement the hash table, if it can be much larger than C blocks. Access to a block should still be efficient. . . .3

11.10 What should you do to make sure that the worst case occupancy of a B$^+$-tree is at least 75%? . . .2

11.11 Suppose I have a relation with n_r tuples on which I want to build a secondary B$^+$-tree.

a. Estimate the cost of building the B$^+$-tree index by inserting one record at a time. Assume each page will hold an average of f entries. . . .3

b. Suggest a more efficient way of constructing the index; just outline the key ideas, no need for a detailed description. . . .3

11.12 Suppose you want to efficiently support queries of the following form

```
select sum(A) from r where v1 < r.B and v2 > r.B
```

where `v1` and `v2` are values supplied at runtime; the same query is invoked repeatedly with different values for `v1` and `v2`. Describe how one could store extra information in the internal nodes of a $B+$-tree index on `r.B` to efficiently compute the above sum. . . .8

11.13 In extendible hashing, if the occupancy of a bucket is too low, what bucket(s) are you allowed to merge the underfull bucket with? . . .2

11.14 Suppose we insert the following records into an extendible hash structure in the order shown: (B,1), (D,1), (D,2), (M,1), (P,1), (P,2), (P,3). Assume that B hashes to (binary) 0000, D hashes to 0101, M to 0111 and P to 0111. Show the resultant extendible hash structure, assuming that each bucket can hold at most 2 pointers. Assume that suffixes of the hash value (as shown in binary above) are used to identify buckets. . . .4

11.15 Suppose that we are using extendible hashing on a file containing records with the following search key values: 2,3,5,7,11,17,19,23,29

Show the extendible hash structure for this file if the hash function is h(x) = x mod 8 and buckets can hold three records. Further, assume that we consider the suffix of the bit representation (i.e., the last global depth bits) to index into the directory. Also, assume buckets can hold three records. Show all the intermediate steps while populating the hash index. Assume initially the hash index consists of two entries in the directory each pointing to a distinct empty bucket.

Suppose that the following modifications are applied to the hash index: Delete 11, Delete 31, Insert 1, Insert 15.

The changes are cumulative – each change is applied before applying the next. Show the hash index after each change. . . .10

11.16 **a.** Outline the steps in building a B$^+$-tree bottom-up (that is, directly from a given relation without performing a series of inserts). ...4

b. Compare the cost of bottom up build as above with building a tree by a sequence of inserts, in terms of I/O as well as in terms of space used. ...2

c. Very briefly suggest how an R-tree can be constructed bottom-up. Your answer need not be technically complete, or optimal in any sense, just briefly outline an idea/heuristic. ...3

11.17 **a.** Is it always a good idea to have an index. In other words, what are the pros and cons for creating an index. ...1

b. Given an SQL query of the form `select` L `from` $r_1, \ldots r_n$ `where P` where P represents a predicate, list indices that may potentiay be useful for answering the query. ...3

11.18 Suppose access to some key values in a B$^+$-tree is very common relative to others, that is the access pattern is highly skewed. Suggest a scheme for caching data at internal nodes of the B$^+$-tree to minimize access times. Assume that with each node you can allocate extra space for caching data. ...8

11.19 Describe an efficient algorithm to build a B$^+$-tree on a file which is sorted on the indexed attribute. Also, outline the cost of the algorithm in terms of I/O's. Assume the file occupies N pages and the resulting B$^+$-tree will occupy M pages. Assume there are $h + 2$ buffer pages where h is the final height of the B$^+$-tree. ...12

C H A P T E R 1 2

Query Processing and Query Optimization (Chapters 12 and 13)

This section contains questions related to Chapters 12 and 13.

Exercises

12.1 Consider external sorting using replacement selection. What is the minimum, average, and maximum length of the runs generated? Explain when the minimum and maximum length runs are generated. . . .3

12.2 Suppose you want to compute the join of relations A and B, where A fits in memory with plenty of space to spare, and B is much larger than memory. What join technique would you use? . . .2

12.3 Very briefly outline how to extend merge-join to compute the full outer natural join. . . .3

12.4 Describe how to compute full outer join by using an extension of the hash join algorithm. . . .4

12.5 (Iterator model) Consider an iterator implementation of nested-loops join of relations a and b. Give pseudocode for the getnext() function. . . .4

12.6 Pipelining is used to avoid writing intermediate results to disk. Suppose you are merge joining the results of two sorts. Describe how the output of the sort can be effectively pipelined to the merge join without being written back to disk. . . .8

12.7 Explain how to implement $A \supset\!\bowtie B$ using hashing, where A is used as the build relation. . . .6

12.8 Explain how to implement the division operator, using sort-merge. (Recall that division of $r(A, B)$ by $s(B)$ finds values a for which (a, x) is in r for every $(x) \in s$.) . . .8

12.9 Suppose I want to provide an iterator interface to external sorting. Outline what should be done in the open(), getnext() and close() functions. . . .5

12.10 When using the block nested loops join algorithm to join relations A and B, suppose memory has M blocks. How many blocks should be given to A and how many to B? What is the cost of the algorithm assuming A has b_a blocks and B has b_b blocks. . . .2 + . . .2

12.11 Given a materialized view $A \bowtie B$, how would you update it if some tuples are added to A; let i_A denote the tuples added to A. . . .3

12.12 Suppose the memory size is M 4096 byte blocks. How big does a relation have to be to require recursive partitioning, if the average size of each partition must be $0.8 * M$ blocks. . . .2

12.13 What is the cost of finding the best left-deep join order and the cost of finding the best right-deep join order? . . .1

12.14 Suppose you have a query that only wants the first ten results in sorted order of attribute A, of $r \bowtie_{r.A=s.A} s$. Give conditions under which it suffices to use only the first ten results (in sorted order of A) of relations r and s to compute the result. . . .3

12.15 What is the motivation behind the optimization heuristic "Push selects through joins". . . .3

12.16 Let r and s be relations with the number of blocks in the two being b_r and b_s respectively. Assume that there are no indices on the two relations and they are not sorted. Assuming infinite memory, what is the cheapest way (in terms of I/O operations) to compute $r \bowtie s$ and what is the amount of memory required for this algorithm. . . .4

12.17 For each of the following pairs of queries, indicate under what conditions are they equivalent (i.e. they return exactly the same answers),

a. `select distinct r.A from r` and
`select distinct r.A from r, s where r.B = s.B` . . .2

b. `select distinct r.A from r` and
`select distinct r.A from r, s` . . .3

12.18 Give examples to show non-equivalence of the following pairs of expressions. Assume the relation schemas are $A(X, Y), B(X, Z)$ and $C(X, W)$.

a. $A \rightarrowtriangle (B \bowtie C)$ and $(A \rightarrowtriangle B) \bowtie C$? . . .3

b. $A \rightarrowtriangle (B \bowtie C)$ and $(A \bowtie B) \rightarrowtriangle C$? . . .3

c. $A \rightarrowtriangle (B \rightarrowtriangle C)$ and $(A \rightarrowtriangle B) \rightarrowtriangle C$? (Hint: consider values common to A and C but not B. . . .4

12.19 Why are the following equivalences incorrect; describe the reason either in words or through a small example. Assume that initially relations have no duplicates or nulls. (Some of the expressions below can generate tuples with nulls.)

a. $\Pi_A(R - S) \leftrightarrow \Pi_A(R) - \Pi_A(S)$. ...4
b. $\sigma_{B<4}(\ {}_A\mathcal{G}_{max(B)}(R)) \leftrightarrow\ {}_A\mathcal{G}_{max(B)}(\sigma_{B<4}(R))$. ...5
c. Does the above equivalence hold if max is replaced by min? Explain very briefly. ...4
d. $(R \supset\!\bowtie S) \supset\!\bowtie T \leftrightarrow R \supset\!\bowtie (S \supset\!\bowtie T)$. ($\supset\!\bowtie$ denotes the natural left outer join). ...10
(Hint: assume that the schemas of the three relations are $R(a, b1), S(a, b2)$ and $T(a, b3)$ respectively.)

12.20 Consider

```
select r.A, r.B, r.C from r
where r.B = (select avg(s.B) from s where s.A=r.A)
```

Describe how to evaluate the above query efficiently using

a. Sorting of both r and s. ...4
b. Caching, assuming there is a clustered index on $s.A$ and $V(B, r)$ is small compared to the size of memory. ...3

12.21 Suppose you have to compute ${}_A\mathcal{G}_{sum(C)}(r)$ as well as ${}_{A,B}\mathcal{G}_{sum(C)}(r)$. Describe how to compute these together using a single sorting of r. ...4

12.22 Consider a relation $r(A, B, C)$, with an index on attribute A. Give an example of a query that can answered using the index only, without looking at the tuples in the relation. ...2

12.23 Suppose you want to get answers to $r \bowtie s$ sorted on an attribute of r, and want only the top n answers for some relatively small n. Give a good way of evaluating the query

a. when the join is on a foreign key of r referencing s. ...3
b. when the join is not on a foreign key. ...3

12.24 Given two SQL queries of the form

select $A_1, \ldots, A_n$
from $R_1, \ldots, R_n$
where P_1
groupby G_1

and

select $B_1, \ldots, B_m$
from $S_1, \ldots, S_n$
where P_2
groupby G_2

give conditions under which the result of the first query can be used to compute the second query.

(Note: R_i and S_i is not the actual name of the relations, some R_i's may be the same as some S_j's, and similarly for the A_i's and B_i's; G_1 and G_2 are lists of groupby attributes and P_1 and P_2 are predicates.) ...6

12.25 Suppose you want to compute the results of all the following queries: ${}_{A,B,C}\mathcal{G}_{count(S)}(r)$, ${}_{A,B}\mathcal{G}_{count(S)}(r)$, ${}_{A}\mathcal{G}_{count(S)}(r)$. Outline an efficient sorting-based algorithm for computing them together. . . .5

12.26 Suppose you want to compute a groupby on a relation, where you believe the number of groups is likely to be much smaller than memory, but aren't 100% sure. Give an algorithm for computing the groupby, which works very well if the number of groups is small enough, and degrades in a smooth fashion if not. . . .10

12.27 Suppose you have a huge relation and want to do a groupby and find groups whose count is greater than 1 percent of the size of the relation. The total number of groups may be very large, so you do not have enough memory to maintain a count for every group.

Suggest an efficient algorithm for doing so *without* sorting or partitioning the data – in fact the data may be read but never written. (Hint: Use some form of hashing where multiple groups may have the same hash value, to filter out many irrelevant groups). . . .10

12.28 Consider a block nested loop join, where memory has $M + 1$ pages. Let the relations r and s have b_r and b_s pages. Suppose we divide memory as follows: one page for output, i pages for r and $M - i$ pages for s.

- **a.** What is the cost of block nested loop join of r and s, with r as the outer relation.
- **b.** Based on your cost formula, what value of i gives the lowest cost.
- **c.** If you could choose which of r and s is the outer relation, how would you choose it? . . .8

12.29 Consider a join $r \bowtie s$, where both r and s are pipelined in (that is, they are generated by other operations, and are piped in a tuple at a time). Let the schemas be $r(A, B)$ and $s(A, C)$. Suppose both r and s, with indices on A, fit in memory. Describe an algorithm that can generate tuples of $r \bowtie s$ without waiting for the entire input to be available. Structure your description by first giving the key idea and then more details. . . .10

12.30 Consider the issue of interesting orders in optimization. Given a set of relations S to be joined, and a subset $S1$ of S, what are the interesting orders of $S1$? . . .4

12.31

- **a.** Suppose the number of interesting orders for a query (and each of its intermediate relations) is "d". Suppose also that there is only one join method available. Consider a join of $2n$ relations.
  - i. What is the number of different left-deep join trees. . . .2
  - ii. How many different plans does the System R optimizer actually examine. . . .4
  - iii. What is the maximum number of plans the optimizer has to store at any given point of time during the optimization phase? . . .7
  (Hint: Recall that given a set of size n, the number of subsets of size i is $\binom{n}{i}$.)

b. Consider queries consisting only of joins (that is, no projections or selections). Is it true that the System R algorithm chooses the "optimal" order amongst all join orders? Why? The "optimal" join order is the one with the minimum estimated cost. . . .3

12.32 **a.** Suppose the view $v = r_1 \bowtie r_2$ has been materialized (computed and stored), and now a set of tuples i_1 is inserted into r_1, and i_2 into r_2. Assume you have the sets i_1 and i_2, as well as the contents of relations r_1 and r_2 *after* the insertion. Give an expression to compute the new tuples to be inserted into the view result computed earlier, in order to keep the materialized view up to date. . . .3

b. Suppose the view $v =_A \mathcal{G}_{sum\ B}(r)$ has been materialized and then a single tuple t is added to r. Describe how to keep the view up to date. . . .2

c. Suppose now that the view is $v_1 =_A \mathcal{G}_{avg\ B}(r)$. What extra do you have to store to allow the view result to be updated efficiently? . . .2

12.33 Suppose you want to build a query optimizer can make use of materialized views. Disregard selections and projections, and assume queries as well as materialized views are just natural joins. Describe how to extend the System R style bottom-up join order optimization algorithm to handle materialized views. . . .10

12.34 Suppose you are given a relation *emp(empid, dept, salary)* and wish to maintain a materialized view *deptsalary(dept, totalsalary)* which stores the total salary for all employees of each department. Suppose the system does not support materialized views but supports triggers.

a. Give triggers on insert and delete on *emp* to record changes into *insert-delta* and *delete-delta* relations. Don't worry about updates. . . .6

b. Write a JDBC program to use the tuples in the delta relations to update the total salary. Be sure to handle new departments, and to remove tuples from the delta relation once they have been processed.

NOTE 1: For simplicity, don't bother about departments that become empty (i.e. their last employee is deleted), but outline (in English, not SQL) how to detect and handle this case.

NOTE 2: Don't bother to give code for setting up or deleting connections, statement objects etc. . . .9

CHAPTER 14

Transactions

Exercises

14.1 Show a sequence of updates (withdrawal of money from an account, for example) by two concurrent transactions that results in an incorrect final state. . . .4

14.2 Consider two transactions, one transfering Rs 50 from account A to account B and another transfering Rs 100 from account B to account A.

- **a.** Show a schedule which is not serial, but is conflict serializable. . . .3
- **b.** Show a schedule which is not conflict serializable and results in an inconsistent database state. . . .3

14.3 Give an example of a schedule that is view serializable but not conflict serializable. What is the essential feature of such schedules? . . .2+ 1

14.4 Give the expansion of ACID. . . .1

14.5 Give 2 reasons for allowing concurrent execution of transactions? . . .2

14.6 Give an example of a serializable but not serial schedule (use only reads and writes, don't include other operations). Explain why it is serializable. . . .6

14.7 Give an example of a schedule that is serializable but not recoverable. . . .3

CHAPTER 15

Concurrency Control

Exercises

15.1 Briefly explain what is recoverability and what is cascade freedom. Mention a protocol that ensures both properties, and explain why it ensures both the above properties. . . .5

15.2 If many transactions update private items (eg. individual account balances) and a common item (eg. cash balance at a branch). What can you do to increase concurrency/throughput, without using any fancy locking modes, by ordering of operations in a transaction. . . .5

15.3 Consider the following locking protocol: all items are ordered, and once an item is unlocked, only higher numbered item may be locked. Locks may be released at any time. Only X-locks are used.

Show by an example that this protocol does not guarantee serializability. . . .2

15.4 Mention one locking protocol which ensures that deadlocks never occur, without ever aborting transactions. Give one major drawback of using the protocol. . . .2

15.5 Is every conflict serializable schedule potentially generatable by two-phase locking? If not, give an example. . . .3

15.6 (Locking)

- **a.** What defines the serialization order of transactions if two-phase locking is used? . . .1
- **b.** What are the minimal changes to two-phase locking to guarantee
  - i. recoverability (and explain why it works) . . .2
  - ii. cascade freedom (and explain why it works) . . .2

15.7 (Deadlock prevention)

a. Outline the wound-wait and the wait-die schemes for deadlock prevention. . . .2+ 2

b. The wound-wait and wait-die are not used much in practise. Briefly explain why this may be the case. . . .2

15.8 (TSO Protocol)

a. Outline the TSO protocol for reads . . .2

b. Outline the TSO protocol for writes (including the Thomas write rule) . . .2 + 1

c. Outline how to modify the TSO protocol to guarantee recoverability and cascade freedom. . . .2

15.9 The Oracle database implements a special concurrency control protocol for read-only transactions, whereby a read-only transaction sees a view of the database containing effects of all transactions that committed before it starts. Assume that updates are immediate, and only physical logging is used. Explain how to use undo logs to efficiently provide such a view to read-only transactions. . . .6

15.10 In multigranularity locking, give the compatibility matrix for the different lock modes (the two normal modes plus the three intention lock modes). Also give an explanation of line each for those entries of the matrix that involve two intention lock modes. . . .4

15.11 Show that given any schedule generated by two-phase locking, the transaction dependency graph is acyclic (thereby you would have shown that two-phase locking generates only conflict serializable schedules). . . .6

15.12 Consider the validation based concurrency control protocol you have studied (which is also known as the optimistic concurrency control protocol). Outline the checks that a transaction must perform during validation. . . .6

15.13 Consider multi-version timestamp ordering based concurrency control. Give conditions under which a version will definitely not be used again (and can thus be garbage collected). . . .4

15.14 Recall the tree locking protocol: the first lock can be on any node of the tree, and subsequently nodes can be locked only if the parent is locked. Locking is not required to be two phase, but still guarantees conflict serializability.

a. Consider the following "dumb forest locking protocol". A forest is a collection of trees. The dumb forest locking protocol uses the tree locking protocol independently on each tree. As before locking is not required to be two-phase. Show, using an example that this protocol does not guarantee conflict serializability.

b. Describe a simple extension of the above locking protocols that can be used on a forest. . . .8

15.15 Multi-version 2PL combines features of 2PL and multiversion concurrency control: updaters use locking to see the latest version of data, while read only transactions use old versions of data based on timestamps. Describe the scheme briefly. Make sure you cover the points below, and any others that are important.

a. What are the actions of read-only transactions on begin/read/commit?
b. What are the actions of update transactions on begin/read/write/commit?
. . .10

15.16 Suppose a set of items forms a directed acyclic graph (DAG). Show that the following protocol assures conflict serializability.

a. The first lock can be on any node
b. Subsequently, a node n can be locked only if the transaction holds a lock on at least one predecessor of n, and the transaction has locked each predecessor of n at some time in the past.

. . .10

CHAPTER 16

Recovery

Exercises

16.1 What information do undo log records contain, and what are they used for? . . .3

16.2 (Recovery)

- **a.** Copying the page table in shadow paging can be quite expensive for very large databases. Suggest an implementation of page tables that can reduce the copying required (assuming transactions are small). . . .4
- **b.** Give two reasons (other than the above) why shadow paging is not used in databases today. . . .2
- **c.** In the advanced recovery algorithm, explain why an action is never (successfully) undone more than once, regardless of failures during rollback. . . .3

16.3 (Aries)

- **a.** If at the beginning of the analysis pass, a page is not in the checkpoint dirty page table, will we need to apply any redo records to it? Why? . . .2
- **b.** What is RecLSN, and how is it used to minimize unnecessary redos? . . .2
- **c.** What problem could arise if there is insufficient log space during recovery? How does Aries prevent this from happening? . . .1 + 1
- **d.** How does Aries handle actions that cannot be undone, such as deleting an operating system file? . . .2
- **e.** Suppose you want a transaction consistent archival dump of your database. Suggest a way to do this with minimal interruption of the system. You may create a point in time where no transactions are active, by aborting all active transactions, but must be able to restart transactions immediately after all active transactions have been aborted. . . .4

16.4 The Oracle database system uses undo log records to provide a snapshot view of the database to read-only transactions. The snapshot view reflects updates of all transactions that had committed when the read-only transaction started; updates of all other transactions are not visible to the read-only transactions.

Give a scheme for buffer handling whereby read-only transactions are given a snapshot view of pages in the buffer. Include details of how to use the log to generate the snapshot view, assuming that the advanced recovery algorithm is used. Assume for simplicity that a logical operation and its undo affect only a single page. . . .7

16.5 Consider "dirty" pages in the buffer (i.e., each page that has some updates that are not yet reflected on disk) Suppose you maintain a table that gives the log sequence number of every dirty page in the buffer and periodically write this table out to the log.

a. Describe how to use this table to minimize work during the redo recovery phase of the advanced recovery scheme. . . .4

b. Convert the above scheme to also write out the list of active transactions with the table. Describe recovery with this addition. . . .3

16.6 **a.** Normally an action such as an insertion/deletion/update of a relation also updates all indices on the relation. Suppose, index maintenance is deferred, that is, it is performed only at the end of the transaction. What problems can this cause for a transaction that involves multiple steps.

b. Consider an update statement that scans a relation and deletes tuples with a salary field of less than 50, and reinserts the tuple with the salary value increased by 10. What problems could arise if the update to the relation is done while the scan is in progress. Describe a technique to avoid the problem. . . .8

16.7 If you wanted high availability (i.e., the system should be up for as much time as possible), would you use 2-very safe or 2-safe replication? Why? . . .2

16.8 Consider a hot-spare system, where there is a primary computer and a backup computer. Log records generated at the primary are sent over the network to the backup computer, where they can be used to keep a copy of the database in sync with the primary. If the primary computer fails, the backup starts processing transactions. There are three logging techniques in such an environment: *one safe* where a transaction commits as soon as the commit record hits the disk at the primary, *two-very-safe* where the transaction commits only if the commit log record hits the disk at the primary and the backup, and *two-safe* which is the same as two-very-safe if both sites are up, but if the backup is down a transaction commits as soon as the commit record hits the disk at the primary.

Describe some benefits and drawbacks of each of the three schemes. . . .10

16.9 Suppose you wanted to maintain a remote backup copy of a database.

a. Briefly outline how you could use the system log generated by the advanced recovery algorithm to keep it synchronized with the primary copy. . . .2

b. Describe what happens on transaction abort. . . .1

c. Describe what should be done when the remote backup fails, and when it recovers. . . .2

d. Describe how the remote backup can take over processing if the primary site fails (assume that all the log records generated at the primary have reached the remote backup). . . .3

CHAPTER 17

Database System Architectures, Parallel and Distributed Databases (Chapters 17, 18 and 19)

This section contains questions for Chapters 17, 18 and 19 from the book.

Exercises

17.1 What are transaction scale up and batch scale up? For each, give an example of a database application class where it is a useful performance metric. . . .3

17.2 If you were building a highly parallel database system, which of the following architectures would you use, and why: shared nothing, shared disk or shared memory. . . .2

17.3 Two-phase commit

- **a.** What is the motivation for two-phase commit and what does it ensure? . . .2
- **b.** Outline the main phases of two-phase commit during normal operation. . . .2
- **c.** If a coordinator fails after a site has voted to commit the transaction, what can the site do? . . .2
- **d.** If a site fails before informing the coordinator that it is ready, what should the coordinator do? . . .1
- **e.** When a failed site recovers, what should it do? . . .2

17.4 In 2PC, suppose the coordinator fails after sending a <prepare T > message. Describe the protocol used by participating sites (before the coordinator recovers) to handle the transaction. . . .4

17.5 Suppose all data is stored partitioned in round robin fashion. The partition operator can be specified in the form $part[pa](r)$ where r is a relation and pa is a list of the partitioning attributes; don't worry about the exact partitioning vector. Give a good parallel plan using the partition operator and relational

algebra operators (operating on local data) for the query `SELECT r.C, count(*) FROM r, s WHERE r.A = S.B AND r.C = s.D GROUP BY r.C` . . .5

17.6 2PC

a. In two-phase commit, outline what steps a site has to take when it recovers and finds a <**ready** T, L> record in its log; assume that the coordinator of T is known, and can be contacted, but the status of the transaction has not been finalized yet. . . .3

b. Now, suppose that the coordinator cannot be contacted. What can a site do to find the status of a transaction T, assuming it knows the set of sites participating in transaction T. . . .3

17.7 Majority protocol

a. Explain how reads and writes are executed with replicas handled using the majority protocol. . . .6

b. The majority protocol requires that multiple replicas be updated atomically. How can this be done? . . .2

c. In the event of a network partition, does the majority protocol ensure consistency or availability? Briefly explain your answer. . . .2

17.8 Consider the PNUTS system.

a. If one of the tablets has too much data, what should be done? . . .2

b. If one of the tablet servers has too much load, what should be done? . . .2

C H A P T E R 2 0

Data Analysis and Mining

Currently we have just the one question below.

Exercises

20.1 Consider the schema $r(A, B, C)$, where C is the class to which the tuple belongs, and A and B are attributes that can be used for classification. Assume all attribute values are single letters, and there are only 2 classes T and F.

Give an example with 4 "training instances" (tuples) and a partitioning condition that improves the "purity" (the exact definition of purity doesn't matter). . . .4

C H A P T E R 2 1

Information Retrieval

Exercises

21.1 Web keyword search systems depend critically on relative ranking of pages. Earlier IR approaches were based on number of occurrences of each query term in a particular document (term frequency), with query words weighted by their rarity (inverse document frequency). Explain why these earlier approaches were susceptible to spamming, that is, sites crafting pages to get an artificially high rank. . . .3

21.2 Information Retrieval.

a. Outline how documents are ranked using Term Frequency (TF) and Inverse Document Frequency (IDF). Use the simplest versions of TF and IDF to show how they are used. . . .3

b. Explain why the simplest definition of TF is inadequate, and list at least one better definition of TF. . . .3

21.3 Suppose you are given an *entropy* function that gives a measure of how disordered the elements of a set are: if all the elements belong to the same class the entropy is 0, and if they are equally distributed amongst all classes, the entropy is 1.

Outline (in english) how a decision tree is constructed recursively, making use of the entropy function. . . .5

21.4 Suppose use a B^+-tree to implement a text-indexing system. For each word, a list of all documents containing the word is to be stored. Suppose that each document is given a number (starting from 1) and a list of documents is represented as an array of bits, 1 indicating the document is in the list, and 0 indicating the document is not in the list.

CHAPTER 22

Object-Based Databases

Exercises

22.1 OODB

a. What is persistence by reachability (from persistent roots), and why is it appropriate for Java? . . .3

b. What is the purpose of the template classes `d_Rel_Ref<class, field>` and `d_Rel_Set<class, field>`, and how do their implementations ensure the goal? . . .4

22.2 Why is persistence by reachability appropriate for persistent Java? . . .3

22.3 What is the template class *D_Rel_Ref* used for in ODMG C++? In other words, what sort of consistency does it enforce? . . .3

22.4 Suggest how to use extensions to aggregation to create nested sets in SQL. Specifically, write an extended SQL query on relation r, to nest all B values corresponding to each A value into a set. . . .4

22.5 What is the common motivation for the template types d_Rel_Ref and d_Rel_Set in ODMG and what is the difference between the two. . . .4

22.6 Give an SQL query with nested subqueries in the **select** clause to do the following. Given a relation account(branchname, number, balance) create a relation nestedaccount(branchname, accounts) where accounts is a nested relation with attributes number and balance.

Make sure each branchname occurs only once in the result. . . .6

22.7 What is the drawback in requiring that each object must have a most-specific-type, which must be a type declared by the user? Illustrate using an example. . . .3

22.8 Given a collection object a of type *d_set<d_ref<customer>>*, write code to iterate over all elements of a. The loop need not actually do anything, just iterate over all objects. . . .4

22.9 SQL:1999 permits inheritance of tables, but insists a tuple must belong to a most specific table.

a. Give an example, using table r with subtables s and t, to illustrate what this means. . . .2

b. Give a real life example to illustrate the shortcomings of this requirement. . . .2

c. Given r is a subtable of s, and A is a primary key for both tables, write a nested query to list all tuples in s that do not correspond to tuples in r. (The "select from ONLY s" syntax can be used to do this, but you should NOT use this syntax). . . .3

22.10 Given the nested relation *r(title varchar(20), authors setof(varchar(20)))*, give an extended SQL query to output a flat table, where each (book,author) pair is separately listed, for example converting (DBConcepts, {Avi, Hank, Sudarshan}) to (DBConcepts, Avi), (DBConcepts, Hank), (DBConcepts, Sudarshan). . . .4

22.11 Hardware swizzling is generally associated with page level locking. Suppose you want to implement fine granularity locking in spite of having hardware swizzling (so as to allow programs to directly access objects without type conversion). Suggest a way of doing this, assuming the following: on dereferencing an invalid pointer, a function is called which returns the correct pointer. . . .10

CHAPTER 23

XML

Exercises

23.1 What is meant by recursion in a DTD. Explain with an example. . . .3

23.2 Given an XML dataset with DTD as follows

```
<!DOCTYPE marks [
  <!ELEMENT coursemarks(coursename, studentmark*) >
  <!ELEMENT studentmark(name, mark)>
  ...
]>
```

with elements coursename, name and mark all being of type PCDATA.
Write queries in XQuery/XPath to do the following

a. Find names and marks of all students in the DBIS course. . . .3

b. Generate the "dense rank" of each student of DBIS. Dense rank is defined as follows: all students with the top mark get a rank of 1, all students with the next highest mark get a rank of 2, and so on. . . .7

23.3 XML

a. Give an XML DTD to represent academic information including

i. Student name and information on courses taken by the student (including course number, year, semester, and grade). Nest course taken details inside student elements.

ii. Course information (number/name/syllabus)

iii. Who taught which course in which year/semester (faculty name, course number and year/semester)

Avoid repetition of course information. . . .7

b. Give an XQuery query to find all students who have taken two courses with the same instructor. Output student name, course number, year, semester and faculty name. Hint: XQuery uses the syntax
for $x **in** pathexpression, ...
let..where..result.. . . .6

23.4 Given a flat XML schema corresponding to the bank application with the elements:

bank((account—depositor))*
account(account-no, branchname, balance) and
depositor(name, account-no).

write an XQuery query to create a nested XML structure where each depositor element has all its corresponding account elements nested inside. . . .8

23.5 Give two XML DTDs corresponding to the relational schemas account(number, branch, balance), depositor(number, name), customer(name, address). The first schema should map relational attributes to XML attributes, and the second one maps them to XML elements.

(Hint: DTD syntax: <!DOCTYPE element [<!ELEMENT name(subelements)> ... <!ATTLIST element attribute type value-spec ... > ...]> where value-spec can be default value, #REQUIRED or #IMPLIED. Multiple attribute-type-value-spec triples can come in the same ATTLIST) . . .3 + 3

23.6 Give XPath queries on each of the above DTDs to print out all accounts in the Powai branch. (Hint: XPath queries are like file paths, with selections within [ ]) . . .2 + 2

CHAPTER 24

Advanced Application Development

Exercises

24.1 Benchmarking: Suppose a system runs transaction type A at 50 transactions per second (tps), and type B at 10 tps.

a. Given an equal mix of transactions of each type, what is the effective throughput of the system (in tps), assuming a single CPU, all data is memory resident, and there is no lock contention? . . .2

b. Now if there are 2 transactions of type A for each transaction of type B, what is the effective throughput of the system? . . .2

c. What will be the effective throughput in case (a) above, if both types of transactions get an exclusive lock on a particular tuple T1 at the beginning of the transaction, and release it at the very end? . . .2

24.2 Show pictorially how a database can be modeled as a queueing system and explain under what circumstances it is appropriate to use the queueing model. . . .6