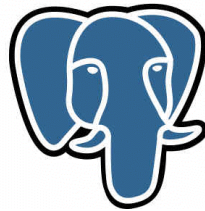


Special Edition for CSEDU Students

TOUCH-N-PASS EXAM CRAM GUIDE SERIES

DBMS – I

PostgreSQL



SYBASE



Prepared By

Sharafat Ibn Mollah Mosharraf

CSE, DU

12th Batch (2005-2006)

Includes Solutions to
DU DBMS – I Final
Exam Questions of
6 Years (2002-2007)

Table of Contents

CHAPTER 1: INTRODUCTION	1
THEORIES.....	1
CHAPTER 2: ENTITY-RELATIONSHIP MODEL	4
QUESTIONS AND ANSWERS	4
CHAPTER 3, 4: RELATIONAL MODEL & SQL	9
POINTS TO BE REMEMBERED.....	9
THE TRICK OF WRITING RA EXPRESSIONS FOR COMPLEX QUERIES	9
COMPLETE CONCEPTS PROBLEM	10
GENERAL STRUCTURE OF QUERY STATEMENTS	16
THEORIES.....	18
CHAPTER 6: INTEGRITY & SECURITY	29
QUESTIONS AND ANSWERS	29
CHAPTER 7: RELATIONAL DATABASE DESIGN	33
CONCEPTS.....	33
QUESTIONS AND ANSWERS	33
CHAPTER 11: STORAGE & FILE STRUCTURE	40
THEORIES.....	40
CHAPTER 12: INDEXING AND HASHING	43
CONCEPTS.....	43
QUESTIONS AND ANSWERS	45

CHAPTER 1

INTRODUCTION

Theories

1.1	<p>What is DBMS? [In-course 2007; 2007. Marks: 1]</p> <p>A database management system (DBMS) is a collection of interrelated data and a set of programs to access those data. The collection of data, usually referred to a database, contains information relevant to an enterprise.</p>												
1.2	<p>Mention some of the areas for database applications. [In-course 1, 2005. Marks: 2]</p> <table style="width: 100%; border: none;"> <tr> <td style="width: 33%;">1. Banking</td> <td style="width: 33%;">2. Airlines /Railways/Road Transport</td> <td style="width: 33%;">3. Universities</td> </tr> <tr> <td>4. Credit Card Transaction</td> <td>5. Telecommunication</td> <td>6. Finance</td> </tr> <tr> <td>7. Sales</td> <td>8. On-line Retailers</td> <td>9. Manufacturing</td> </tr> <tr> <td>10. Human Resources</td> <td>11. Internet</td> <td></td> </tr> </table>	1. Banking	2. Airlines /Railways/Road Transport	3. Universities	4. Credit Card Transaction	5. Telecommunication	6. Finance	7. Sales	8. On-line Retailers	9. Manufacturing	10. Human Resources	11. Internet	
1. Banking	2. Airlines /Railways/Road Transport	3. Universities											
4. Credit Card Transaction	5. Telecommunication	6. Finance											
7. Sales	8. On-line Retailers	9. Manufacturing											
10. Human Resources	11. Internet												
1.3	<p>List four significant differences between a file-processing system and a DBMS.</p> <p>Some main differences between a database management system and a file-processing system are:</p> <ul style="list-style-type: none"> • Both systems contain a collection of data and a set of programs which access that data. A database management system coordinates both the physical and the logical access to the data, whereas a file-processing system coordinates only the physical access. • A database management system reduces the amount of data duplication by ensuring that a physical piece of data is available to all programs authorized to have access to it, whereas data written by one program in a file-processing system may not be readable by another program. • A database management system is designed to allow flexible access to data (i.e., queries), whereas a file-processing system is designed to allow predetermined access to data (i.e., compiled programs). • A database management system is designed to coordinate multiple users accessing the same data at the same time. A file-processing system is usually designed to allow one or more programs to access different data files at the same time. In a file-processing system, a file can be accessed by two programs concurrently only if both programs have read-only access to the file. 												
1.4	<p>What are the disadvantages of DBMS?</p> <p>Two disadvantages associated with database systems are listed below.</p> <ol style="list-style-type: none"> a. Setup of the database system requires more knowledge, money, skills, and time. b. The complexity of the database may result in poor performance. 												
1.5	<p>Explain the difference between physical and logical data independence.</p> <p>Physical data independence is the ability to modify the physical scheme without making it necessary to rewrite application programs. Such modifications include changing from unblocked to blocked record storage, or from sequential to random access files.</p> <p>Logical data independence is the ability to modify the conceptual scheme without making it necessary to rewrite application programs. Such a modification might be adding a field to a record; an application program's view hides this change from the program.</p>												
1.6	<p>What are five main functions of a database administrator?</p> <p>Five main functions of a database administrator are:</p> <ol style="list-style-type: none"> 1. To create the scheme definition 2. To define the storage structure and access methods 3. To modify the scheme and/or physical organization when necessary 4. To grant authorization for data access 5. To specify integrity constraints 												

<p>1.7</p>	<p>Classify database users. [2004. Marks: 2]</p> <p>Database users are differentiated by the way they expect to interact with the system. There are four types of database users:</p> <ol style="list-style-type: none"> 1. Naive users – are unsophisticated users who interact with the system by invoking one of the permanent application programs that have been written previously. 2. Application programmers – are computer professionals who write application programs. 3. Sophisticated users – interact with the system without writing programs. Instead, they form their requests in a database query language. 4. Specialized users – are sophisticated users who write specialized database applications that do not fit into the traditional data processing framework.
<p>1.8</p>	<p>What are the jobs of a DBA? [In-course 2007; 2007; 2004. Marks: 3]</p> <p>The functions of a database administrator (DBA) include:</p> <ol style="list-style-type: none"> 1. Schema definition: The DBA creates the original database schema by executing a set of data definition statements in the DDL. 2. Storage structure and access method definition: File organization (sequential, heap, hash, B+ tree), organization of records in a file (fixed length or variable length), index definition (ordered index, hash index). 3. Schema and physical-organization modification: The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve the performance. 4. Granting of authorization for data access: By granting different types of authorization, the DBA can regulate which parts of the database various users can access. 5. Specifying integrity constraints: The DBA implements key declaration (primary key, foreign key), trigger, assertion, business rules of the organization. 6. Acting as liaison with users. 7. Routine maintenance: <ol style="list-style-type: none"> i. Periodically backing up the database, either onto tapes or remote servers, to prevent loss of data in case of disasters. ii. Ensuring that enough disk space is available for normal operations and upgrading disk space as required. iii. Monitoring jobs running on the database and ensuring better performance.
<p>1.9</p>	<p>What can be done using DML? What are the classes of DML? [In-course 1, 2005. Marks: 4]</p> <p>DML is a language that enables users to access or manipulate data as organized by appropriate data model. The types of accesses are:</p> <ol style="list-style-type: none"> 1. The retrieval of information stored in the database – Query 2. The insertion of new information into the database – insert 3. The deletion of information from the database – delete 4. The modification of information stored in the database – update <p>Classes of DML:</p> <p>There are basically two types:</p> <ol style="list-style-type: none"> 1. Procedural DMLs – user specifies what data are required and how to get or compute the data. E.g. Relational Algebra. 2. Nonprocedural / Declarative DMLs – user specifies what data are required without specifying how to get or compute the data. E.g. SQL.

1.10	<p>How do you classify query languages? Give examples of each type. [In-course 1, 2008; 2006, Marks: 2]</p> <p>Query languages can be classified into two categories:</p> <ol style="list-style-type: none"> 1. Procedural: Relational Algebra 2. Non-procedural: Tuple Relational Calculus, Domain Relational Calculus 										
1.11	<p>What are the differences between schema and instance? [In-course 2007, Marks: 2]</p> <table border="1" data-bbox="181 383 1501 775"> <thead> <tr> <th data-bbox="181 383 842 434">Schema</th> <th data-bbox="842 383 1501 434">Instance</th> </tr> </thead> <tbody> <tr> <td data-bbox="181 434 842 555">1. The overall design of a database is called the database schema.</td> <td data-bbox="842 434 1501 555">1. The collection of information stored in a database at a particular moment is called an instance of the database.</td> </tr> <tr> <td data-bbox="181 555 842 602">2. A relation schema is a type definition.</td> <td data-bbox="842 555 1501 602">2. A relation is an instance of a schema.</td> </tr> <tr> <td data-bbox="181 602 842 649">3. Schemas are changed infrequently, if at all.</td> <td data-bbox="842 602 1501 649">3. Instances are changed frequently.</td> </tr> <tr> <td data-bbox="181 649 842 775">4. This corresponds to the variable declaration (with type definition) of a programming language.</td> <td data-bbox="842 649 1501 775">4. The values of the variable in a program at a point in time correspond to an instance of the database schema.</td> </tr> </tbody> </table>	Schema	Instance	1. The overall design of a database is called the database schema.	1. The collection of information stored in a database at a particular moment is called an instance of the database.	2. A relation schema is a type definition.	2. A relation is an instance of a schema.	3. Schemas are changed infrequently, if at all.	3. Instances are changed frequently.	4. This corresponds to the variable declaration (with type definition) of a programming language.	4. The values of the variable in a program at a point in time correspond to an instance of the database schema.
Schema	Instance										
1. The overall design of a database is called the database schema.	1. The collection of information stored in a database at a particular moment is called an instance of the database.										
2. A relation schema is a type definition.	2. A relation is an instance of a schema.										
3. Schemas are changed infrequently, if at all.	3. Instances are changed frequently.										
4. This corresponds to the variable declaration (with type definition) of a programming language.	4. The values of the variable in a program at a point in time correspond to an instance of the database schema.										
1.12	<p>What is data dictionary? [2004. Marks: 1]</p> <p>A data dictionary contains metadata (data about data). The data dictionary is considered to be a special type of table, which can only be accessed and updated by the database system itself (not a regular user). A database system consults the data dictionary before reading or modifying actual data.</p>										
1.13	<p>What are the components of query processor? [In-course 1, 2005. Marks: 3]</p> <p>The components of query processor are:</p> <ol style="list-style-type: none"> 1. DDL interpreter: Interprets DDL statements and records the definitions in the data dictionary. 2. DML compiler: Translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands. 3. Query evaluation engine: Executes low-level instructions generated by the DML compiler. 										

CHAPTER 2

ENTITY-RELATIONSHIP MODEL

Questions and Answers

2.1	<p>Why E-R model is used for data manipulation? [2002. Marks: 2]</p> <p>E-R model is used for data manipulation because:</p> <ol style="list-style-type: none"> 1. It can express the overall logical structure of a database graphically. 2. E-R diagrams are simple and clear.
2.2	<p>What is the basic difference between E-R diagram and Schema diagram? [In-course 1, 2005. Marks: 1]</p> <p>The basic difference between E-R diagram and schema diagram is that E-R diagrams do not show foreign key attributes explicitly, whereas schema diagrams show them explicitly.</p>
2.3	<p>Define the following:</p> <ol style="list-style-type: none"> 1. Composite attribute [2007, 2003. Marks: 1] 2. Multivalued attribute [2007, 2003. Marks: 1] 3. Derived attribute [2007, 2003; In-course 1, 2005. Marks: 1] <p>Composite attributes: Attributes that can be divided into subparts are called composite attributes. For example, the composite attribute <i>address</i> can be divided into attributes <i>street-number</i>, <i>street-name</i> and <i>apartment-number</i>.</p> <p>Multivalued attributes: Attributes that have multiple values for a particular entity are called multivalued attributes. For example, an <i>employee</i> may have multiple telephone numbers. So, the attribute <i>telephone-no</i> is a multivalued attribute.</p> <p>Derived attribute: If the value of an attribute can be derived from the values of other related attributes or entities, then that attribute is called a derived attribute. For example, if an entity set <i>employee</i> has two attributes <i>date-of-birth</i> and <i>age</i>, then the attribute <i>age</i> is a derived attribute as it can be derived from the attribute <i>date-of-birth</i>.</p>
2.4	<p>Explain the difference between a weak entity set and a strong entity set. [In-course 2, 2007; 2005; 2003. Marks: 2]</p> <p>A strong entity set has a primary key. All tuples in the set are distinguishable by that key. A weak entity set has no primary key unless attributes of the strong entity set on which it depends are included. Tuples in a weak entity set are partitioned according to their relationship with tuples in a strong entity set. Tuples within each partition are distinguishable by a discriminator, which is a set of attributes.</p>
2.5	<p>Show with an example the association between a weak entity set and a strong entity set using E-R diagram. [In-course 2, 2007; 2003. Marks: 1]</p> <p>The diagram illustrates a weak entity 'loan-payment' (represented by a double-bordered diamond) connected to two strong entities: 'loan' (rectangle) and 'payment' (rectangle). The 'loan' entity has two attributes: 'loan-number' and 'amount'. The 'payment' entity has three attributes: 'payment-date', 'payment-number', and 'payment-amount'. The 'loan-payment' entity is connected to both 'loan' and 'payment' entities, indicating a relationship between them.</p>
2.6	<p>We can convert any weak entity set to a strong entity set by simply adding appropriate</p>

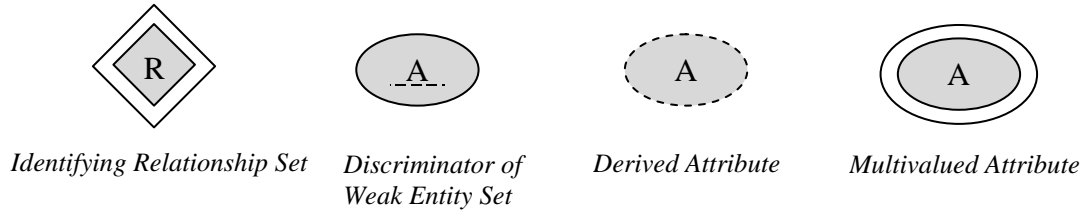
	<p>attributes. Why, then, do we have weak entity sets?</p> <p>We have weak entities for several reasons:</p> <ul style="list-style-type: none"> • We want to avoid the data duplication and consequent possible inconsistencies caused by duplicating the key of the strong entity. • Weak entities reflect the logical structure of an entity being dependent on another entity. • Weak entities can be deleted automatically when their strong entity is deleted. • Weak entities can be stored physically with their strong entities.
2.7	<p>What is the purpose of constraints in database? [2002. Marks: 2]</p> <p>The purposes of constraints in database are:</p> <ol style="list-style-type: none"> 1. To implement data check. 2. To centralize and simplify the database, so to make the development of database applications easier and more reliable.
2.8	<p>What are the constraints used in E-R model? [In-course 2, 2007. Marks: 1]</p> <p>Constraints used in E-R model:</p> <ol style="list-style-type: none"> 1. Cardinality Constraints 2. Participation Constraints 3. Key Constraints
2.9	<p>What participation constraints are used in E-R model? [2006. Marks: 1] OR, Explain the participation constraints in E-R model. [In-course 2, 2007. Marks: 1] OR, Explain with example the participation constraints in E-R model. [2003. Marks: 3]</p> <p>The participation constraints used in E-R model are:</p> <ol style="list-style-type: none"> 1. Total 2. Partial <p>The participation of an entity set E in a relationship set R is said to be <i>total</i> if every entity in E participates in at least one relationship in R. If only some entities in E participate in relationships in R, the participation of entity set E in relationship R is said to be <i>partial</i>.</p> <p>For example, we expect every <i>loan</i> entity to be related to at least one <i>customer</i> through the <i>borrower</i> relationship. Therefore the participation of <i>loan</i> in the relationship set <i>borrower</i> is total.</p> <p>In contrast, an individual can be a bank customer whether or not she has a loan with the bank. Hence, it is possible that only some of the <i>customer</i> entities are related to the <i>loan</i> entity set through the <i>borrower</i> relationship, and the participation of <i>customer</i> in the <i>borrower</i> relationship set is therefore partial.</p>
2.10	<p>Explain the distinction between total and partial constraints.</p> <p>In a total design constraint, each higher-level entity must belong to a lower-level entity set. The same need not be true in a partial design constraint. For instance, some employees may belong to no work-team.</p>
2.11	<p>Let R be binary relationship between A and B entity sets.</p> <ol style="list-style-type: none"> 1. Show the mapping cardinalities using E-R diagrams. [In-course 1, 2005. Marks: 2] 2. How primary keys can be defined for the relationship set R for different mapping cardinalities? [2006. Marks: 2] 3. How can you combine the tables (if possible) for different mapping cardinalities? [2004. Marks: 3] <p>1. Mapping Cardinalities:</p> <p style="text-align: center;"> <i>One-to-One</i> <i>One-to-Many</i> <i>Many-to-One</i> <i>Many-to-Many</i> </p> <p>2. Primary keys for R:</p>

1. One-to-One: PK_A or PK_B [PK_A means *Primary Key* of the entity set A]
2. One-to-Many: PK_B
3. Many-to-One: PK_A
4. Many-to-Many: PK_A and PK_B

3. Combination of tables for different mapping cardinalities:

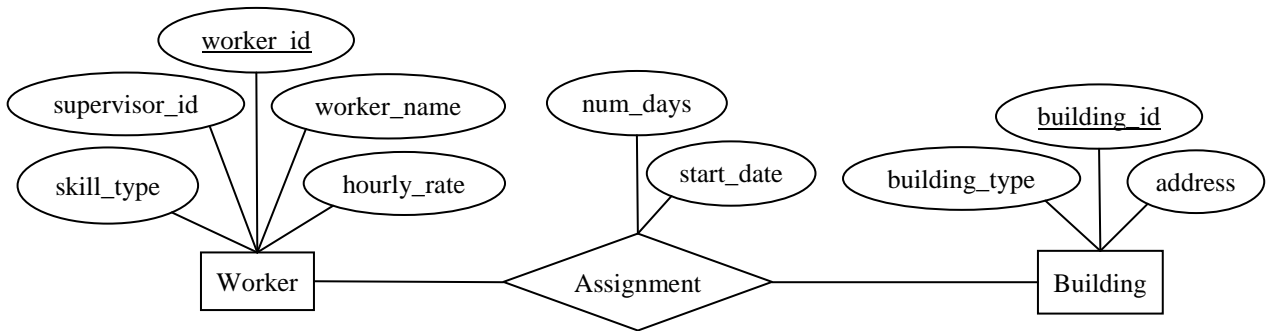
1. One-to-One: –
2. One-to-Many: B and AB
3. Many-to-One: A and AB
4. Many-to-Many: –

2.12 Draw the symbols of *identifying relationship set*, *discriminator of weak entity set*, *derived attribute* and *multivalued attribute* used in E-R model. [2004. Marks: 2]



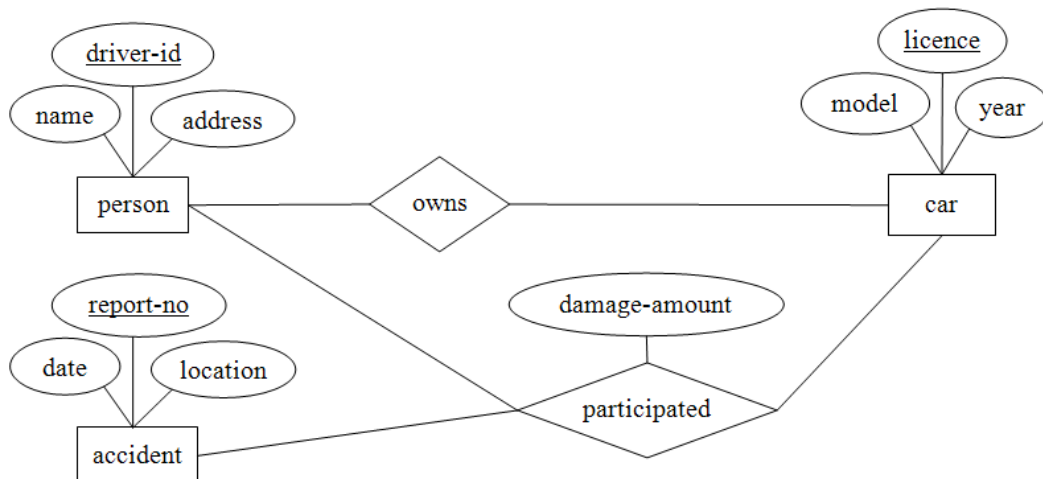
2.13 Draw the E-R diagram for the following relation schemas: [In-course 2, 2007. Marks: 1.5]

Worker (worker_id, worker_name, hourly_rate, skill_type, supervisor_id)
Assignment (worker_id, building_id, start_date, num_days)
Building (building_id, address, building_type)

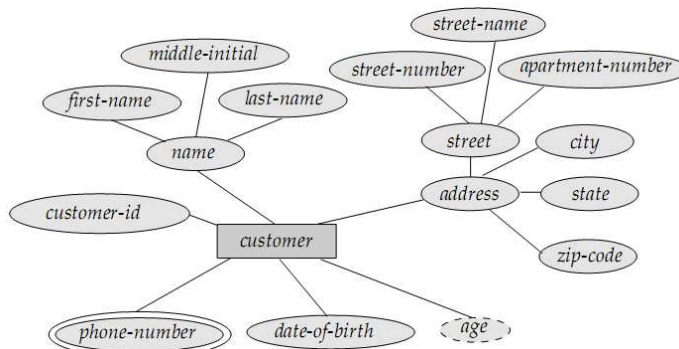


2.14 Give the E-R diagram for the following database: [In-course 1, 2005; 2003. Marks: 2]

person (driver-id, name, address)
car (licence, model, year)
accident (report-no, date, location)
owns (driver-id, licence)
participated (driver-id, licence, report-no, damage-amount)

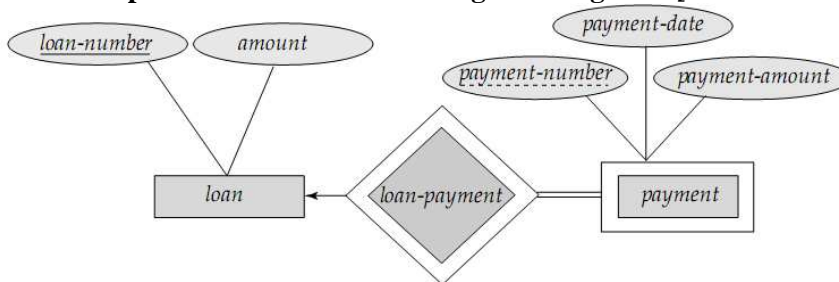


2.15 What will be the tabular representation of the following E-R diagram? [In-course 2, 2007; In-course 1, 2005. Marks: 2]



customer (customer-id, first-name, middle-initial, last-name, date-of-birth, street-number, street-name, apartment-number, city, state, zip-code)
 customer-phone (phone-number, customer-id)

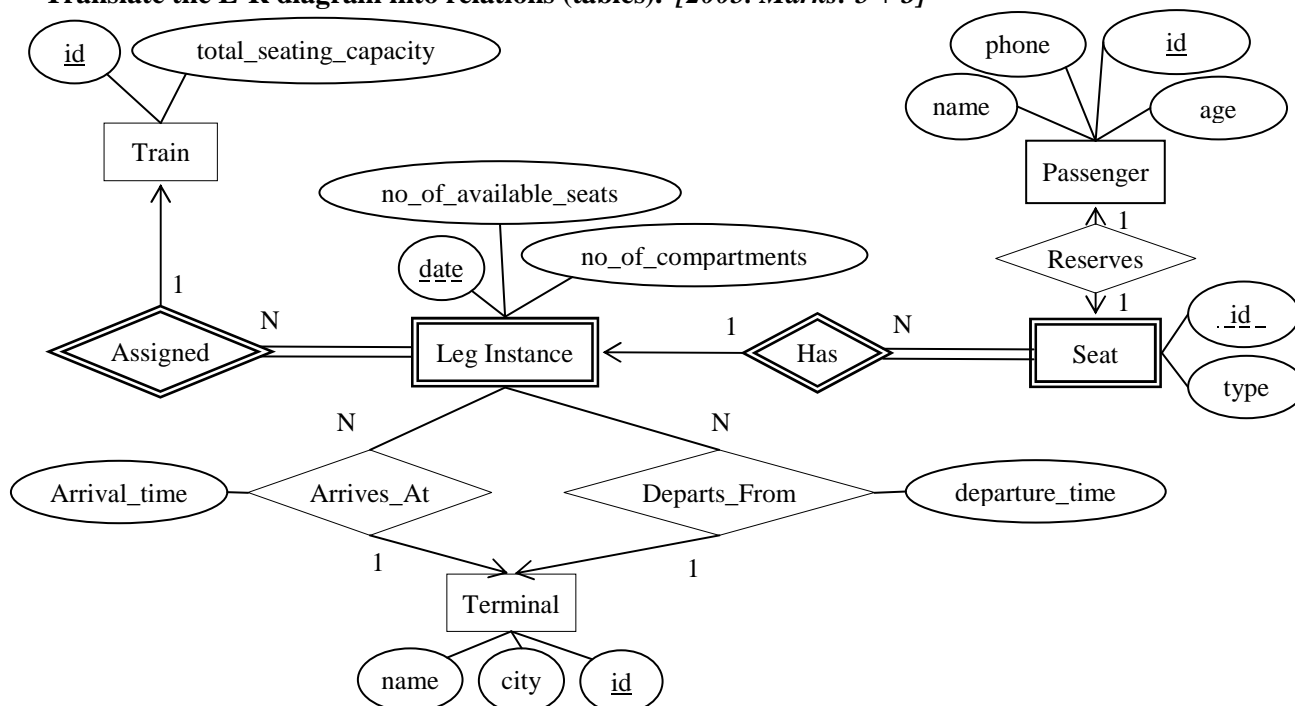
2.16 What will be the schema representation of the following E-R diagram? [2007. Marks: 2]



loan (loan-number, amount)
 loan-payment (payment-number, loan-number, payment-date, payment-amount)

2.17 We are interested to make a database for Railway Reservation System. (We will limit only for inter-city train between Dhaka and Sylhet.) Generally, a passenger takes flight of inter-city train that operates between Dhaka-Chittagong-Dhaka and Dhaka-Sylhet-Dhaka. Each train is identified by an ID and total seating capacity. Each train is assigned a leg instance (an instance of a flight on a specific date) for which we will keep number of compartments, number of available seats and date. Passenger reserves seat of each leg instance. For seat, we will keep seat ID and type. Each leg instance departs from a terminal and arrives to a terminal. We will keep departure time and arrival time; and for terminal, we will store its ID, name and city. For each passenger, we will store name, phone and age.

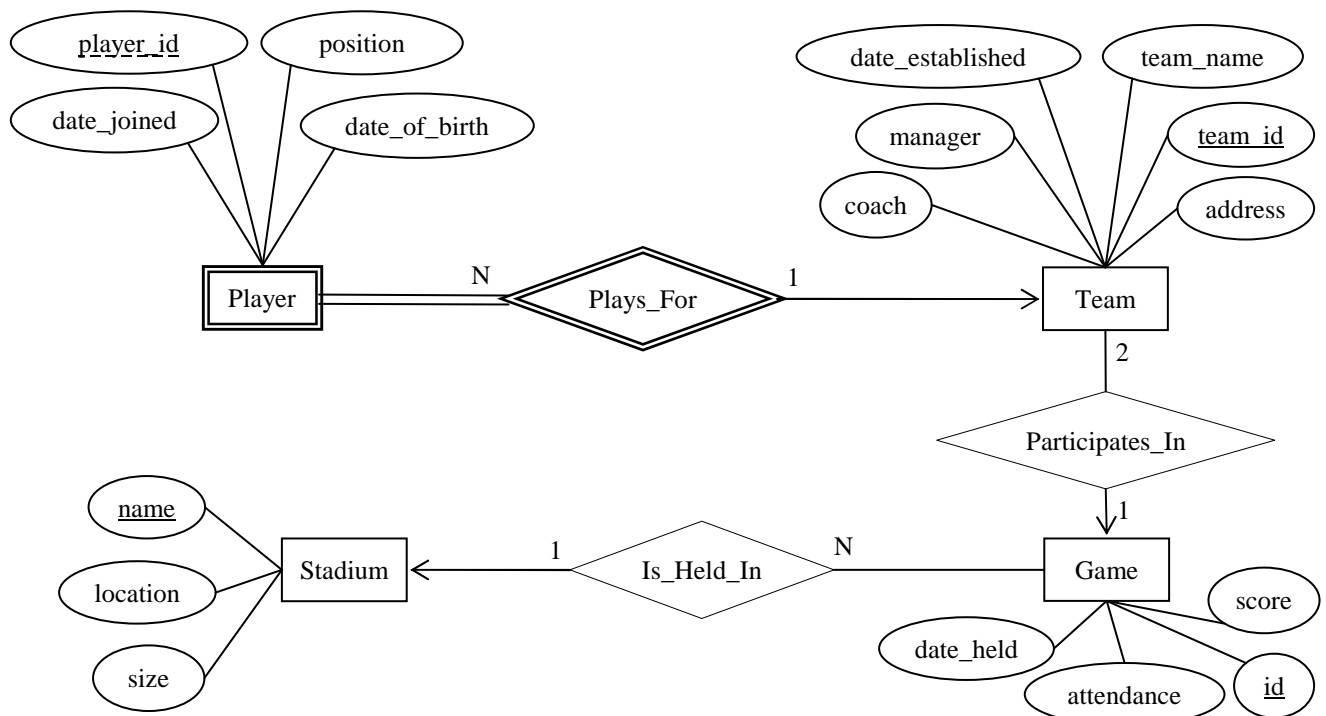
- Develop a complete E-R diagram (including cardinalities). Make reasonable assumptions during your development phases, if needed and state them clearly.
- Translate the E-R diagram into relations (tables). [2005. Marks: 5 + 3]



Train (id, total_seating_capacity)
Terminal (id, name, city)
Leg_Instance (date, train_id, no_of_compartments, no_of_available_seats)
Departure (terminal_id, date, train_id, departure_time)
Arrival (terminal_id, date, train_id, arrival_time)
Reservation (seat_id, date, train_id, seat_type, passenger_id, passenger_name, age, phone)

2.18 A database is being constructed to keep track of the teams and games of a football league. A team has a number of players. For the team, we are interested to store team id, team name, address, date established, name of manager, and name of coach. For the player, we will store player id in team, date of birth, date joined, position etc. Each team plays games against other team in a round robin fashion. For each game, we will store game id, date held, score and attendance (an attribute to designate whether the participating teams have attended the game). Games are generally taking place at various stadiums of the country. For each stadium, we will keep its size, name and location.

- i. Develop a complete E-R diagram (including cardinalities). Make reasonable assumptions during your development phases, if needed and state them clearly.
- ii. Translate the E-R diagram into relations (tables). [2003. Marks: 6 + 4]



Team (team_id, team_name, date_established, address, manager, coach)
Player (player_id, team_id, date_of_birth, date_joined, position)
Stadium (name, location, size)
Game (id, date_held, attendance, score, stadium_name)
Team_Game (team_id, game_id)

CHAPTER 3, 4

RELATIONAL MODEL & SQL

Points to be Remembered

3.1	The order in which tuples or attributes appear in a relation is irrelevant, since a relation is a set of tuples – sorted or unsorted does not matter.														
3.2	To represent string values, in RA, double quotes (" ") are used, whereas in SQL, single-quotes (' ') are used.														
3.3	Note the difference in representation of the following operators in SQL and RA: <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>SQL</td> <td>>=</td> <td><=</td> <td><></td> <td>and</td> <td>or</td> <td>not</td> </tr> <tr> <td>RA</td> <td>≥</td> <td>≤</td> <td>≠</td> <td>∧</td> <td>∨</td> <td>¬</td> </tr> </table>	SQL	>=	<=	<>	and	or	not	RA	≥	≤	≠	∧	∨	¬
SQL	>=	<=	<>	and	or	not									
RA	≥	≤	≠	∧	∨	¬									
3.4	In the projection operation, duplicate rows are eliminated in RA (as RA considers relations as <i>sets</i>); whereas SQL retains duplicate rows by default (since duplicate elimination is time consuming). To force the elimination of duplicate, a keyword <code>distinct</code> is inserted after <code>select</code> .														
3.5	SQL does not allow the use of <code>distinct</code> with <code>count(*)</code> (however, it can be used with <code>count</code> for a single attribute, e.g. <code>count(distinct A)</code>). <code>distinct</code> can be used with <code>min</code> and <code>max</code> , but result does not change.														
3.6	If a <code>where</code> clause and <code>having</code> clause appear in the same query, SQL applies the predicate in the <code>where</code> clause first. Tuples satisfying the <code>where</code> predicate are then placed into groups by the <code>group by</code> clause. SQL then applies the <code>having</code> clause, if it is present, to each group; it removes the groups that do not satisfy the <code>having</code> clause predicate. The <code>select</code> clause uses the remaining groups to generate the tuples of the result relation.														
3.7	The input to <code>sum</code> and <code>avg</code> must be a collection of numbers, but the other aggregate functions (<code>count</code> , <code>min</code> and <code>max</code>) can operate on collection of non-numeric data types, such as string, as well.														
3.8	Aggregate functions cannot be composed in SQL. Thus, we cannot use <code>max(avg(...))</code> .														
3.9	Every <i>derived</i> table must have its own alias. Wrong: <code>select * from (select x from y where p = q) where a = b;</code> Right: <code>select * from (select x from y where p = q) as new_table where a = b;</code>														
3.10	The use of a null value in arithmetic and comparison operations causes several complications. The result of any arithmetic expression involving null returns null. So <code>5 + null</code> returns null. Any comparison with null (other than <code>is null</code> and <code>is not null</code>) returns unknown. So, <code>5 < null</code> or <code>null <> null</code> or <code>null = null</code> returns unknown.														
3.11	All aggregate functions except <code>count(*)</code> ignore tuples with null values on the aggregated attributes.														
3.12	If we use an arithmetic expression in the <code>select</code> clause, the resultant attribute does not have a name.														

The Trick of Writing RA Expressions for Complex Queries

If you find that writing the RA expression of a query is getting difficult, then think of the query in terms of views. Views create multiple tables and *multiple simple* queries to perform a *single complex* query. As you're able to write the RA expressions for simple queries, you'll now be able to solve the complex queries.

For applications of this trick, see the following queries:

Complete Concepts Problem – query no. 20, 21, 23.

Theory 4.10 – query no. 4

Theory 4.11 – query no. 2 and 4

Complete Concepts Problem

Consider the database schema below:

employee (*ename*, *street*, *city*)

works (*ename*, *cname*, *salary*, *jdate*)

company (*cname*, *city*)

manages (*ename*, *mname*)

Note: A manager is also an employee of a company.

Give SQL and RA expressions for the following queries:

Imp. Level	Diff. Level	Queries
0	0	1. Find the names of all employees who work for First Bank Corporation.
0	1	2. Find the names and cities of residence of all employees who work for First Bank Corporation.
5	2	3. Find the names, street address, and cities of residence of all employees who work for First Bank Corporation and earn more than Tk. 30000.
5	2	4. Find names, street addresses and cities of residence of all employees who work under manager Sabbir and who joined before January 01, 2009.
1	1	5. Find the names of all employees in this database who live in the same city as the company for which they work.
5	5	6. Find the names of all employees who live in the same city and on the same street as do their managers.
3	3	7. Find the names of the employees living in the same city where Rahim is residing.
0	0	8. Find the names of all employees in this database who do not work for First Bank Corporation.
3	5	9. Find the names of all employees who earn more than <i>every</i> employee of Small Bank Corporation.
5	5	10. Find the names of all employees who earn more than <i>any</i> employee of Small Bank Corporation.
2	2	11. Assume the companies may be located in several cities. Find all companies located in every city in which Small Bank Corporation is located.
5	4	12. Give all employees of First Bank Corporation a 10 percent salary raise.
5	5	13. Give all managers in the database a 10% salary raise.
1	5	14. Give all managers in this database a 10 percent salary raise, unless the salary would be greater than Tk.100,000. In such cases, give only a 3 percent raise.
5	5	15. Increase the salary of employees by 10% for the companies those are located in Bogra.
2	3	16. Modify the database so that Rahim now lives in Bhola.
1	1	17. Delete all tuples in the <i>works</i> relation for employees of Small Bank Corporation.
5	3	18. Delete records from <i>works</i> that contain employees living in Rajshahi.
4	2	19. Display the average salary of each company except Square Pharma.
5	4	20. Find the company with the most employees.
4	4	21. Find the company that has the smallest payroll.
4	3	22. Find the company with payroll less than Tk. 100000.
3	5	23. Find those companies whose employees earn a higher salary, on average, than the average salary of Small Bank Corporation.

Note: the *Imp. Level* column in the above table means how much important that query is for the exam (range: 0 – 5, where 0 means *not important at all* and 5 means *most important*); and the *Diff. Level* field means how difficult the problem is (range: 0-5, where 0 means *very easy* and 5 means *very difficult*).

Sample Data

Table Name	Data																																												
<i>employee</i>	<table border="1"> <thead> <tr> <th>ename</th> <th>street</th> <th>city</th> </tr> </thead> <tbody> <tr> <td>Barkat</td> <td><i>x</i></td> <td>Bogra</td> </tr> <tr> <td>Jabbar</td> <td><i>x</i></td> <td>Comilla</td> </tr> <tr> <td>Jubayer</td> <td><i>u</i></td> <td>Faridpur</td> </tr> <tr> <td>Najmun Nahar</td> <td><i>y</i></td> <td>Sylhet</td> </tr> <tr> <td>Oronno</td> <td><i>z</i></td> <td>Dhaka</td> </tr> <tr> <td>Rafique</td> <td><i>z</i></td> <td>Rajshahi</td> </tr> <tr> <td>Rahim</td> <td><i>w</i></td> <td>Dhaka</td> </tr> <tr> <td>Sabbir</td> <td><i>v</i></td> <td>Chittagong</td> </tr> <tr> <td>Salam</td> <td><i>y</i></td> <td>Comilla</td> </tr> <tr> <td>Sharafat</td> <td><i>w</i></td> <td>Dhaka</td> </tr> </tbody> </table>	ename	street	city	Barkat	<i>x</i>	Bogra	Jabbar	<i>x</i>	Comilla	Jubayer	<i>u</i>	Faridpur	Najmun Nahar	<i>y</i>	Sylhet	Oronno	<i>z</i>	Dhaka	Rafique	<i>z</i>	Rajshahi	Rahim	<i>w</i>	Dhaka	Sabbir	<i>v</i>	Chittagong	Salam	<i>y</i>	Comilla	Sharafat	<i>w</i>	Dhaka											
ename	street	city																																											
Barkat	<i>x</i>	Bogra																																											
Jabbar	<i>x</i>	Comilla																																											
Jubayer	<i>u</i>	Faridpur																																											
Najmun Nahar	<i>y</i>	Sylhet																																											
Oronno	<i>z</i>	Dhaka																																											
Rafique	<i>z</i>	Rajshahi																																											
Rahim	<i>w</i>	Dhaka																																											
Sabbir	<i>v</i>	Chittagong																																											
Salam	<i>y</i>	Comilla																																											
Sharafat	<i>w</i>	Dhaka																																											
<i>works</i>	<table border="1"> <thead> <tr> <th>ename</th> <th>cname</th> <th>salary</th> <th>jdate</th> </tr> </thead> <tbody> <tr> <td>Rahim</td> <td>First Bank Corporation</td> <td>50000</td> <td>2008-01-01</td> </tr> <tr> <td>Barkat</td> <td>First Bank Corporation</td> <td>40000</td> <td>2007-01-01</td> </tr> <tr> <td>Salam</td> <td>First Bank Corporation</td> <td>60000</td> <td>2009-07-01</td> </tr> <tr> <td>Rafique</td> <td>Small Bank Corporation</td> <td>30000</td> <td>2009-06-08</td> </tr> <tr> <td>Sharafat</td> <td>First Bank Corporation</td> <td>80000</td> <td>2005-06-01</td> </tr> <tr> <td>Jabbar</td> <td>Small Bank Corporation</td> <td>10000</td> <td>2009-06-05</td> </tr> <tr> <td>Najmun Nahar</td> <td>Small Bank Corporation</td> <td>20000</td> <td>2009-06-30</td> </tr> <tr> <td>Oronno</td> <td>The ONE Limited</td> <td>50000</td> <td>2007-06-01</td> </tr> <tr> <td>Jubayer</td> <td>Square Pharma</td> <td>15000</td> <td>2008-01-01</td> </tr> <tr> <td>Sabbir</td> <td>Vegabond Company</td> <td>100000</td> <td>2001-01-01</td> </tr> </tbody> </table>	ename	cname	salary	jdate	Rahim	First Bank Corporation	50000	2008-01-01	Barkat	First Bank Corporation	40000	2007-01-01	Salam	First Bank Corporation	60000	2009-07-01	Rafique	Small Bank Corporation	30000	2009-06-08	Sharafat	First Bank Corporation	80000	2005-06-01	Jabbar	Small Bank Corporation	10000	2009-06-05	Najmun Nahar	Small Bank Corporation	20000	2009-06-30	Oronno	The ONE Limited	50000	2007-06-01	Jubayer	Square Pharma	15000	2008-01-01	Sabbir	Vegabond Company	100000	2001-01-01
ename	cname	salary	jdate																																										
Rahim	First Bank Corporation	50000	2008-01-01																																										
Barkat	First Bank Corporation	40000	2007-01-01																																										
Salam	First Bank Corporation	60000	2009-07-01																																										
Rafique	Small Bank Corporation	30000	2009-06-08																																										
Sharafat	First Bank Corporation	80000	2005-06-01																																										
Jabbar	Small Bank Corporation	10000	2009-06-05																																										
Najmun Nahar	Small Bank Corporation	20000	2009-06-30																																										
Oronno	The ONE Limited	50000	2007-06-01																																										
Jubayer	Square Pharma	15000	2008-01-01																																										
Sabbir	Vegabond Company	100000	2001-01-01																																										
<i>company</i>	<table border="1"> <thead> <tr> <th>cname</th> <th>city</th> </tr> </thead> <tbody> <tr> <td>Anonymous IT</td> <td>Chittagong</td> </tr> <tr> <td>Dream Tech</td> <td>Chittagong</td> </tr> <tr> <td>First Bank Corporation</td> <td>Dhaka</td> </tr> <tr> <td>JONS IT (Pvt.) Limited</td> <td>Sylhet</td> </tr> <tr> <td>Small Bank Corporation</td> <td>Dhaka</td> </tr> <tr> <td>Square Pharma</td> <td>Bogra</td> </tr> <tr> <td>The ONE Limited</td> <td>Dhaka</td> </tr> <tr> <td>Unique Softs</td> <td>Dhaka</td> </tr> <tr> <td>Unknown Systems</td> <td>Rajshahi</td> </tr> <tr> <td>Vegabond Company</td> <td>Bogra</td> </tr> </tbody> </table>	cname	city	Anonymous IT	Chittagong	Dream Tech	Chittagong	First Bank Corporation	Dhaka	JONS IT (Pvt.) Limited	Sylhet	Small Bank Corporation	Dhaka	Square Pharma	Bogra	The ONE Limited	Dhaka	Unique Softs	Dhaka	Unknown Systems	Rajshahi	Vegabond Company	Bogra																						
cname	city																																												
Anonymous IT	Chittagong																																												
Dream Tech	Chittagong																																												
First Bank Corporation	Dhaka																																												
JONS IT (Pvt.) Limited	Sylhet																																												
Small Bank Corporation	Dhaka																																												
Square Pharma	Bogra																																												
The ONE Limited	Dhaka																																												
Unique Softs	Dhaka																																												
Unknown Systems	Rajshahi																																												
Vegabond Company	Bogra																																												
<i>manages</i>	<table border="1"> <thead> <tr> <th>ename</th> <th>mname</th> </tr> </thead> <tbody> <tr> <td>Rahim</td> <td>Sharafat</td> </tr> <tr> <td>Barkat</td> <td>Sharafat</td> </tr> <tr> <td>Salam</td> <td>Sharafat</td> </tr> <tr> <td>Rafique</td> <td>Oronno</td> </tr> <tr> <td>Jabbar</td> <td>Oronno</td> </tr> <tr> <td>Najmun Nahar</td> <td>Sabbir</td> </tr> <tr> <td>Jubayer</td> <td>Sabbir</td> </tr> </tbody> </table>	ename	mname	Rahim	Sharafat	Barkat	Sharafat	Salam	Sharafat	Rafique	Oronno	Jabbar	Oronno	Najmun Nahar	Sabbir	Jubayer	Sabbir																												
ename	mname																																												
Rahim	Sharafat																																												
Barkat	Sharafat																																												
Salam	Sharafat																																												
Rafique	Oronno																																												
Jabbar	Oronno																																												
Najmun Nahar	Sabbir																																												
Jubayer	Sabbir																																												

employee (ename, street, city)
works (ename, cname, salary, jdate)
company (cname, city)
manages (ename, mname)

1. Find the names of all employees who work for First Bank Corporation.

SQL: `select ename from works where cname = 'First Bank Corporation';`

RA: $\Pi_{ename} (\sigma_{cname = \text{"First Bank Corporation"}} (\text{works}))$

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

SQL: `select ename, city from employee natural join works where cname = 'First Bank Corporation';`

RA: $\Pi_{ename, city} (\sigma_{cname = \text{"First Bank Corporation"}} (\text{employee} \bowtie \text{works}))$

3. Find the names, street address, and cities of residence of all employees who work for First Bank Corporation and earn more than Tk. 30000.

SQL: `select ename, street, city from employee natural join works where cname = 'First Bank Corporation' and salary > 30000;`

RA: $\Pi_{ename, street, city} (\sigma_{cname = \text{"First Bank Corporation"} \wedge salary > 30000} (\text{employee} \bowtie \text{works}))$

4. Find names, street addresses and cities of residence of all employees who work under manager Sabbir and who joined before January 01, 2009.

SQL: `select ename, street, city from employee natural join works natural join manages where mname = 'Sabbir' and jdate < '01-JAN-09';`

RA: $\Pi_{ename, street, city} (\sigma_{mname = \text{"Sabbir"} \wedge jdate < \text{"01-jan-09"}} (\text{employee} \bowtie \text{works} \bowtie \text{manages}))$

5. Find the names of all employees in this database who live in the same city as the company for which they work.

SQL: `select ename from employee natural join works natural join company;`

RA: $\Pi_{ename} (\text{employee} \bowtie \text{works} \bowtie \text{company})$

6. Find the names of all employees who live in the same city and on the same street as do their managers.

SQL: `select employee.ename from employee natural join manages, employee as emp where mname = emp.ename and employee.street = emp.street and employee.city = emp.city;`

RA: $\Pi_{employee.ename}$

$(\sigma_{mname = emp.ename \wedge employee.street = emp.street \wedge employee.city = emp.city} (\text{employee} \bowtie \text{manages} \times \rho_{emp} (\text{employee})))$

7. Find the names of the employees living in the same city where Rahim is residing.

SQL: `select ename from employee where city = (select city from employee where ename = 'Rahim');`

RA: $t \leftarrow \Pi_{city} (\sigma_{ename = \text{"Rahim"}} (\text{employee}))$

$\Pi_{ename} (\text{employee} \bowtie t)$

8. Find the names of all employees in this database who do not work for First Bank Corporation.

SQL: `select ename from works where cname <> 'First Bank Corporation';`

RA: $\Pi_{ename} (\sigma_{cname \neq \text{"First Bank Corporation"}} (\text{works}))$

employee (ename, street, city)
works (ename, cname, salary, jdate)
company (cname, city)
manages (ename, mname)

9. Find the names of all employees who earn more than every employee of Small Bank Corporation.

SQL: `select ename from works where salary > (
 select max(salary) from works where cname = 'Small Bank Corporation'
);`

RA: $t \leftarrow \mathcal{G} \max(\text{salary}) \text{ as } \text{max_salary} (\sigma_{\text{cname} = \text{"Small Bank Corporation"}}(\text{works}))$
 $\Pi_{\text{ename}} (\sigma_{\text{salary} > \text{max_salary}}(\text{works} \times t))$

OR, $t_1 \leftarrow \Pi_{\text{works.salary}} (\sigma_{\text{works.salary} < \text{w.salary} \text{ and } \text{w.cname} = \text{"Small Bank Corporation"}}(\text{works} \times \rho_{\text{w}}(\text{works})))$
 $t_2 \leftarrow \Pi_{\text{salary}} (\sigma_{\text{w.cname} = \text{"Small Bank Corporation"}}(\text{works})) - t_1$
 $\Pi_{\text{ename}} (\sigma_{\text{works.salary} > t_2.\text{salary}}(\text{works} \times t_2))$

10. Find the names of all employees who earn more than any employee of Small Bank Corporation.

SQL: `select ename from works where salary > (
 select min(salary) from works where cname = 'Small Bank Corporation'
);`

RA: $t \leftarrow \mathcal{G} \min(\text{salary}) \text{ as } \text{min_salary} (\sigma_{\text{cname} = \text{"Small Bank Corporation"}}(\text{works}))$
 $\Pi_{\text{ename}} (\sigma_{\text{salary} > \text{min_salary}}(\text{works} \times t))$

OR, $t_1 \leftarrow \Pi_{\text{works.salary}} (\sigma_{\text{works.salary} > \text{w.salary} \text{ and } \text{w.cname} = \text{"Small Bank Corporation"}}(\text{works} \times \rho_{\text{w}}(\text{works})))$
 $t_2 \leftarrow \Pi_{\text{salary}} (\sigma_{\text{w.cname} = \text{"Small Bank Corporation"}}(\text{works})) - t_1$
 $\Pi_{\text{ename}} (\sigma_{\text{works.salary} > t_2.\text{salary}}(\text{works} \times t_2))$

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

SQL: `select cname from company where city in (
 select city from company where cname = 'Small Bank Corporation'
);`

RA: $\text{city} \leftarrow \Pi_{\text{city}} (\sigma_{\text{cname} = \text{"Small Bank Corporation"}}(\text{company}))$
 $\Pi_{\text{cname}} (\text{company} \bowtie \text{city})$

OR, $\Pi_{\text{cname}} (\text{company} \div (\Pi_{\text{city}} (\sigma_{\text{cname} = \text{"Small Bank Corporation"}}(\text{company}))))$

12. Give all employees of First Bank Corporation a 10 percent salary raise.

SQL: `update works set salary = salary * 1.1 where cname = 'First Bank Corporation';`

RA: $t \leftarrow \Pi_{\text{ename, cname, salary} * 1.1, \text{jdate}} (\sigma_{\text{cname} = \text{"First Bank Corporation"}}(\text{works}))$
 $\text{works} \leftarrow t \cup (\text{works} - \sigma_{\text{cname} = \text{"First Bank Corporation"}}(\text{works}))$

13. Give all managers in the database a 10% salary raise.

SQL: `update works set salary = salary * 1.1 where ename in (
 select distinct mname from manages
);`

RA: $t_1 \leftarrow \Pi_{\text{works.ename, cname, salary, jdate}} (\sigma_{\text{works.ename} = \text{mname}}(\text{works} \times \text{manages}))$
 $t_2 \leftarrow \Pi_{\text{works.ename, cname, salary} * 1.1, \text{jdate}} (t_1)$
 $\text{works} \leftarrow (\text{works} - t_1) \cup t_2$

employee (ename, street, city)
works (ename, cname, salary, jdate)
company (cname, city)
manages (ename, mname)

14. Give all managers in this database a 10 percent salary raise, unless the salary would be greater than Tk.100,000. In such cases, give only a 3 percent raise.

SQL: `update works set salary = case`
 `when salary * 1.1 > 100000 then salary * 1.03`
 `else salary * 1.1`
 `end`
 `where ename in (`
 `select distinct mname from manages`
 `);`

RA: $t_1 \leftarrow \Pi_{works.ename, cname, salary, jdate} (\sigma_{works.ename = mname} (works \times manages))$
 $t_2 \leftarrow \Pi_{works.ename, cname, salary * 1.03, jdate} (\sigma_{t_1.salary * 1.1 > 100000} (t_1))$
 $t_2 \leftarrow t_2 \cup (\Pi_{works.ename, cname, salary * 1.1, jdate} (\sigma_{t_1.salary * 1.1 \leq 100000} (t_1)))$
 $works \leftarrow (works - t_1) \cup t_2$

15. Increase the salary of employees by 10% for the companies those are located in Bogra.

SQL: `update works set salary = salary * 1.1 where cname in (`
 `select cname from company where city = 'Bogra'`
 `);`

RA: $t_1 \leftarrow \Pi_{ename, cname, salary, jdate} (\sigma_{city = "Bogra"} (works \bowtie company))$
 $t_2 \leftarrow \Pi_{ename, cname, salary * 1.1, jdate} (t_1)$
 $works \leftarrow (works - t_1) \cup t_2$

16. Modify the database so that Rahim now lives in Bhola.

SQL: `update employee set city = 'Bhola' where ename = 'Rahim';`

RA: $t \leftarrow \Pi_{ename, street, "Bhola"} (\sigma_{ename = "Rahim"} (employee))$
 $works \leftarrow (works - (\sigma_{ename = "Rahim"} (employee))) \cup t$

17. Delete all tuples in the works relation for employees of Small Bank Corporation.

SQL: `delete from works where cname = 'Small Bank Corporation';`

RA: $works \leftarrow works - (\sigma_{cname = "Small Bank Corporation"} (works))$

18. Delete records from works that contain employees living in Rajshahi.

SQL: `delete from works where ename in (`
 `select ename from employee where city = 'Rajshahi'`
 `);`

RA: $t \leftarrow \Pi_{ename} (\sigma_{city = "Rajshahi"} (employee))$
 $works \leftarrow works - \Pi_{ename, cname, salary, jdate} (works \bowtie t)$

19. Display the average salary of each company except Square Pharma.

SQL: `select cname, avg(salary) from works where cname <> 'Square Pharma' group by cname;`

RA: $cname \curvearrowright avg(salary) (\sigma_{cname \neq "Square Pharma"} (works))$

employee (ename, street, city)
works (ename, cname, salary, jdate)
company (cname, city)
manages (ename, mname)

20. Find the company with the most employees.

SQL: `select cname, count(distinct ename) from works group by cname
 having count(distinct ename) >= all (
 select count(distinct ename) from works group by cname
);`

RA: $t_1 \leftarrow \text{cname} \mathcal{G} \text{count}(\text{ename}) \text{ as num_employees} (\text{works})$
 $t_2 \leftarrow \mathcal{G} \text{max}(\text{num_employees}) \text{ as num_employees} (t_1)$
 $\Pi_{\text{cname}} (t_1 \bowtie t_2)$

21. Find the company that has the smallest payroll¹. [Similar to query 20]

SQL: `select cname, sum(salary) from works group by cname
 having sum(salary) <= all (
 select sum(salary) from works group by cname
);`

RA: $t_1 \leftarrow \text{cname} \mathcal{G} \text{sum}(\text{salary}) \text{ as payroll} (\text{works})$
 $t_2 \leftarrow \mathcal{G} \text{min}(\text{payroll}) \text{ as payroll} (t_1)$
 $\Pi_{\text{cname}} (t_1 \bowtie t_2)$

22. Find the company with payroll less than Tk. 100000.

SQL: `select cname, sum(salary) from works group by cname having sum(salary) < 100000;`

RA: $t \leftarrow \text{cname} \mathcal{G} \text{sum}(\text{salary}) (\text{works})$
 $\Pi_{\text{cname}} (\sigma_{\text{payroll} < 100000} (\rho_{\text{c_payroll}} (\text{cname}, \text{payroll}) (t)))$

23. Find those companies whose employees earn a higher salary, on average, than the average salary of Small Bank Corporation.

SQL: `select cname from works group by cname
 having avg(salary) > (
 select avg(salary)
 from works
 where cname = 'Small Bank Corporation'
);`

RA: $t_1 \leftarrow \text{cname} \mathcal{G} \text{avg}(\text{salary}) (\text{works})$
 $t_2 \leftarrow \sigma_{\text{cname} = \text{'Small Bank Corporation'}} (t_1)$
 $\Pi_{t_3.\text{cname}} (\sigma_{t_3.\text{avg_salary} > \text{small-bank.avg_salary}} (\rho_{t_3(\text{cname}, \text{avg_salary})} (t_1) \times \rho_{\text{small-bank}(\text{cname}, \text{avg_salary})} (t_2)))$

¹ **Payroll:** The total amount of money paid by a company as salary for all the employers.

General Structure of Query Statements

Legend:	
<u>Choose this or this</u>	Choose <i>either</i> the statement <i>above</i> the line <i>or</i> the statement <i>below</i> the line at a time
[<i>optional</i>]	You can use it or leave it
”	Comma-separated list

General Structure of CREATE TABLE statement:

```
CREATE TABLE table-name (  
    column_name column_data_type [PRIMARY KEY]  
    [NOT NULL]  
    PRIMARY KEY (column-name)  
    CONSTRAINT constraint-name FOREIGN KEY (column-name) REFERENCES table-name (column-name)  
    CHECK (expression)  
);
```

column_data_types:

1. **CHAR** (*number_of_characters*) example: CHAR(30)
2. **VARCHAR** (*maximum_number_of_characters*) example: VARCHAR(255)
3. **INTEGER** (*number_of_digits*) example: INTEGER(10)
4.

<u>NUMERIC</u> <u>DECIMAL</u> <u>FLOAT</u> <u>DOUBLE PRECISION</u>	(<i>total_number_of_digits_including_decimals, number_of_decimal_digits</i>)
	example: DECIMAL(5, 2) [for 999.99]
5. **DATE**
6. **TIME**
7. **DATETIME**

Example of CREATE TABLE statement:

```
create table account (  
    account_no char(5),  
    branch_name varchar(15),  
    balance number(10,2)not null,  
    constraint a_pk primary key(account_no),  
    constraint a_fk foreign key (branch_name) references branch(branch_name),  
    constraint a_chk1 check (balance>=0),  
    constraint a_chk2 check (account_no like 'A-%')  
);
```

General Structure of DROP TABLE statement:

```
DROP TABLE table-name ;
```

General Structure of INSERT statement:

```
INSERT INTO table-name [(column-names,,)] VALUES [(values,,)];
```

Examples of INSERT statement:

```
insert into account values ('a-101', 'downtown', 500);  
insert into account (account_no, branch_name, balance)  
  values ('a-101', 'downtown', 500);
```

General Structure of SELECT statement:

```
SELECT [DISTINCT] column-name_or_function [AS alias],  
      *  
      table-name [AS alias] ,,  
FROM table-name [AS alias] LEFT OUTER JOIN table-name [AS alias] ON expression  
table-name [AS alias] RIGHT OUTER JOIN table-name [AS alias] USING (column-name,,)  
table-name [AS alias] FULL OUTER JOIN table-name [AS alias]  
table-name [AS alias] NATURAL JOIN table-name [AS alias]  
table-name [AS alias] NATURAL LEFT OUTER JOIN table-name [AS alias]  
table-name [AS alias] NATURAL RIGHT OUTER JOIN table-name [AS alias]  
[WHERE expression]  
[GROUP BY column-name_or_alias_or_function]  
[HAVING expression]  
[ORDER BY column-name_or_alias_or_function [ASC / DESC]  
[LIMIT number-of-rows  
start-index, number-of-rows-from-start-index ];
```

Example of SELECT statement:

```
select account_no, avg(balance) as average_balance  
from account left outer join depositor using (account_no)  
where balance > 1000  
group by branch_name  
having count(account_no) >= 10  
order by average_balance desc  
limit 0, 100;
```

General Structure of UPDATE statement:

```
UPDATE table-name [AS alias] ,,  
table-name [AS alias] LEFT OUTER JOIN table-name [AS alias] ON expression  
table-name [AS alias] RIGHT OUTER JOIN table-name [AS alias] USING (column-name,,)  
table-name [AS alias] FULL OUTER JOIN table-name [AS alias]  
table-name [AS alias] NATURAL JOIN table-name [AS alias]  
table-name [AS alias] NATURAL LEFT OUTER JOIN table-name [AS alias]  
table-name [AS alias] NATURAL RIGHT OUTER JOIN table-name [AS alias]  
SET column-name = column-value ,,  
[WHERE expression];
```

Example of UPDATE statement:

```
update account set balance = balance * 1.1 where balance >= 100000;
```

Theories

3.1	<p>List two reasons why <i>null</i> values might be introduced into the database.</p> <p><i>Nulls</i> may be introduced into the database because the actual value is either unknown or does not exist. For example, an employee whose address has changed and whose new address is not yet known should be retained with a <i>null</i> address. If employee tuples have a composite attribute <i>dependents</i>, and a particular employee has no dependents, then that tuple's <i>dependents</i> attribute should be given a <i>null</i> value.</p>
3.2	<p>List two reasons why we may choose to define a view.</p> <ol style="list-style-type: none">1. Security conditions may require that the entire logical database be not visible to all users.2. We may wish to create a personalized collection of relations that is better matched to a certain user's intuition than is the actual logical model.
3.3	<p>List two major problems with processing update operations expressed in terms of views.</p> <p>Views present significant problems if updates are expressed with them. The difficulty is that a modification to the database expressed in terms of a view must be translated to a modification to the actual relations in the logical model of the database.</p> <ol style="list-style-type: none">1. Since the view may not have all the attributes of the underlying tables, insertion of a tuple into the view will insert tuples into the underlying tables, with those attributes not participating in the view getting null values. This may not be desirable, especially if the attribute in question is part of the primary key of the table.2. If a view is a join of several underlying tables and an insertion results in tuples with nulls in the join columns, the desired effect of the insertion will not be achieved. In other words, an update to a view may not be expressible at all as updates to base relations.
3.4	<p>What are the conditions of updating a view?</p> <ol style="list-style-type: none">1. The <code>from</code> clause has only one database relation.2. The <code>select</code> clause contains only attribute names of the relation and does not have any expression, aggregates or <code>distinct</code> specifications.3. Any attribute not listed in the <code>select</code> clause can be set to null.4. The query does not have a <code>group by</code> or <code>having</code> clause.
3.5	<p>What is materialized view? [In-course-1, 2008; 2005. Marks: 1]</p> <p>A view which makes sure that if the actual relations used in the view definition change, the view is kept up-to-date, is called materialized view.</p>
3.6	<p>Define the following:</p> <ol style="list-style-type: none">1. Domain2. Atomic Domain3. Non-Atomic Domain4. Tuple Variable <ol style="list-style-type: none">1. Domain: For each attribute, there is a set of permitted values, which are called domain (<i>D</i>) of that attribute. For the attribute <i>branch-name</i>, the domain is the set of all branch names.2. Atomic Domain: A domain is atomic if elements of the domain are considered to be indivisible parts. Example: set of integers: 23, 45, 5, 78 etc.3. Non-Atomic Domain: If elements of a domain can be divided into several parts, the domain is called non-atomic domain. Example: set of all sets of integers: {23, 12, 4; 5, 65, 4; 34, 23, 98}, employee-id: HR001, IT005

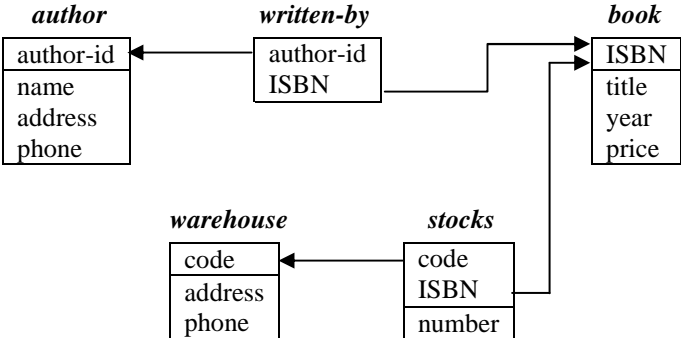
	<p>4. Tuple Variable: A tuple variable is a variable whose domain is the set of all tuples. For example, $t[\text{account-number}] = \text{"A-101"}$, $t[\text{branch-name}] = \text{"Mirpur"}$. Alternatively, $t[1]$, $t[2]$ denote the value of tuple t on first and second attributes and so on.</p>
3.7	<p>What are the fundamental operations used in Relational Algebra? What are the conditions for set (union, set-intersect and set-difference) operations in RA? [2005. Marks: 1 + 1] OR, What are the conditions for set operations to be valid? [In-course 1, 2008. Marks: 1]</p> <p>The fundamental operations used in Relational Algebra are:</p> <ol style="list-style-type: none"> 1. Select (unary) 2. Project (unary) 3. Rename (unary) 4. Cartesian Product (binary) 5. Union (binary) 6. Set-Difference (binary) <p>The conditions for set operations are:</p> <ol style="list-style-type: none"> 1. The relations must be of the same arity. That means they must have the same number of attributes. 2. The domains of i^{th} attribute of the first set and the i^{th} attribute of the second set must be the same, for all i.
3.8	<p>What are the conditions for insertion?</p> <p>The conditions for insertion are:</p> <ol style="list-style-type: none"> 1. The attribute values for inserted tuples must be members of the attribute's domain. 2. Tuples inserted must be of the same arity.
3.9	<p>Why set-intersection operation is not included in fundamental relational algebra operations? [In-course 2007; 2007. Marks: 1]</p> <p>Because set-intersection operation can be done using fundamental operations. If r_1 and r_2 are two sets, then their intersection can be expressed as:</p> $r_1 \cap r_2 = r_1 - (r_1 - r_2) = r_2 - (r_2 - r_1)$
3.10	<p>Give an example of generalized projection. [In-course 2007, Marks: 1] OR, Give a relational algebra expression to represent generalized projection. [In-course 2008, Marks: 1]</p> <p>SQL: <code>select student_name, marks + 5 from result;</code> RA: $\Pi_{\text{student_name, marks} + 5}(\text{result})$</p>
3.11	<p>Let $r(R)$ and $s(S)$ be two relations. Give the relational algebra expression for natural join (\bowtie) and the outer joins (\rceil, $\bowtie\lrcorner$, $\lrcorner\bowtie$) of the said relations. [2005. Marks: 2]</p> <p>$r \bowtie s = \Pi_{R \cup S} (\sigma_{r.A_1 = s.A_1 \wedge r.A_2 = s.A_2 \dots \wedge r.A_n = s.A_n} (r \times s))$, where $R \cap S = \{A_1, A_2, \dots, A_n\}$</p> <p>$r \rceil s = (r \bowtie s) \cup (r - \Pi_R (r \bowtie s)) \times \{(\text{null}, \text{null}, \dots, \text{null})\}$</p> <p>$r \bowtie\lrcorner s = (r \bowtie s) \cup (s - \Pi_S (r \bowtie s)) \times \{(\text{null}, \text{null}, \dots, \text{null})\}$</p> <p>$r \lrcorner\bowtie s = (r \bowtie s) \cup (r - \Pi_R (r \bowtie s)) \times \{(\text{null}, \text{null}, \dots, \text{null})\} \cup (s - \Pi_S (r \bowtie s)) \times \{(\text{null}, \text{null}, \dots, \text{null})\}$</p>
3.12	<p>The outer-join operations extend the natural-join operation so that tuples from the participating relations are not lost in the result of the join. Describe how the theta join operation can be extended so that tuples from the left, right, or both relations are not lost from the result of a theta join.</p>

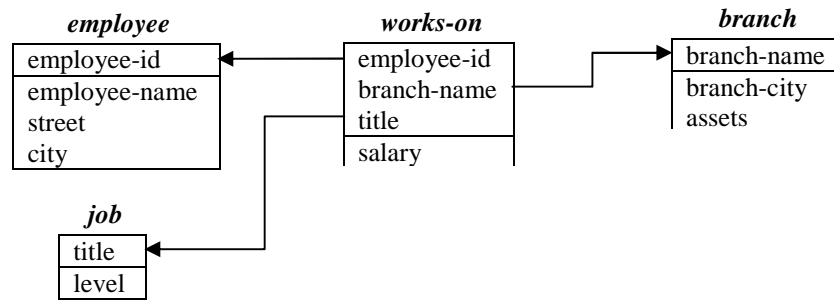
	$r \bowtie_{\theta} s = (r \bowtie s) \cup (r - \prod_R (r \bowtie_{\theta} s)) \times \{(null, null, \dots, null)\}$ $r \ltimes_{\theta} s = (r \bowtie s) \cup (s - \prod_S (r \bowtie_{\theta} s)) \times \{(null, null, \dots, null)\}$ $r \overline{\bowtie}_{\theta} s = (r \bowtie s) \cup (r - \prod_R (r \bowtie_{\theta} s)) \times \{(null, null, \dots, null)\} \cup (s - \prod_S (r \bowtie_{\theta} s)) \times \{(null, null, \dots, null)\}$
3.13	<p>With example, show the difference between Cartesian product (\times) and natural join (\bowtie). [2005. Marks: 2]</p> <p>Let, $R_1 = (A, B)$ and $R_2 = (B, C)$ be two relation schema.</p> <p>Again, let $r_1(R_1) = \{\{a, 1\}, \{b, 2\}\}$ and $r_2(R_2) = \{\{1, x\}, \{2, y\}\}$</p> <p>Then, $r_1 \times r_2 = \{\{a, 1, 1, x\}, \{a, 1, 2, y\}, \{b, 2, 1, x\}, \{b, 2, 2, y\}\}$</p> <p>And $r_1 \bowtie r_2 = \{\{a, 1, x\}, \{b, 2, y\}\}$</p> <p>That is, the Cartesian product operation results in all the combinations of all the tuples from both tables, whereas the natural join operation results in only the tuple combinations from both tables where the values of the common attributes (in this example, the attribute 'B') are the same.</p>
3.14	<p>For a given relation schema, <i>works</i> (<i>employee_name</i>, <i>company_name</i>, <i>salary</i>), give a relational algebra expression using all aggregate functions where the grouping is done on company name. [2007, Marks: 1]</p> <p>$company_name \mathcal{G} \text{ sum}(\text{salary}), \text{ avg}(\text{salary}), \text{ max}(\text{salary}), \text{ min}(\text{salary}), \text{ count}(\text{employee_name}) (\text{works})$</p>
3.15	<p>Give the equivalent relational algebra expression of the following SQL form: select A_1, A_2, \dots, A_n from r_1, r_2, \dots, r_n where P [2005. Marks: 1]</p> <p>$\prod_{A_1, A_2, \dots, A_n} (\sigma_P (r_1 \times r_2 \dots \times r_n))$</p>
3.16	<p>Write short notes on natural join, theta join and aggregate functions.</p> <p>Natural Join:</p> <p>The natural join is a binary operation that allows us to combine certain selection and a Cartesian product into one operation. It is denoted by the “join” symbol \bowtie.</p> <p>The natural join operation forms:</p> <ol style="list-style-type: none"> i) A Cartesian product of two arguments ii) Performs a selection forcing equality on those attributes that appear in both relation schemas iii) Removes duplicate attributes <p>Theta Join:</p> <p>The theta join operation is an extension to the natural join operation that allows us to combine a selection and a Cartesian product into a single operation. Consider relations $r(R)$ and $s(S)$; let θ be predicate on attributes in the schema $R \cup S$. The theta join operation is defined as follows:</p> $R \bowtie_{\theta} S = \sigma_{\theta} (r \times s)$ <p>Aggregate Functions:</p> <p>Aggregate functions take a collection of values and return a single value as a result. It is denoted by calligraphic \mathcal{G}, \mathcal{G}. For a collection of values $\{1, 1, 3, 4, 4, 11\}$:</p> <ol style="list-style-type: none"> 1. sum returns the sum of the values: 24 2. avg returns the average of the values: 4 3. count returns the number of the elements in the collection: 6 4. min returns the minimum value of the collection: 1 5. max returns the maximum value of the collection: 11

3.17	<p>With example, explain the importance of outer joins. [In-course 2007, Marks: 2]</p> <p>When joining two or more tables, if we want to keep all the records from one table and want to know which records from the other tables don't match with them, then outer join can be used to solve the problem easily.</p> <p>For example, if we want to know which records in two tables (e.g., x and y) do not match, then we can write the following query using outer join:</p> <pre>select * from x natural full outer join y where x.some_attribute is null or y.some_attribute is null;</pre>
3.18	<p>Let R = (A, B, C); and let r₁ and r₂ both be relations on schema R. Give an expression in SQL that is equivalent to each of the following queries. [2003. Marks: 4]</p> <ol style="list-style-type: none"> r₁ ∪ r₂ r₁ ∩ r₂ r₁ - r₂ Π_{AB}(r₁) ⋈ Π_{BC}(r₂) <ol style="list-style-type: none"> select * from r1 union select * from r2; select * from r1 intersect select * from r2 ; select * from r1 minus select * from r2; select * from (select A, B from r1) as x natural join (select B, C from r2) as y;²
3.19	<p>Give names of the aggregate functions that ignore null values in their input collection. [2004. Marks: 1]</p> <p>sum, avg, min, max</p>
3.20	<p>What aggregate functions can be used for string type data? [In-course 1, 2008]</p> <p>count, min, max</p>
3.21	<p>With examples define the terms Superkey, Candidate Key and Primary Key. [2006, Marks: 3]</p> <p>Superkey:</p> <p>A superkey is a set of one or more attributes that, taken collectively, allow us to identify uniquely a tuple in the relation. For example:</p> $\text{Branch_schema} = (\text{branch_name}, \text{branch_city}, \text{assets})$ <p>In Branch_schema above, {branch_name}, {branch_name, branch_city}, {all attributes} are all superkeys.</p> <p>Formal definition:³ Let R be a relation schema. If it is said that a subset K of R is a superkey of R, it restricts consideration to relations r(R) in which no two distinct tuples have the same values on all attributes in K. That is, if t₁ and t₂ are in r and t₁ ≠ t₂, then t₁[K] ≠ t₂[K].</p> <p>Candidate Key:</p> <p>The superkey, for which no proper subset is a superkey, is a candidate key.</p> <p>For example, in Branch_schema above, {branch_name} is a candidate key.</p> <p>Primary Key:</p> <p>The primary key is a candidate key that is chosen by the database designer as the principal means of identifying tuples within a relation.</p> <p>In the Branch_schema above, {branch_name} is a primary key.</p>

² **Note:** Every derived table must have its own alias. So, the aliases x and y must be put to execute the query successfully.

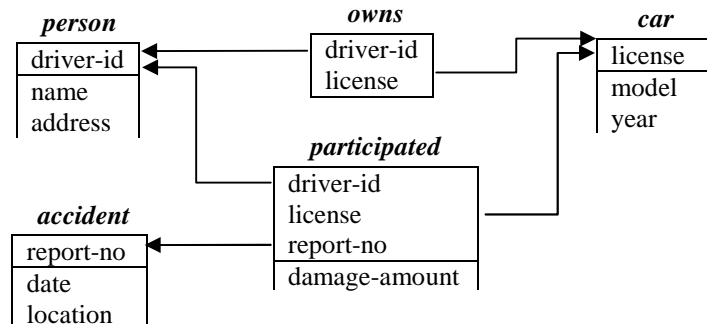
³ **Note:** At exam, don't write formal definitions if the marks are little, for example in this question (only 3 marks for 3 definitions).

3.22	<p>Define foreign key.</p> <p>A relation schema R_2 may include among its attributes the primary key of another relation schema R_1. This attribute is called a foreign key from R_2, referencing R_1. The relation r_2 is also called the referencing relation of the foreign key dependency and r_1 is called the referenced relation of the foreign key.</p> <p>The attribute <i>branch_name</i> in <i>Account-schema</i> is a foreign key from <i>Account_schema</i> referencing <i>Branch_schema</i>.</p> <p>Formal definition: Let $r_1(R_1)$ and $r_2(R_2)$ be relations with primary keys K_1 and K_2, respectively. It is said that a subset α of R_2 is a foreign key referencing K_1 in r_1 if it is required that, for every t_2 in r_2, there must be a tuple t_1 in r_1 such that $t_1[k_1] = t_2[\alpha]$.</p>
3.23	<p>Identify the relations among primary key, candidate key and super key. [2003. Marks: 3]</p> <p>Primary Key \subseteq Candidate Key \subseteq Super Key</p>
3.24	<p>Let $R = (P, Q, R, S)$. If PQ and QS can uniquely identify a tuple in the relation $r(R)$ separately, how many superkeys, candidate keys and primary keys are there? [In-course 1, 2008. Marks: 2]</p> <p>Super Keys: 6 – {P, Q}, {P, Q, R}, {P, Q, S}, {Q, S}, {Q, R, S}, {P, Q, R, S}</p> <p>Candidate Keys: 2 – {P, Q}, {Q, S}</p> <p>Primary Key: 1 – either {P, Q}, or {Q, S}</p>
3.25	<p>Why do we need the rename operation?</p> <ol style="list-style-type: none"> Two relations in the <code>from</code> clause may have attributes with the same name, so an attribute name is duplicated in the result. If we use an arithmetic expression in the <code>select</code> clause, the resultant attribute does not have a name. If an attribute name can be derived from the base relation, we may want to change the attribute name in the result to make it more meaningful.
3.26	<p>Give the schema diagram for the following database: [2006, Marks: 2]</p> <p><i>book</i> (<u>ISBN</u>, title, year, price) <i>author</i> (<u>author-id</u>, name, address, url) <i>warehouse</i> (<u>code</u>, address, phone) <i>written-by</i> (<u>author-id</u>, ISBN) <i>stocks</i> (<u>code</u>, ISBN, number)</p>  <pre> graph TD author[author: author-id, name, address, phone] written_by[written-by: author-id, ISBN] book[book: ISBN, title, year, price] warehouse[warehouse: code, address, phone] stocks[stocks: code, ISBN, number] author --> written_by written_by --> book written_by --> stocks warehouse --> stocks </pre>
3.27	<p>Draw the schema diagram for the following part of the bank database: [In-course 1, 2008; In-course 2, 2007. Marks: 1.5]</p> <p><i>employee</i> (<u>employee-id</u>, employee-name, street, city) <i>branch</i> (<u>branch-name</u>, branch-city, assets) <i>job</i> (<u>title</u>, level) <i>works-on</i> (<u>employee-id</u>, <u>branch-name</u>, <u>title</u>, salary)</p>



3.28 Give the schema diagram for the following part of database: [2004. Marks: 2]

person (driver-id, name, address)
car (license, model, year)
accident (report-no, date, location)
owns (driver-id, license)
participated (driver-id, license, report-no, damage-amount)



4.1 What are the join types and conditions that are permitted in SQL? [2005. Marks: 2]

Join types: inner join, left outer join, right outer join, full outer join.
Join conditions: natural, on <predicate>, using (A₁, A₂, ..., A_n).

4.2 Show that, in SQL, $x \not\in S$ is identical to $\text{not in } S$.

Let the set S denote the result of an SQL subquery. We compare $(x \not\in S)$ with $(x \text{ not in } S)$. If a particular value x_1 satisfies $(x_1 \not\in S)$ then for all elements y of S $x_1 \neq y$. Thus, x_1 is not a member of S and must satisfy $(x_1 \text{ not in } S)$. Similarly, suppose there is a particular value x_2 which satisfies $(x_2 \text{ not in } S)$. It cannot be equal to any element w belonging to S , and hence $(x_2 \not\in S)$ will be satisfied. Therefore, the two expressions are equivalent.

4.3 Why duplicates are retained in SQL? [2004. Marks: 1]

Duplicates are retained in SQL because:
 1. Eliminating them is costly.
 2. Retaining duplicates is important in computing sum or average.

4.4 What is the difference between 'Embedded SQL' and 'Dynamic SQL'? [2004. Marks: 2]

Dynamic SQL component allows programs to construct and submit SQL queries at run time. In contrast, embedded SQL statements must be completely present at compile time; they are compiled by the embedded SQL preprocessor.

4.5 Describe the circumstances in which you would choose to use embedded SQL rather than SQL alone or only a general-purpose programming language.

Writing queries in SQL is typically much easier than coding the same queries in a general-purpose programming language. However not all kinds of queries can be written in SQL. Also non-declarative actions such as printing a report, interacting with a user, or sending the results of a query to a graphical user interface cannot be done from within SQL. Under circumstances in which we want the best of both worlds, we can choose embedded SQL or dynamic SQL, rather than using SQL alone or using only a general-purpose programming language.

Embedded SQL has the advantage of programs being less complicated since it avoids the clutter of the ODBC or JDBC function calls, but requires a specialized preprocessor.

4.6

Consider the database schema below:

employee (*ename*, *street*, *city*)

emp_company (*ename*, *cname*, *salary*, *jdate*)

company (*cname*, *city*)

manager (*ename*, *mname*, *shift*)

Note: A manager is also an employee of a company.

Give SQL and RA expressions for the following queries: [In-course-1, 2007. Marks: 2.5 each.]

1. Find names, street addresses and cities of residence of all employees who work under manager Sabbir and who joined before January 01, 2006.
2. Find the names of the employees living in the same city where Rahim is residing.
3. Display the average salary of each company except Square Pharma.
4. Increase the salary of employees by 10% for the companies those are located in Bogra.
5. Delete records from *emp_company* that contain employees living in Rajshahi.

1. Similar to Complete Concepts Problem – Query No. 4
2. Similar to Complete Concepts Problem – Query No. 7
3. Similar to Complete Concepts Problem – Query No. 19
4. Similar to Complete Concepts Problem – Query No. 15
5. Similar to Complete Concepts Problem – Query No. 18

4.7

Consider the database schema below:

employee (*employee-id*, *employee-name*, *street*, *city*)

branch (*branch-name*, *branch-city*, *assets*)

job (*title*, *level*)

works (*employee-id*, *branch-name*, *title*, *salary*)

Give SQL and RA expressions for the following queries: [2007; Marks: 2.5 each.]

1. Find names, street addresses and cities of residence and job level of all employees who work for *Dhanmondi* branch and earn more than Tk. 10000.
2. Find the no. of employees and their total salaries for the branches located at *Khulna* city.
3. Give all *Executives* of *Mirpur* branch a 10 percent salary raise.
4. Find the branches with payroll less than Tk. 100000.

1. Similar to Complete Concepts Problem – Query No. 3
2. SQL: `select count(employee-id), sum(salary) from branch natural join works where branch-city = 'Khulna' group by branch-name;`
RA: `branch-name Gcount(employee-id), sum(salary) (σbranch-city = "Khulna" (branch ⋈ works))`
3. Similar to Complete Concepts Problem – Query No. 12
4. Similar to Complete Concepts Problem – Query No. 22

4.8

Consider the database schema below:

employee (*person-name*, *street*, *city*)

works (*person-name*, *company-name*, *salary*)

company (*company-name*, *city*)

manages (*person-name*, *manager-name*)

Note: A manager is also an employee of a company.

Give SQL and RA expressions for the following queries: [2006; Marks: 2.5 each.]

1. Find names, street addresses and cities of residence of all employees who work for First Bank Corporation and earn more than Tk. 30000.

2. Find all employees in the database who earn more than any employee of Medium Bank Corporation.
3. Give all managers of First Bank Corporation in the database a 10% salary raise.
4. Find those companies whose employees earn a higher salary, on average, than the average salary of Small Bank Corporation.
5. Find the company that has the smallest payroll.

1. Similar to Complete Concepts Problem – Query No. 3
2. Similar to Complete Concepts Problem – Query No. 10
3. Similar to Complete Concepts Problem – Query No. 12
4. Similar to Complete Concepts Problem – Query No. 23
5. Similar to Complete Concepts Problem – Query No. 21

4.9

Consider the database schema below:

client (client-no, name, address, city)

product (product-no, description, profit-percent, qty-in-hand, reorder-level, cost-price)

salesman (salesman-no, name, address, city, sale-amt)

salesorder (order-no, order-date, client-no, del-add, salesman-no, del-date, order-status)

order-detail (order-no, product-no, qty-ordered, qty-delivered)

Give SQL and RA expressions for the following queries: [2005; Marks: 2.5 each.]

1. Find the list of all clients who stay in cities Dhaka or Khulna.
2. Find the products with their description whose selling price is greater than 2000 and less than or equal to 5000. [Selling price can be found from cost-price and profit-percent]
3. Find the total ordered and delivered quantity for each product with a product range of P0035 to P0056.
4. Find the clients with their names and order numbers whose orders are handled by the salesman Mr. X.
5. Find the product no and description of non-moving products, i.e., products not being sold.

1. SQL: `select name from client where city = 'Dhaka' or city = 'Khulna';`
OR: `select name from client where city in ('Dhaka', 'Khulna');`

RA: $\Pi_{name} (\sigma_{city = "Dhaka" \vee city = "Khulna"} (client))$

2. SQL: `select product-no, description from product where (profit-percent / 100 * cost-price + cost-price) > 2000 and (profit-percent / 100 * cost-price + cost-price) <= 5000;`

RA: $\Pi_{product-no, description} (\sigma_{(profit-percent / 100 * cost-price + cost-price) > 2000 \wedge (profit-percent / 100 * cost-price + cost-price) \leq 5000} (product))$

3. SQL: `select sum(qty-ordered), sum(qty-delivered) from order-detail where product-no between 'P0035' and 'P0056' group by product-no;`

RA: $product-no \mathcal{G} sum(qty-ordered), sum(qty-delivered) (\sigma_{product-no \geq "P0035" \wedge product-no \leq "P0056"} (order-detail))$

4. SQL: `select client.name, order-no from client, salesman, salesorder where client.client-no = salesorder.client-no and salesman.salesman-no = salesorder.salesman-no and salesman.name = 'Mr. X';`

RA: $\Pi_{client.name, order-no} (\sigma_{client.client-no = salesorder.client-no \wedge salesman.salesman-no = salesorder.salesman-no \wedge salesman.name = "Mr. X"} (client \times salesman \times salesorder))$

5. SQL: `select product-no, description from product minus select product-no, description from product natural join order-detail;`

RA: $\Pi_{\text{product-no, description}}(\text{product}) - \Pi_{\text{product-no, description}}(\text{product} \bowtie \text{order-detail})$

OR, SQL: `select product-no, description from product left outer join order-detail using (product-no) where order-no = null;`

RA: $\Pi_{\text{product-no, description}}(\sigma_{\text{order-no} = \text{null}}(\text{product} \boxtimes \text{order-detail}))$

4.10 Consider the part of a bank database schema below:

Worker (worker-id, worker-name, hourly-rate, skill-type, supervisor-id)

Assignment (worker-id, building-id, start-date, num-days)

Building (building-id, address, building-type)

Notes: 1. skill-types are: Electric, Plumbing, Roofing, Framing etc.

2. building-types are: Office, Hospital, Residence, Warehouse etc.

3. supervisors are also workers of self-supervision.

Give SQL and RA expressions for the following queries: [In-course 1, 2008; 2004; Marks: 2.5 each.]

1. What are the skill types of workers assigned to building 'B02' (building-id)?
2. List the name of the workers assigned to 'warehouse' (building-type) buildings.
3. Find the no. of workers for each building where more than 5 workers are working for it.
4. Give 5% hourly wage increment for the workers working for 'hospital' buildings.

1. SQL: `select skill-type from worker natural join assignment where building-id = 'B02';`

RA: $\Pi_{\text{skill-type}}(\sigma_{\text{building-id} = \text{"B02"}}(\text{worker} \bowtie \text{assignment}))$

2. SQL: `select worker-name from worker natural join assignment natural join building where building-type = 'warehouse';`

RA: $\Pi_{\text{worker-name}}(\sigma_{\text{building-type} = \text{"B02"}}(\text{worker} \bowtie \text{assignment} \bowtie \text{building}))$

3. SQL: `select building-id, count(worker-id) from assignment group by building-id having count(worker-id) > 5;`

RA: $t_1 \leftarrow \text{building-id} \text{ } \rho_{\text{count}(\text{worker-id}) \text{ as no-of-workers}}(\text{assignment})$
 $\sigma_{\text{no-of-workers} > 5}(t_1)$

4. SQL: `update worker set hourly-rate = hourly-rate * 1.05 where worker-id in (select worker-id from assignment natural join building where building-type = 'hospital');`

RA: $t_1 \leftarrow \Pi_{\text{worker-id}}(\sigma_{\text{building-type} = \text{"hospital"}}(\text{assignment} \bowtie \text{building}))$

$t_2 \leftarrow \Pi_{\text{worker-id, worker-name, hourly-rate, skill-type, supervisor-id}}(\text{worker} \bowtie t_1)$

$t_3 \leftarrow \Pi_{\text{worker-id, worker-name, hourly-rate} * 1.05, \text{skill-type, supervisor-id}}(\text{worker} \bowtie t_1)$

$\text{worker} \leftarrow (\text{worker} - t_2) \cup t_3$

4.11 Consider the part of a bank database schema below:

Branch (branch-name, branch-city, assets)

Customer (customer-name, customer-street, customer-city)

Loan (loan-no, branch-name, amount)

Borrower (customer-name, loan-no)

Account (account-no, branch-name, balance)

Depositor (customer-name, account-no)

Give SQL and RA expressions for the following queries: [2003; Marks: 3 each.]

1. Find all customers who have either an account or a loan (but not both) at the bank.
2. Find the average account balance of those branches where the total account balance for individual branch is greater than 160,000.
3. Find the number of depositors for each branch.
4. Find the branch that has the highest average balance.

1. SQL: `select customer-name
from depositor natural full outer join borrower
where loan-number is null or account-number is null`

RA: $\Pi_{\text{customer-name}} (\sigma_{\text{loan-number} = \text{null} \vee \text{account-number} = \text{null}} (\text{depositor} \bowtie \text{borrower}))$

2. SQL: `select branch-name, avg(balance) from account
group by branch-name having sum(balance) > 16000;`

OR: `create view sum_bal as
(select branch-name from account group by branch-name
having sum(balance) > 16000);`

`create view average_bal as
(select branch-name, avg(balance) as avg_bal from account
group by branch-name);`

`select * from sum_bal natural join average_bal;`

RA: $t_1 \leftarrow \text{branch-name } \mathcal{G}_{\text{sum}(\text{balance})} \text{ as sum_bal } (\text{account})$
 $t_2 \leftarrow \sigma_{\text{sum_bal} > 160000} (t_1)$
 $t_3 \leftarrow \text{branch-name } \mathcal{G}_{\text{avg}(\text{balance})} \text{ as avg_bal } (t_2)$

3. SQL: `select count(distinct customer-name)
from depositor natural join account group by branch-name;`

RA: $\text{branch-name } \mathcal{G}_{\text{count}(\text{customer-name})} (\text{depositor} \bowtie \text{account})$

4. SQL: `select branch-name from account group by branch-name
having avg(balance) >= all
(select avg(balance) from account group by branch-name);`

OR: `create view avg_balance as
select branch-name, avg(balance) as avg_bal
from account group by branch-name;`

`create view max_balance as
select max(avg_bal) as max_bal from avg_bal;`

`select branch-name from avg_balance, max_balance where avg_bal = max_bal;`

RA: $t_1 \leftarrow \text{branch-name } \mathcal{G}_{\text{avg}(\text{balance})} \text{ as avg_bal } (\text{account})$
 $t_2 \leftarrow \mathcal{G}_{\text{max}(\text{avg_bal})} \text{ as max_bal } (t_1)$
 $\Pi_{\text{branch-name}} (\sigma_{\text{avg_bal} = \text{max_bal}} (t_1 \times t_2))$

4.12 Consider the part of a company database schema below:

employee (person-name, street, city)

works (person-name, company-name, salary)

company (company-name, city)

manages (person-name, manager-name)

Give SQL and RA expressions for the following queries: [In-course 1, 2005; Marks: 2.5 each.]

1. Find the names of all employees who live in the same city as the company for which they work.
2. Find all employees in the database who earn more than any employee of Beximco Textiles Limited.

3. Give all managers in the database a 10% salary raise, unless the salary would be greater than 100,000.
4. Find those companies whose employees earn a higher salary, on average, than the average salary of Beximco Textiles Limited.
5. Find the company with the smallest payroll.
6. Find the names, street addresses and cities of residence of all employees who work for Padma Textile Limited and earn more than 11,000.

1. *Similar to Complete Concepts Problem – Query No. 5*
2. *Similar to Complete Concepts Problem – Query No. 10*
3. *Almost Similar to Complete Concepts Problem – Query No. 14*

SQL: `update works set salary = salary * 1.1
where salary * 1.1 <= 100000 and person-name in (
select distinct manager-name from manages
) ;`

RA: $t_1 \leftarrow \Pi_{works.person-name, company-name, salary} (\sigma_{works.person-name = manager-name} (works \times manages))$
 $t_2 \leftarrow \Pi_{works.person-name, company-name, salary} (\sigma_{salary * 1.1 \leq 100000} (t_1))$
 $t_3 \leftarrow \Pi_{works.person-name, company-name, salary * 1.1} (t_2)$
 $works \leftarrow (works - t_2) \cup t_3$

4. *Similar to Complete Concepts Problem – Query No. 23*
5. *Similar to Complete Concepts Problem – Query No. 21*
6. *Similar to Complete Concepts Problem – Query No. 3*

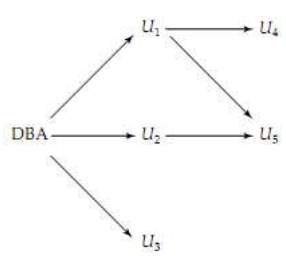
CHAPTER 6

INTEGRITY & SECURITY

Questions and Answers

6.1	<p>What integrity constraints are used in database? [2004. Marks: 1]</p> <p>The integrity constraints used in database are:</p> <ol style="list-style-type: none">1. Domain Constraints2. Referential Integrity Constraints
6.2	<p>Define foreign key and dangling tuples. How foreign key defines acceptability of dangling tuples? [2003. Marks: 4]</p> <p>Consider a pair of relations $r(R)$ and $s(S)$, and the natural join $r \bowtie s$. There may be a tuple t_r in r that does not join with any tuple in s. That is, there is no t_s in s such that $t_r[R \cap S] = t_s[R \cap S]$. Such tuples are called <i>dangling tuples</i>.</p> <p>Foreign key defines acceptability of dangling tuples by permitting the use of <i>null</i> values. Attributes of foreign keys are allowed to be null, provided that they have not otherwise been declared to be non-null. If all the columns of a foreign key are non-null in a given tuple, the usual definition of foreign-key constraints is used for that tuple. If any of the foreign-key columns is null, the tuple is defined automatically to satisfy the constraint.</p>
6.3	<p>SQL allows a foreign-key dependency to refer to the same relation, as in the following example:</p> <pre>create table manager (employee-name char(20), manager-name char(20), primary key employee-name, foreign key (manager-name) references manager on delete cascade)</pre> <p>Here, <i>employee-name</i> is a key to the table <i>manager</i>, meaning that each employee has at most one manager. The foreign-key clause requires that every manager also be an employee. Explain exactly what happens when a tuple in the relation <i>manager</i> is deleted.</p> <p>The tuples of all employees of the manager, at all levels, get deleted as well.</p> <p>This happens in a series of steps. The initial deletion will trigger deletion of all the tuples corresponding to direct employees of the manager. These deletions will in turn cause deletions of second level employee tuples, and so on, till all direct and indirect employee tuples are deleted.</p>
6.4	<p>What is a trigger and what are its parts? [2002. Marks: 3] OR, With example define trigger. [2006, Marks: 2. 2004, Marks: 1]</p> <p>A <i>trigger</i> is a statement that the system executes automatically as a side effect of a modification to the database.</p> <p>The parts of a trigger are:</p> <ol style="list-style-type: none">1. An event – which causes the trigger to be checked2. A condition – that must be satisfied for trigger execution to proceed.3. The actions – that are to be taken when the trigger executes

6.5	<p>Consider the following schemas:</p> <p><i>Account = (Account_number, Branch name, Balance)</i> <i>Depositor = (Customer_id, Account_number)</i></p> <p>Write an SQL trigger to carry out the following action: after update on account for each owner of the account, if the account balance is negative, delete the owner from the account and depositor relation. [2007. Marks: 2]</p> <p>create trigger <i>check-delete-trigger</i> after delete on <i>Account</i> referencing old row as <i>orow</i> for each row delete from <i>depositor</i> where <i>Depositor.Customer_id not in</i> <i>(select Customer_id from Depositor</i> <i>where Account_number <> orow.Account_number)</i> end</p>
6.6	<p>Define assertion with example. [2004. Marks: 1]</p> <p>An assertion is a predicate expressing a condition that we wish the database always to satisfy.</p> <p>For example, the following assertion ensures that the assets value for the Perryridge branch is equal to the sum of all the amounts lent by the Perryridge branch.</p> <p>create assertion <i>perry check</i> (not exists (select * from branch where branch-name = 'Perryridge' and <i>assets <> (select sum (amount) from loan where branch-name = 'Perryridge')))</i></p>
6.7	<p>Write an assertion for the bank database to ensure that the assets value for the Perryridge branch is equal to the sum of all the amounts lent by the Perryridge branch.</p> <p><i>See the example of assertion in Question and Answer 6.6.</i></p>
6.8	<p>What is database security? What security measures are taken to protect the database? [2003. Marks: 4]</p> <p><i>Database security</i> refers to protection from malicious access.</p> <p>To protect the database, the following levels of security measures are taken:</p> <ul style="list-style-type: none"> ➤ Database system. Some database-system users may be authorized to access only a limited portion of the database. Other users may be allowed to issue queries, but may be forbidden to modify the data. It is the responsibility of the database system to ensure that these authorization restrictions are not violated. ➤ Operating system. No matter how secure the database system is, weakness in operating system security may serve as a means of unauthorized access to the database. ➤ Network. Since almost all database systems allow remote access through terminals or networks, software-level security within the network software is as important as physical security, both on the Internet and in private networks. ➤ Physical. Sites with computer systems must be physically secured against armed or surreptitious entry by intruders. ➤ Human. Users must be authorized carefully to reduce the chance of any user giving access to an intruder in exchange for a bribe or other favors.
6.9	<p>What are the different forms of authorizations used in SQL? [2005. Marks: 2]</p> <p>The different forms of authorizations used in SQL are:</p> <ol style="list-style-type: none"> 1. Authorization on data (instance) – <i>read, insert, delete</i> and <i>update</i> authorizations 2. Authorization on database schema – <i>index, resource, alteration</i> and <i>drop</i> authorizations

6.10	<p>What is the use of role in database? [2004. Marks: 2]</p> <p>Authorizations can be granted to roles, in exactly the same fashion as they are granted to individual users. Each database user is granted a set of roles (which may be empty) that he or she is authorized to perform.</p>
6.11	<p>What is authorization graph? [2006. Marks: 1]</p> <p>The graph representing the passing of authorization from one user to another is called an <i>authorization graph</i>.</p> <div style="text-align: center;">  <pre> graph LR DBA --> U1 DBA --> U2 DBA --> U3 U1 --> U4 U1 --> U5 U2 --> U5 </pre> </div> <p style="text-align: center;">Figure: Authorization graph.</p>
6.12	<p>How can you relate views with authorization? [2004. Marks: 2]</p>
6.13	<p>What do you understand by authentication? [2004. Marks: 1]</p> <p><i>Authentication</i> refers to the task of verifying the identity of a person / software connecting to a database.</p>
6.14	<p>What are the different forms of authentication used in the database? [2006. Marks: 1]</p> <p>The different forms of authentication used in database are:</p> <ol style="list-style-type: none"> 1. Password-based authentication 2. Challenge-response system
6.15	<p>What are the properties of a good encryption technique? [2005. Marks: 1]</p> <p>The properties of a good encryption technique are:</p> <ol style="list-style-type: none"> 1. It is relatively simple for authorized users to encrypt and decrypt data. 2. It depends not on the secrecy of the algorithm, but rather on a parameter of the algorithm called the <i>encryption key</i>. 3. Its encryption key is extremely difficult for an intruder to determine.
6.16	<p>How public key encryption maintains security? [2005. Marks: 2]</p> <p>If user U_1 wants to store encrypted data, U_1 encrypts them using public key E_1 and decryption requires the private key D_1.</p> <p>If user U_1 wants to share data with U_2, U_1 encrypts the data using E_2, the public key of U_2. Since only user U_2 knows how to decrypt the data (using D_2), information is transferred securely.</p>
6.17	<p>Describe challenge-response system for authentication. [2004. Marks: 3]</p> <p>In the challenge-response system, the database system sends a challenge string to the user. The user encrypts the challenge string using a secret password as encryption key, and then returns the result. The database system can verify the authenticity of the user by decrypting the string with the same secret password, and checking the result with the original challenge string. This scheme ensures that no passwords travel across the network.</p> <p>Public-key systems can be used for encryption in challenge–response systems. The database system encrypts a challenge string using the user’s public key and sends it to the user. The user decrypts the string using her private key, and returns the result to the database system. The database system then checks the response. This scheme has the added benefit of not storing the secret password in the database, where it could potentially be seen by system administrators.</p>

6.18	What are the advantages of encrypting data stored in the database? <ol style="list-style-type: none"><li data-bbox="229 152 1511 226">1. Encrypted data allows authorized users to access data without worrying about other users or the system administrator gaining any information.<li data-bbox="229 241 1511 349">2. Encryption of data may simplify or even strengthen other authorization mechanisms. For example, distribution of the cryptographic key amongst only trusted users is both, a simple way to control read access, and an added layer of security above that offered by views.
6.19	Perhaps the most important data items in any database system are the passwords that control access to the database. Suggest a scheme for the secure storage of passwords. Be sure that your scheme allows the system to test passwords supplied by users who are attempting to log into the system. <p data-bbox="181 533 1511 640">A scheme for storing passwords would be to encrypt each password, and then use a hash index on the user-id. The user-id can be used to easily access the encrypted password. The password being used in a login attempt is then encrypted and compared with the stored encryption of the correct password.</p> <p data-bbox="181 656 1511 728">An advantage of this scheme is that passwords are not stored in clear text and the code for decryption need not even exist.</p>

CHAPTER 7

RELATIONAL DATABASE DESIGN

Concepts

7.1	<p>How to decompose a relation into BCNF with dependency preservation</p> <p>Let, R be the relation and FD be the set of functional dependencies.</p> <ol style="list-style-type: none"> 1. Check the first functional dependency, $\alpha \rightarrow \beta$. If α is <i>trivial</i> or a <i>superkey</i>, then R is in BCNF. So, go for the next functional dependency. 2. If α is not a superkey, then decompose R into $R_1 = (\alpha, \beta)$ and $R_2 = (R - \beta)$. 3. Repeat the steps 1 and 2 for each decomposed relation until it is found that <i>each functional dependency holds for at least one of the relations</i>. 4. If you find one or more functional dependencies that don't hold for any of the relations, then start over again by reordering the elements of FD.
------------	---

Questions and Answers

7.1	<p>Define functional dependency.</p> <p>Consider a relation schema R, α and β are set of attributes of R and let $\alpha \subseteq R$ and $\beta \subseteq R$. The functional dependency $\alpha \rightarrow \beta$ holds on schema R if, in any legal relation $r(R)$, for all pairs of tuples t_1 and t_2 in r such that $t_1[\alpha] = t_2[\alpha]$, it is also the case that $t_1[\beta] = t_2[\beta]$.</p>																								
7.2	<p>Define superkey using functional dependency. [2004. Marks: 2]</p> <p>A set of attributes K is a superkey of a relation schema R if $K \rightarrow R$. That is K is a superkey if, whenever $t_1[K] = t_2[K]$, it is also the case that $t_1[R] = t_2[R]$ (i.e., $t_1 = t_2$).</p>																								
7.3	<p>Define trivial functional dependency. [In-course 2, 2007; In-course 2, 2008. Marks: 1]</p> <p>A functional dependency of the form $\alpha \rightarrow \beta$ is trivial if $\beta \subseteq \alpha$. For example, $A \rightarrow A$, $AB \rightarrow A$ etc.</p>																								
7.4	<p>Why certain functional dependencies are called trivial functional dependencies?</p> <p>Certain functional dependencies are called trivial functional dependencies because they are satisfied by all relations.</p>																								
7.5	<p>List all nontrivial functional dependencies (with no common attributes) satisfied by the following relation: [In-course 2, 2007; In-course 2, 2008; 2005; Marks: 2]</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr><th>A</th><th>B</th><th>C</th></tr> </thead> <tbody> <tr><td>a1</td><td>b1</td><td>c1</td></tr> <tr><td>a1</td><td>b1</td><td>c2</td></tr> <tr><td>a2</td><td>b1</td><td>c1</td></tr> <tr><td>a2</td><td>b1</td><td>c3</td></tr> </tbody> </table> <p style="text-align: center;">$A \rightarrow B$ $C \rightarrow B$ $AC \rightarrow B$</p>	A	B	C	a1	b1	c1	a1	b1	c2	a2	b1	c1	a2	b1	c3									
A	B	C																							
a1	b1	c1																							
a1	b1	c2																							
a2	b1	c1																							
a2	b1	c3																							
7.6	<p>List all nontrivial functional dependencies satisfied by the following relation: [2003. Marks: 3]</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr><th>A</th><th>B</th><th>C</th><th>D</th></tr> </thead> <tbody> <tr><td>a1</td><td>b1</td><td>c1</td><td>d1</td></tr> <tr><td>a1</td><td>b2</td><td>c1</td><td>d2</td></tr> <tr><td>a2</td><td>b2</td><td>c2</td><td>d2</td></tr> <tr><td>a2</td><td>b2</td><td>c2</td><td>d3</td></tr> <tr><td>a3</td><td>b3</td><td>c2</td><td>d4</td></tr> </tbody> </table> <p style="text-align: center;">$A \rightarrow C$ $AB \rightarrow C$ $AD \rightarrow C$ $CD \rightarrow A$ $ABD \rightarrow C$ $BCD \rightarrow A$ $D \rightarrow B$ $AD \rightarrow B$ $BC \rightarrow A$ $CD \rightarrow B$ $ACD \rightarrow B$</p>	A	B	C	D	a1	b1	c1	d1	a1	b2	c1	d2	a2	b2	c2	d2	a2	b2	c2	d3	a3	b3	c2	d4
A	B	C	D																						
a1	b1	c1	d1																						
a1	b2	c1	d2																						
a2	b2	c2	d2																						
a2	b2	c2	d3																						
a3	b3	c2	d4																						

7.7	<p>Show functional dependencies to indicate the following: [2004, 2007. Marks: 2]</p> <p>i) A one-to-one relationship set exists between entity sets <i>account</i> and <i>customer</i>. ii) A many-to-one relationship set exists between entity sets <i>account</i> and <i>customer</i>.</p> <p>Where:</p> <p><i>customer</i> (<u>customer-name</u>, customer-street, customer-city) <i>account</i> (<u>account-number</u>, balance)</p> <p>Let Pk(r) denote the primary key attribute of a relation r.</p> <ol style="list-style-type: none"> The functional dependencies Pk(account) → Pk(customer) and Pk(customer) → Pk(account) indicate a one-to-one relationship, because any two tuples with the same value for account must have the same value for customer, and any two tuples agreeing on customer must have the same value for account. The functional dependency Pk(account) → Pk(customer) indicates a many-to-one relationship since any account value which is repeated will have the same customer value, but many account values may have the same customer value.
7.8	<p>Why Armstrong's axioms are called 'sound' and 'complete'? [2005. Marks: 1]</p> <p>Armstrong's axioms are <i>sound</i>, because they do not generate any <i>incorrect</i> functional dependencies.</p> <p>They are <i>complete</i>, because, for a given set F of functional dependencies, they allow us to generate all F^+.</p>
7.9	<p>Use the definition of functional dependency to argue that each of Armstrong's axioms (reflexivity, augmentation, and transitivity) is sound.</p> <p>The definition of functional dependency is: $\alpha \rightarrow \beta$ holds on R if in any legal relation $r(R)$, for all pairs of tuples t_1 and t_2 in r such that $t_1[\alpha] = t_2[\alpha]$, it is also the case that $t_1[\beta] = t_2[\beta]$.</p> <p>Reflexivity rule: If α is a set of attributes, and $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$.</p> <p>Assume $\exists t_1$ and t_2 such that $t_1[\alpha] = t_2[\alpha]$</p> <p>$\therefore t_1[\beta] = t_2[\beta]$ [since $\beta \subseteq \alpha$] $\therefore \alpha \rightarrow \beta$ [Definition of FD]</p> <p>Augmentation rule: If $\alpha \rightarrow \beta$, and γ is a set of attributes, then $\gamma\alpha \rightarrow \gamma\beta$.</p> <p>Assume $\exists t_1, t_2$ such that $t_1[\gamma\alpha] = t_2[\gamma\alpha]$</p> <p>$t_1[\gamma] = t_2[\gamma]$ [$\gamma \subseteq \gamma\alpha$] $t_1[\alpha] = t_2[\alpha]$ [$\alpha \subseteq \gamma\alpha$] $t_1[\beta] = t_2[\beta]$ [Definition of $\alpha \rightarrow \beta$] $t_1[\gamma\beta] = t_2[\gamma\beta]$ [$\gamma\beta = \gamma \cup \beta$] $\therefore \gamma\alpha \rightarrow \gamma\beta$ [Definition of FD]</p> <p>Transitivity rule: If $\alpha \rightarrow \beta$ and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$.</p> <p>Assume $\exists t_1, t_2$ such that $t_1[\alpha] = t_2[\alpha]$</p> <p>$t_1[\beta] = t_2[\beta]$ [Definition of $\alpha \rightarrow \beta$] $t_1[\gamma] = t_2[\gamma]$ [Definition of $\beta \rightarrow \gamma$] $\therefore \alpha \rightarrow \gamma$ [Definition of FD]</p>
7.10	<p>Use Armstrong's axioms to prove the soundness of Union, Decomposition and Pseudotransitivity rules. [In-course 2, 2007; In-course 2, 2008; 2007; Marks: 3. 2003, Marks: 5]</p> <p>Union rule: If $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$ then $\alpha \rightarrow \beta\gamma$.</p> <p>We derive:</p>

	$\alpha \rightarrow \beta$ [Given] $\alpha\alpha \rightarrow \alpha\beta$ [Augmentation rule] $\alpha \rightarrow \alpha\beta \dots(i)$ [Union of identical sets] Again, $\alpha \rightarrow \gamma$ [Given] $\alpha\beta \rightarrow \gamma\beta \dots(ii)$ [Augmentation rule] $\therefore \alpha \rightarrow \beta\gamma$ [From (i) and (ii) using transitivity rule and set union commutativity] Decomposition rule: If $\alpha \rightarrow \beta\gamma$, then $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$. We derive: $\alpha \rightarrow \beta\gamma$ [Given] $\beta\gamma \rightarrow \beta$ [Reflexivity rule] $\therefore \alpha \rightarrow \beta$ [Transitivity rule] Again, $\beta\gamma \rightarrow \gamma$ [Reflexive rule] $\therefore \alpha \rightarrow \gamma$ [Transitive rule] Pseudotransitivity rule: If $\alpha \rightarrow \beta$ and $\gamma\beta \rightarrow \delta$, then $\alpha\gamma \rightarrow \delta$. We derive: $\alpha \rightarrow \beta$ [Given] $\alpha\gamma \rightarrow \gamma\beta$ [Augmentation rule and set union commutativity] $\gamma\beta \rightarrow \delta$ [Given] $\therefore \alpha\gamma \rightarrow \delta$ [Transitivity rule]									
7.11	<p>Consider the following proposed rule for functional dependencies: If $\alpha \rightarrow \beta$ and $\delta \rightarrow \beta$, then $\alpha \rightarrow \delta$. Prove that this rule is not sound by showing a relation r that satisfies $\alpha \rightarrow \beta$ and $\delta \rightarrow \beta$, does not satisfy $\alpha \rightarrow \delta$. [2004. Marks: 2]</p> <p>Consider the following relation r :</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>A</th> <th>B</th> <th>C</th> </tr> </thead> <tbody> <tr> <td>a1</td> <td>b1</td> <td>c1</td> </tr> <tr> <td>a1</td> <td>b1</td> <td>c2</td> </tr> </tbody> </table> <p>Let, $\alpha = A, \beta = B, \gamma = C$.</p> <p>From the above relation, we see that $A \rightarrow B$ and $C \rightarrow B$ (i.e., $\alpha \rightarrow \beta$ and $\gamma \rightarrow \beta$). However, it is not the case that $A \rightarrow C$ (i.e., $\alpha \rightarrow \gamma$) since the same A (α) value is in two tuples, but the C (γ) value in those tuples disagree.</p>	A	B	C	a1	b1	c1	a1	b1	c2
A	B	C								
a1	b1	c1								
a1	b1	c2								
7.12	<p>Define closure of attribute sets, α^+.</p> <p>Let α be a set of attributes. We call the set of all attributes functionally determined by α under a set F of functional dependencies the closure of α under F, and we denote it by α^+.</p>									
7.13	<p>What are uses of ‘closure of attribute sets’, α^+? [In-course 2, 2007; In-course 2, 2008; 2007; 2004; Marks: 2]</p> <ol style="list-style-type: none"> 1. To test if α is a superkey, we compute α^+ and check if α^+ contains all attributes of R. If it contains, α is a superkey of R. 2. For a given set of F, we can check if a functional dependency $\alpha \rightarrow \beta$ holds (or is in F^+), by checking if $\beta \subseteq \alpha^+$. That is, we compute α^+ by using attribute closure and then check if it contains β. 3. It gives us an alternate way to compute F^+: For each $\gamma \subseteq R$, we find the closure γ^+, and for each $S \subseteq \gamma^+$, we output a functional dependency $\gamma \rightarrow S$. 									
7.14	<p>Compute the closure of the attribute/s to list the candidate key/s for relation schema $R = (A, B, C, D, E)$ with functional dependencies $F = \{A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A\}$ [2005. Marks: 2]</p>									

7.15	<p>Define canonical cover, F_C. [2004. Marks: 2]</p> <p>A canonical cover F_C for F is a set of dependencies such that F logically implies all dependencies in F_C and F_C logically implies all dependencies in F. Furthermore, F_C must have the following properties:</p> <ol style="list-style-type: none"> 1. No functional dependency in F_C contains an extraneous attribute. 2. Each left side of a functional dependency in F_C is unique. That is, there are no two dependencies $\alpha_1 \rightarrow \beta_1$ and $\alpha_2 \rightarrow \beta_2$ in F_C such that $\alpha_1 = \alpha_2$.
7.16	<p>What is the advantage of using canonical cover, F_C? [In-course 2, 2007, Marks: 1]</p> <p>The advantage of using canonical cover is that the effort spent in checking for dependency violations can be minimized.</p>
7.17	<p>What are the design goals for relational database design? [In-course 2, 2007; 2005 Marks: 1]</p> <p>Explain why each is desirable.</p> <p>The design goals for relational database design are:</p> <ol style="list-style-type: none"> 1. Lossless-join decompositions 2. Dependency preserving decompositions 3. Minimization of repetition of information <p>They are desirable so we can maintain an accurate database, check correctness of updates quickly, and use the smallest amount of space possible.</p>
7.18	<p>Explain what is meant by <i>repetition of information</i>, <i>inability to represent information</i> and <i>loss of information</i>. Explain why each of these properties may indicate a bad relational database design. [2006. Marks: 4.5]</p> <p><i>Repetition of information</i> is a condition in a relational database where the values of one attribute are determined by the values of another attribute in the same relation, and both values are repeated throughout the relation. This is a bad relational database design because it increases the storage required for the relation and it makes updating the relation more difficult.</p> <p><i>Inability to represent information</i> is a condition where a relationship exists among only a proper subset of the attributes in a relation. This is bad relational database design because all the unrelated attributes must be filled with null values otherwise a tuple without the unrelated information cannot be inserted into the relation.</p> <p><i>Loss of information</i> is a condition of a relational database which results from the decomposition of one relation into two relations and which cannot be combined to recreate the original relation. It is a bad relational database design because certain queries cannot be answered using the reconstructed relation that could have been answered using the original relation.</p>
7.19	<p>Explain the condition for lossless-join decomposition. [2004. Marks: 2]</p> <p>Let R be a relation schema, F be a set of functional dependencies on R; and R_1 and R_2 form a decomposition of R.</p> <p>The decomposition is a lossless-join decomposition of R if at least one of the following functional dependencies are in F^+ :</p> <ol style="list-style-type: none"> 1. $R_1 \cap R_2 \rightarrow R_1$ 2. $R_1 \cap R_2 \rightarrow R_2$ <p>In other words, if $R_1 \cap R_2$ (the attribute involved in the natural join) forms a superkey of either R_1 or R_2, the decomposition of R is lossless-join decomposition.</p>
7.20	<p>Suppose that we decompose the schema $R = (A, B, C, D, E)$ into (A, B, C) and (A, D, E). Show that this decomposition is lossless-join decomposition if the following set F of functional dependencies holds: [2006. Marks: 2]</p> <p>$A \rightarrow BC$ $CD \rightarrow E$</p>

B → D
E → A

A decomposition $\{R_1, R_2\}$ is a lossless-join decomposition if

$R_1 \cap R_2 \rightarrow R_1$
 or $R_1 \cap R_2 \rightarrow R_2$.

Let $R_1 = (A, B, C)$, $R_2 = (A, D, E)$, and $R_1 \cap R_2 = A$.

Since A is a candidate key (because, the closure of A is R),

$\therefore R_1 \cap R_2 \rightarrow R_1$.

7.21 Suppose that we decompose the schema $R = (A, B, C, D, E)$ into (A, B, C) and (C, D, E) . Show that this decomposition is *not* lossless-join decomposition if the following set F of functional dependencies holds: [2006. Marks: 2]

A → BC
CD → E
B → D
E → A

Let, r be a relation as follows:

A	B	C	D	E
a1	b1	c1	d1	e1
a2	b2	c1	d2	e2

With $R_1 = (A, B, C)$ and $R_2 = (C, D, E)$:

$\Pi_{R_1}(r)$ would be:

A	B	C
a1	b1	c1
a2	b2	c1

$\Pi_{R_2}(r)$ would be:

C	D	E
c1	d1	e1
c1	d2	e2

$\Pi_{R_1}(r) \bowtie \Pi_{R_2}(r)$ would be:

A	B	C	D	E
a1	b1	c1	d1	e1
a1	b1	c1	d2	e2
a2	b2	c1	d1	e1
a2	b2	c1	d2	e2

Clearly, $\Pi_{R_1}(r) \bowtie \Pi_{R_2}(r) \neq r$.

Therefore, this is a lossy join.

7.22 Deduce the condition for dependency preservation using restrictions for decomposing a given schema R and a set of FDs F. Decompose the schema $R = (A, B, C, D, E)$ with functional dependencies $F = \{ A \rightarrow B, BC \rightarrow D \}$ into BCNF with dependency preservation. [2005. Marks: 2 + 1]

Let F be a set of functional dependencies on schema R. Let R_1, R_2, \dots, R_n be a decomposition of R.

The *restriction* of F to R_i is the set of all functional dependencies in F^+ that include only attributes of R_i .

The set of restrictions F_1, F_2, \dots, F_n is the set of dependencies that can be checked efficiently.

Let $F' = F_1 \cup F_2 \cup \dots \cup F_n$.

	<p>F' is a set of functional dependencies on schema R, but in general, $F' \neq F$</p> <p>However, it may be that $F'^+ = F^+$.</p> <p>If this is so, then every functional dependency in F is implied by F', and if F' is satisfied, then F must also be satisfied.</p> <p>Therefore, the condition for dependency preservation using restrictions for decomposing a given schema R and a set of FDs F is that $F'^+ = F^+$.</p> <p>Decomposition of the schema R:</p> <p>We change the order of the FDs in F such that $F = \{BC \rightarrow D, A \rightarrow B\}$.</p> <p>Now, the FD $BC \rightarrow D$ holds on R, but BC is not a superkey. So, we decompose R into $R_1 = (B, C, D)$ and $R_2 = (A, B, C, E)$</p> <p>R_1 is in BCNF. However, the FD $A \rightarrow B$ holds on R_2, but A is not a superkey. So, we decompose R_2 into $R_3 = (A, B)$ and $R_4 = (A, C, E)$</p> <p>Now, R_3 and R_4 both are in BCNF. [R_4 is in BCNF as only trivial functional dependencies exist in R_4]</p> <p>So, the final decomposed relations are: $R_1 = (B, C, D)$, $R_3 = (A, B)$ and $R_4 = (A, C, E)$.</p>
7.23	<p>Consider a relation schema $R = (A, B, C, D)$ and with functional dependencies $F = \{A \rightarrow BC, B \rightarrow D, D \rightarrow B\}$. Show the BCNF decomposition of the above schema with dependency preservation with causes. [In-course 2, 2008; Marks: 2]</p> <p>The FD $A \rightarrow BC$ holds on R, and A is a superkey ($\because A \rightarrow BC$ and $B \rightarrow D, \therefore A \rightarrow BCD$, or, $A \rightarrow ABCD$).</p> <p>Therefore, we go for the next FD, $B \rightarrow D$. This holds on R, but B is not a superkey. So, we decompose R into $R_1 = (B, D)$ and $R_2 = (A, B, C)$</p> <p>Now, both R_1 and R_2 are in BCNF as R_1 satisfies $B \rightarrow D$ and R_2 satisfies $A \rightarrow BC$.</p> <p>So, the final decomposition of R is: $R_1 = (B, D)$ and $R_2 = (A, B, C)$.</p>
7.24	<p>What are the differences between BCNF and 3NF? [In-course 2, 2008; 2002, 2004, 2007. Marks: 3]</p> <p>For a functional dependency $\alpha \rightarrow \beta$, 3NF allows this dependency in a relation if each attribute A in $\beta - \alpha$ is contained in any candidate key for R. However, BCNF does not allow this condition.</p> <p>It is always possible to find a dependency-preserving lossless-join decomposition that is in 3NF. However, it is not always possible to find such decomposition that is in BCNF.</p> <p>Repetition of information occurs in 3NF, whereas no repetition of information occurs in BCNF.</p>
7.25	<p>In designing a relational database, why might we choose a non-BCNF design?</p> <p>BCNF is not always dependency preserving. Therefore, we may want to choose another normal form (specifically, 3NF) in order to make checking dependencies easier during updates. This would avoid joins to check dependencies and increase system performance.</p>
7.26	<p>What is multivalued dependency? Give an example. [2002. Marks: 4]</p> <p>Let R be a relation schema and let $\alpha \subseteq R$ and $\beta \subseteq R$. The multivalued dependency $\alpha \twoheadrightarrow \beta$ holds on R if in any legal relation $r(R)$, for all pairs of tuples t_1 and t_2 in r such that $t_1[\alpha] = t_2[\alpha]$, there exist tuples t_3 and t_4 in r such that:</p> <p>$t_1[\alpha] = t_2[\alpha] = t_3[\alpha] = t_4[\alpha]$ $t_3[\beta] = t_1[\beta]$ $t_4[\beta] = t_2[\beta]$</p>

$$t_3[R - \beta] = t_2[R - \beta]$$

$$t_4[R - \beta] = t_1[R - \beta]$$

Example:

Name	Address	Car
Tom	North Road	Toyota
Tom	Oak Street	Honda
Tom	North Road	Honda
Tom	Oak Street	Toyota

In the above relation, **Name** \twoheadrightarrow **Address** and **Name** \twoheadrightarrow **Car**.

7.27 Give an example of a relation schema R and a set of dependencies that R is in BCNF, but not in 4NF. [2006. Marks: 2]

$R = (\text{loan-no}, \text{amount}, \text{customer-name}, \text{customer-address})$

$F = \{\text{loan-no} \rightarrow \text{amount}\}$

If we assume that each customer might have more than one addresses, then the functional dependency $\text{customer-name} \rightarrow \text{customer-address}$ cannot be enforced. Thus, R is in BCNF. However, it is not in 4NF as it contains multivalued values for customer-address and therefore there occurs repetition of information in the loan-no and amount fields.

7.28 An employee database is to hold information about employees, the department they are in and the skills which they hold. The attributes to be stored are:

$(\text{emp-id}, \text{emp-name}, \text{emp-phone}, \text{dept-name}, \text{dept-phone}, \text{dept-mgrid}, \text{skill-id}, \text{skill-name}, \text{skill-date}, \text{skill-level})$

An employee may have many skills such as word-processing, typing, librarian... The date on which the skill was last tested and the level displayed at that test are recorded for the purposes of assigning work and determining salary. An employee is attached to one department and each department has a unique manager.

- i. Derive a functional dependency set for the above database, stating clearly any assumptions that you make.
- ii. Derive a set of BCNF relations, indicating the primary key of each relation.

[2002. Marks: 4 + 4]

CHAPTER 11

STORAGE & FILE STRUCTURE

Theories

11.1	<p>Define the term: RAID [In-course 2, 2008; 2003, 2005. Marks: 1]</p> <p>RAID (<i>Redundant Array of Independent Disks</i>) is a technology which makes use of two or more hard drives in order to improve performance, reliability or create larger data volumes.</p>
11.2	<p>What is mirroring / shadowing? [2004, 2005. Marks: 1]</p> <p>The simplest but most expensive approach to introduce redundancy is to duplicate every disk. This technique is called <i>mirroring</i> or <i>shadowing</i>.</p>
11.3	<p>What is striping?</p> <p>In context of RAID, striping means splitting and writing data across multiple drives to increase throughput.</p> <p>There are two types of striping:</p> <ol style="list-style-type: none"> 1. Bit-level Striping: consists of splitting the bits of each <i>byte</i> across multiple disks. 2. Block-level Striping: stripes <i>blocks</i> across multiple disks.
11.4	<p>How RAID improves reliability via redundancy? [2006. Marks: 2]</p> <ol style="list-style-type: none"> 1. The chance that at least one disk out of a set of N disks will fail is much higher than the chance that a specific disk will fail. 2. Redundancy stores extra information that is not needed normally, but that can be used in the event of failure of a disk to rebuild the lost information.
11.5	<p>How RAID improves performance via parallelism? [In-course 2, 2008; Marks: 2]</p> <ol style="list-style-type: none"> 1. With disk mirroring, the rate at which read requests can be handled is doubled, since read requests can be sent to either disk. 2. The transfer rate of each read is the same as in a single disk system, but the number of reads per unit time is doubled.
11.6	<p>What are the advantages of having large number of disks in a system? [2004. Marks: 2]</p> <ol style="list-style-type: none"> 1. Improving the rate at which data can be read or written, if the disks are operated in parallel. Several independent reads or writes can also be performed in parallel. 2. Improving the reliability of data storage – because redundant information can be stored on multiple disks. Thus failure of one disk does not lead to loss of information.
11.7	<p>What are the factors for choosing a RAID level? [2003. Marks: 2]</p> <ol style="list-style-type: none"> 1. Monetary cost of extra disk storage requirements. 2. Performance requirements in terms of number of I/O operations. 3. Performance when a disk has failed. 4. Performance during rebuild (while the data in a failed disk is being rebuilt on a new disk). 5. How many disks should be in an array? 6. How many bits should be protected by each parity bit?
11.8	<p>Which RAID level is used for storage of log files in a database? Justify your answer. [2007. Marks: 2]</p> <p>RAID 1 is used for storage of log files in a database, because for storing log files, fault tolerance is needed on a limited volume of data (the limit is the capacity of 1 disk).</p>

11.9 What are possible ways of organizing the records in files? What does reorganization do? [2004. Marks: 4 + 1]

OR, Classify file organization. Why reorganization is required in sequential file organization? [2003. Marks: 2 + 1]

OR, What are different types of organization of records in files? What do you understand by reorganization? [2006. Marks: 3 + 1]

File organization:

- 1. Heap file organization:** Any record can be placed anywhere in the file where there is space for the record. There is no ordering of records. Typically, there is a single file for each relation.
- 2. Sequential file organization:** Records are stored in sequential order, based on the value of a “search key” of each record.
- 3. Hashing file organization:** A hash function is computed on some attribute of each record. The result of the function specifies in which block of the file the record should be placed.
- 4. Multitable Clustering file organization:** Records of several different relations can be stored in the same file. Related records of the different relations are stored on the same block so that one I/O operation fetches related records from all the relations.

Reorganization:

The sequential file organization will work well if relatively few records need to be stored in overflow blocks. Eventually, however, the correspondence between search-key order and physical order may be totally lost. In such cases sequential processing will become much less efficient. At this point, the file should be *reorganized* so that it is once again physically in sequential order.

11.10 Describe slotted page structure for organizing records within a single block. [In-course 2, 2008; 2004. Marks: 4]

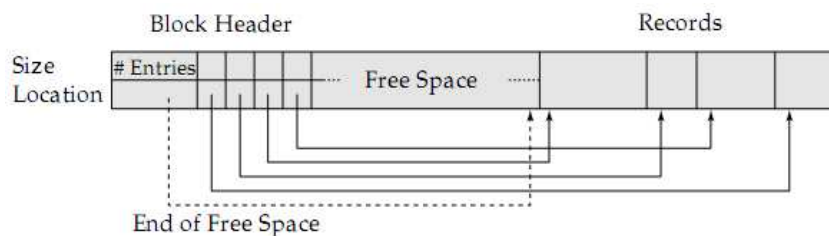


Figure 11.10: Slotted page structure.

In slotted page structure, there is a header at the beginning of each block, containing:

1. No of record entries in the header
2. The end of free space in the block
3. An array whose entries contain the location and size of each record.

The actual records are allocated contiguously in the block, starting from the end of the block. The free space in the block is contiguous, between the final entry in the header file and the first record.

If a record is inserted, space is allocated for it at the end of the free space and an entry containing its size and location is added to the header.

If a record is deleted, the space it occupies is freed and its entry is set to deleted. Further, the records in the block before the deleted record are moved, so that the free space created by deletion gets occupied and all free space is again between the final entry in the header array and the first record. The end-of-free-space pointer in the header is appropriately updated as well.

Records can be grown and shrunk by similar techniques, as long as there is space in the block. The cost of moving the record is not so high, since the size of a block is limited: A typical value is 4KB.

The slotted page structure requires that there be no pointers that point directly to records. Instead, pointers must point to the entry in the header that contains the actual location of the record. This level of indirection allows records to be moved to prevent fragmentation of space inside a block, while supporting indirect pointers to the record.

11.11 **How variable-length records are represented by fixed-length records? [2005. Marks: 2]**
Not present at sir's lecture. Therefore, it is assumed to be not important for exam. However, the answer is in the book – topic no. 11.6.2.2, page no. 420 (according to 4th edition).

11.12 **Consider a relational database with two relations:**
course (course-name, room, instructor)
enrollment (course-name, student-name, grade)
Define instances of these relations for two courses, each of which enrolls three students. Give the file structure of these relations that uses clustering. [2004. Marks: 2]

Instances of the given relations:

course-name	room	instructor	
Java	420	SMH	c_1
OS	320	MHK	c_2

course relation

course-name	student-name	grade	
Java	X	A	e_1
Java	Y	B	e_2
Java	Z	C	e_3
OS	X	A	e_4
OS	Y	B	e_5
OS	Z	C	e_6

enrollment relation

Clustering file structure of these relations:

- Block 0 contains: c_1, e_1, e_2 and e_3
- Block 1 contains: c_2, e_4, e_5 and e_6

11.13 **Give one advantage and disadvantage of the following strategies for storing a relational database: [2007. Marks: 1 + 1]**

- a. Store each relation in one file**
- b. Store multiple relations / the entire database in one file**

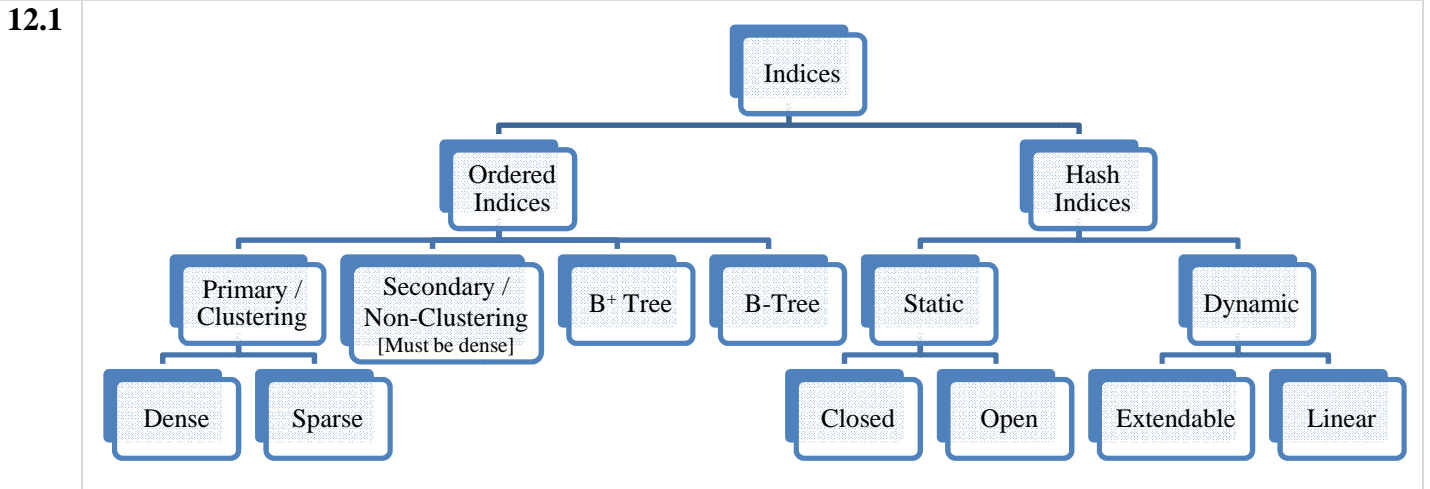
- a. Advantages of storing a relation as a file include using the file system provided by the OS, thus simplifying the DBMS, but incur the disadvantage of restricting the ability of the DBMS to increase performance by using more sophisticated storage structures.
- b. By using one file for multiple relations, these complex structures can be implemented through the DBMS, but this increases the size and complexity of the DBMS.

11.14 **Define seek time. [In-course 2, 2008. Marks: 1]**
 The time it takes for a disk read/write head to move to a specific data track is called *seek time*.

CHAPTER 12

INDEXING AND HASHING

Concepts



12.2

Search Key
An attribute or set of attributes used to look up records in a file is called a *search key*.

Primary / Clustering Index
If the file containing the records is sequentially ordered, a *primary index* is an index whose search key also defines the sequential order of the file.

Secondary / Non-Clustering Index
Indices whose search key specifies an order different from the sequential order of the file are called *secondary indices*.

Index-Sequential Files
Files that are ordered sequentially on some search key and have a primary index on that search key are called *index-sequential files*.

Dense Index
Dense index is the index where an index record appears for every search-key value in the file.

Sparse Index
Sparse index is the index where an index record appears for only some of the search-key values in the file.

Multilevel Index
An index with two or more levels is called a *multilevel index*.

B⁺ Tree
A *B⁺ tree* is a type of index which takes the form of a balanced tree in which every path from the root of the tree to a leaf of the tree is of the same length.
In a B⁺ tree, each non-leaf node in the tree has between $\lceil n / 2 \rceil$ and n children, where n is fixed for a particular tree. Each leaf has between $\lceil (n - 1) / 2 \rceil$ and $n - 1$ values. The ranges of values in each leaf do not overlap.

B-Tree
A *B-tree* index is similar to B⁺ tree index except that search-key values in a B-tree appear only once.

Hashing
Hashing is a process of generating an index or address basing on some data. A hash function is used to compute the location of a record from a given search-key value.

Hash Index

A *hash index* is an index which organizes the search keys, with their associated pointers, into a hash file structure.

Hash Function

A *hash function* is any well-defined procedure or mathematical function which converts a large, possibly variable-sized amount of data into a small datum, usually a single integer that may serve as an index into an array.

The values returned by a hash function are called *hash values*, *hash codes*, *hash sums*, or simply *hashes*.

12.3

Advantages of Dense Index

It is generally faster to locate a record if we have a dense index rather than a sparse index.

Advantages of Sparse Index

However, sparse indices have advantages over dense indices in that they require less space and they impose less maintenance overhead for insertions and deletions.

Advantages of Multilevel Index

Searching for records with a multilevel index requires significantly fewer I/O operations than does searching for records by binary search.

Advantages of Primary Index

A sequential scan in primary index order is efficient because records in the file are stored physically in the same order as the index order.

Advantages of Secondary Index

Secondary indices improve the performance of queries that use keys other than the search key of the primary index.

Disadvantages of Secondary Index

Secondary indices impose a significant overhead on modification of the database.

Disadvantages of Index-Sequential File Organization

The main disadvantage of the index-sequential file organization is that performance degrades as the file grows, both for index lookups and for sequential scans through the data. Although this degradation can be remedied by reorganization of the file, frequent reorganizations are undesirable.

Disadvantages of B⁺ Tree

1. Imposes performance overhead on insertion and deletion.
2. Adds space overhead – Since nodes may be as much as half empty (if they have the minimum number of children), there is some wasted space.

Advantages of B-Tree

1. May use less tree nodes than a corresponding B⁺ tree.
2. Sometimes it is possible to find the desired value before reaching a leaf node.

Disdvantages of B-Tree

1. Only a small fraction of desired values are found before reaching a leaf node.
2. Fewer search-keys appear in non-leaf nodes; hence, fan-out is reduced. Thus, B-trees typically have greater depth than a corresponding B⁺ tree.
3. Insertion and deletion are more complicated than in B⁺ trees.
4. Implementation is harder than B⁺ trees, since leaf and non-leaf nodes are of different sizes.

Advantages of Hashing

1. Allows to avoid accessing an index structure.
2. Provides a way of constructing indices.

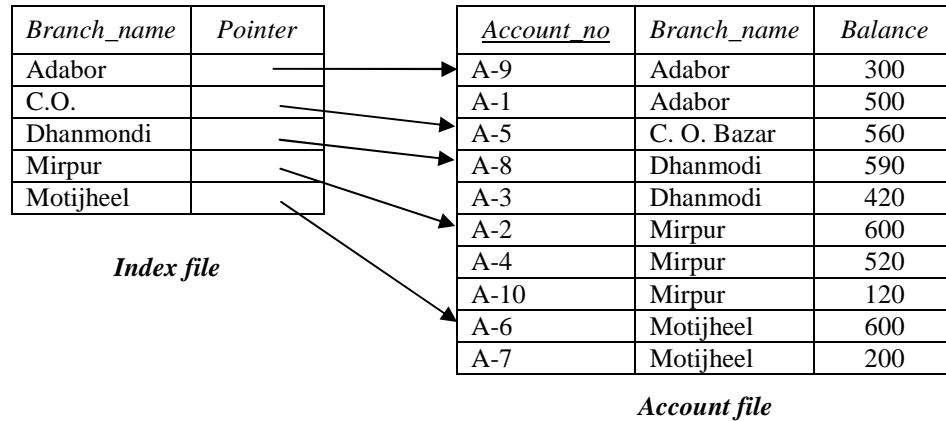
Questions and Answers

12.1	<p>Define the followings:</p> <ul style="list-style-type: none"> i. Primary Index [2006, Marks: 0.5. 2004, Marks: 1.5] ii. Secondary Index [2006. Marks: 0.5] iii. Sparse Index [2006. Marks: 0.5] iv. Dense Index [2006. Marks: 0.5] v. B⁺ tree [2004. Marks: 1.5] vi. Hashing [2005, Marks: 1. 2004, Marks: 1] <p>See Concept 12.2.</p>										
12.2	<p>Since indices speed query processing, why might they not be kept on several search keys? Reasons for not keeping several search indices include:</p> <ol style="list-style-type: none"> Every index requires additional CPU time and disk I/O overhead during inserts and deletions. Indices on non-primary keys might have to be changed on updates, although an index on the primary key might not (this is because updates typically do not modify the primary key attributes). Each extra index requires additional storage space. For queries which involve conditions on several search keys, efficiency might not be bad even if only some of the keys have indices on them. Therefore database performance is improved less by adding indices when many indices already exist. 										
12.3	<p>What are the differences between a <i>primary index</i> and a <i>secondary index</i>? [2005, Marks: 2. 2003, Marks: 3]</p> <table border="1" data-bbox="196 1003 1505 1435"> <thead> <tr> <th data-bbox="196 1003 850 1059">Primary Index</th> <th data-bbox="850 1003 1505 1059">Secondary Index</th> </tr> </thead> <tbody> <tr> <td data-bbox="196 1059 850 1216">1. If the file containing the records is sequentially ordered, a <i>primary index</i> is an index whose search key also defines the sequential order of the file.</td> <td data-bbox="850 1059 1505 1216">1. Indices whose search key specifies an order different from the sequential order of the file are called <i>secondary indices</i>.</td> </tr> <tr> <td data-bbox="196 1216 850 1301">2. There can be only one primary index for a relation.</td> <td data-bbox="850 1216 1505 1301">2. There can be many secondary indices for a relation.</td> </tr> <tr> <td data-bbox="196 1301 850 1386">3. A sequential scan in primary index order is efficient.</td> <td data-bbox="850 1301 1505 1386">3. Performance of sequential scan in secondary index order is poor.</td> </tr> <tr> <td data-bbox="196 1386 850 1435">4. Can be dense or sparse.</td> <td data-bbox="850 1386 1505 1435">4. Can be only dense.</td> </tr> </tbody> </table>	Primary Index	Secondary Index	1. If the file containing the records is sequentially ordered, a <i>primary index</i> is an index whose search key also defines the sequential order of the file.	1. Indices whose search key specifies an order different from the sequential order of the file are called <i>secondary indices</i> .	2. There can be only one primary index for a relation.	2. There can be many secondary indices for a relation.	3. A sequential scan in primary index order is efficient.	3. Performance of sequential scan in secondary index order is poor.	4. Can be dense or sparse.	4. Can be only dense.
Primary Index	Secondary Index										
1. If the file containing the records is sequentially ordered, a <i>primary index</i> is an index whose search key also defines the sequential order of the file.	1. Indices whose search key specifies an order different from the sequential order of the file are called <i>secondary indices</i> .										
2. There can be only one primary index for a relation.	2. There can be many secondary indices for a relation.										
3. A sequential scan in primary index order is efficient.	3. Performance of sequential scan in secondary index order is poor.										
4. Can be dense or sparse.	4. Can be only dense.										
12.4	<p>Clustering indices may allow faster access to data than a non-clustering index affords. When must we create a non-clustering index despite the advantages of a clustering index? Explain your answer. [2007. Marks: 2]</p> <p>If we need to lookup a record using a search-key other than the search-key on which the file is stored sequentially, then we must create a non-clustering index to improve the performance of look-up.</p>										
12.5	<p>When is it preferable to use a dense index rather than a sparse index? Explain your answer. [2006, Marks: 2. 2004, Marks: 3]</p> <p>It is preferable to use a dense index instead of a sparse index when the file is not sorted on the indexed field (such as when the index is a secondary index) or when the index file is small compared to the size of memory.</p>										
12.6	<p>Why is sparse index used in database? [2002. Marks: 4]</p> <p>Sparse index is used in database because:</p> <ol style="list-style-type: none"> It requires less space. It imposes less maintenance overhead for insertions and deletions. 										
12.7	<p>What is the purpose of multilevel indexing? [2005. Marks: 1]</p> <p>The purpose of multilevel indexing is to reduce I/O operations on indices when records are searched.</p>										

12.8 Is it possible in general to have two primary indices on the same relation for different search keys? Explain your answer. [2007. Marks: 2]

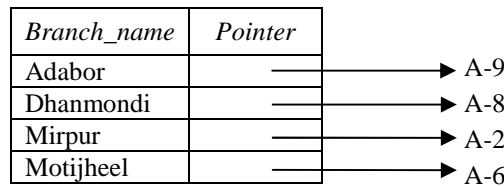
In general, it is not possible to have two primary indices on the same relation for different keys because the tuples in a relation would have to be stored in different order to have same values stored together. We could accomplish this by storing the relation twice and duplicating all values, but for a centralized system, this is not efficient.

12.9 Consider the following *dense primary index file* corresponding to the sequential file *Account* sorted on the attribute *branch_name*.

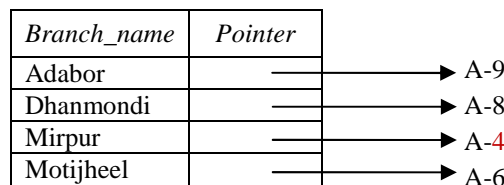


Now make necessary modification to the index file after deletion of the record for the account no 'A-5' and then 'A-2'. [2007. Marks: 1 + 1]

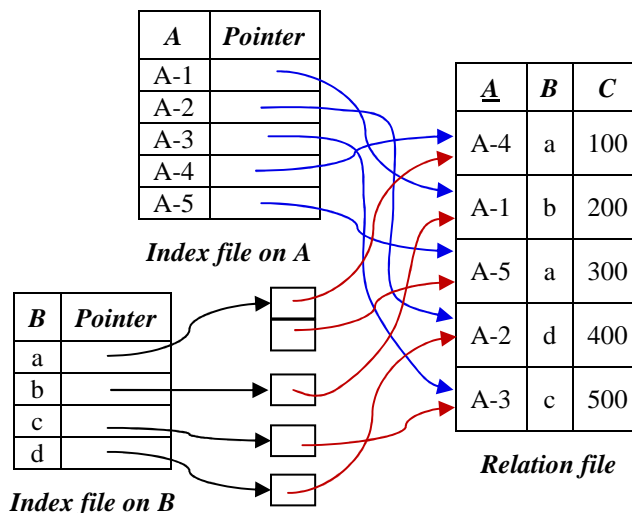
Index file after deletion of the record for the account no 'A-5':



Index file after deletion of the record for the account no 'A-2':



12.10 Let $R = (\underline{A}, B, C)$ is a relation schema with A as candidate key. The relation $r(R)$ is sorted on attribute C . Draw two secondary indices on candidate key A and non-candidate key B filling data for different attributes. [2007. Marks: 3]

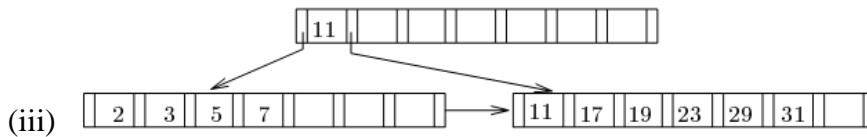
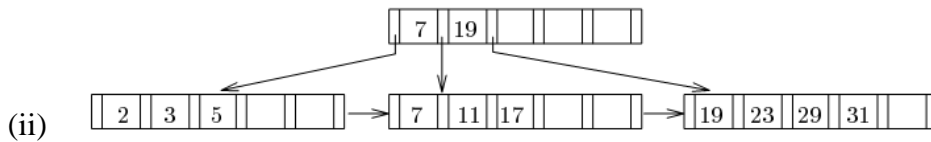
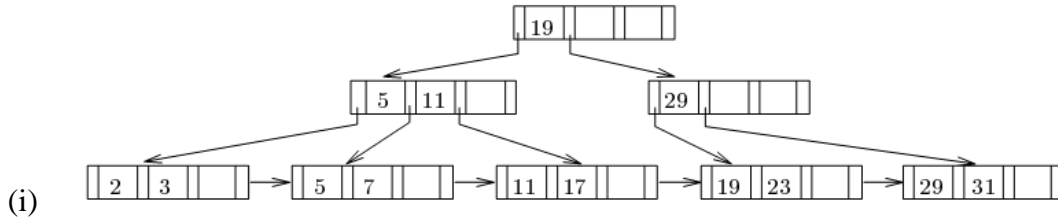


12.11 Consider a sequential file sorted with non-primary key. Draw a secondary index on the file for a search key which is the primary key of the file. [In-course 2, 2005. Marks: 2]
See Question and Answer 12.10 – Index file on A.

12.12 What are the disadvantages of index-sequential file? [In-course 2, 2005. Marks: 1]
The main disadvantage of the index-sequential file organization is that performance degrades as the file grows, both for index lookups and for sequential scans through the data. Although this degradation can be remedied by reorganization of the file, frequent reorganizations are undesirable.

12.13 Construct a B⁺ tree for the following set of key values and for the (i) four (ii) six (iii) eight pointers that will fit in one node: [2006, Marks: 3. 2003, Marks: 2. (each)]

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



12.14 For each B⁺ tree of Question and Answer 12.13, show the form of the tree after each of the following series of operations:

1. Insert 9
2. Insert 10
3. Insert 8
4. Delete 23
5. Delete 19

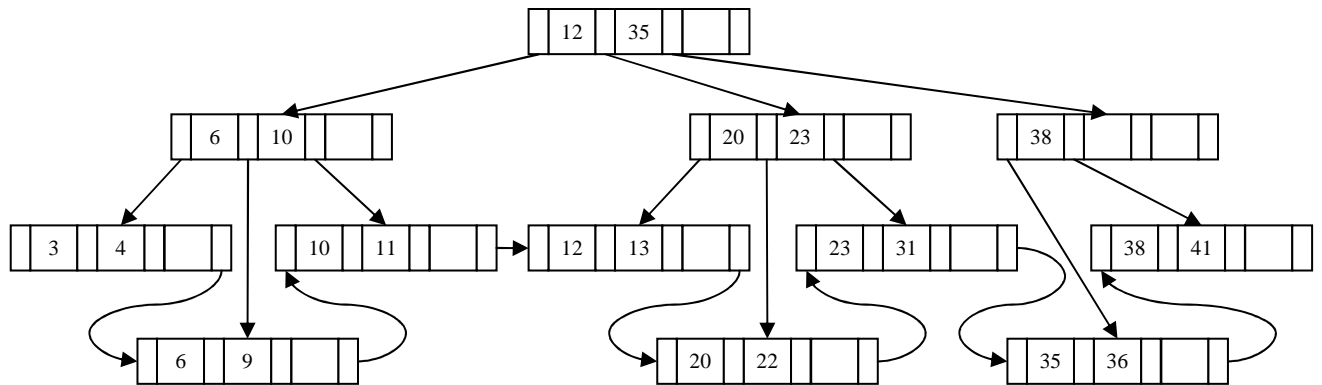
Structure	Operation	Form of Tree After Operation
(i)	1. Insert 9	
	2. Insert 10	
	3. Insert 8	

	4. Delete 23	
	5. Delete 19	
(ii)	1. Insert 9	
	2. Insert 10	
	3. Insert 8	
	4. Delete 23	
	5. Delete 19	
(iii)	1. Insert 9	
	2. Insert 10	
	3. Insert 8	
	4. Delete 23	
	5. Delete 19	

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

3, 4, 6, 9, 10, 11, 12, 13, 20, 22, 23, 31, 35, 36, 38, 41

Assume that the tree is initially empty and values are added in ascending order. The number of pointers that will fit in one node is four. [2004. Marks: 4]



12.16 For a B⁺ tree structure, search key value size = 12 bytes, pointer size = 8 bytes, block size = 388 bytes and there are 1000 search key values. How many nodes are required to access for an index lookup for the worst case? [In-course 2, 2005. Marks: 2]

Given,

Search-key value size, $S_k = 12$ bytes

Pointer size, $S_p = 8$ bytes

Block size, $S_b = 388$ bytes

Number of search-key values in file, $K = 1000$

$$\begin{aligned} \therefore \text{Maximum number of pointers in a node, } n &= \lfloor S_b / (S_k + S_p) \rfloor \\ &= \lfloor 388 / (12 + 8) \rfloor \\ &= 19 \end{aligned}$$

$$\begin{aligned} \therefore \text{Number of nodes required to access for an index lookup} &= \lceil \log_{\lceil n/2 \rceil} (K) \rceil \\ &= \lceil \log_{\lceil 19/2 \rceil} (1000) \rceil \\ &= 3 \end{aligned}$$

Answer: 3.

12.17 If an index structure occupies 1000 disk blocks, how many block reads will be required in the best case and the worst case to find a desired index entry, if no index entry is in the overflow block? [In-course 2, 2005. Marks: 2]

In the best case, 1 block read will be required to find a desired index entry.

In the worst case, $\lceil \log_2(1000) \rceil$ or 10 block reads will be required.

12.18 Describe the format of nodes of B⁺ tree. [2002. Marks: 2]

Non-leaf nodes: Each non-leaf node in the tree has between $\lceil n / 2 \rceil$ and n pointers, where n is fixed for a particular tree.

Root node: The root node can hold fewer than $\lceil n / 2 \rceil$ pointers, but not less than two pointers unless the tree consists of only one node.

Leaf nodes: Each leaf has between $\lceil (n - 1) / 2 \rceil$ and $n - 1$ values. The ranges of values in each leaf do not overlap.

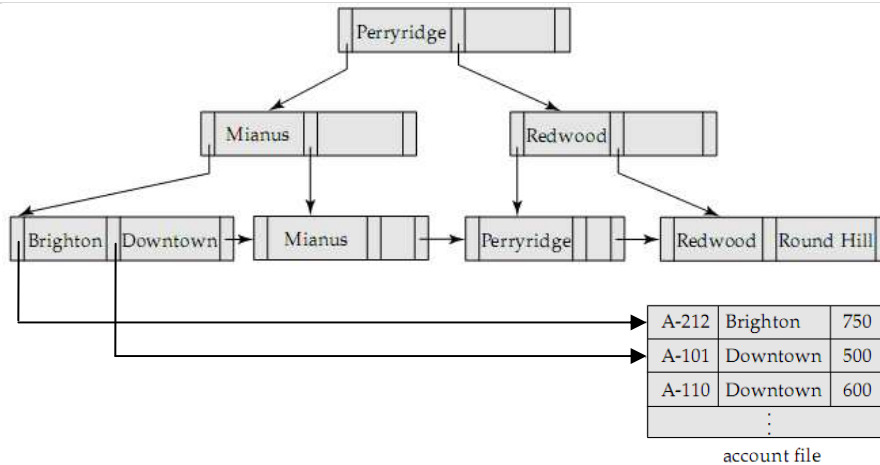


Figure: B+ tree node format. $[n = 3]$

12.19 What is the fan-out of B⁺ tree indexing? [2002. Marks: 2]
OR, What is fan-out of a node? [In-course 2, 2005. Marks: 1]

The number of pointers in a node of a B⁺ tree is called the *fan-out* of the node.

A non-leaf node has a fan-out between $\lceil n / 2 \rceil$ and n . The root node can hold fewer than $\lceil n / 2 \rceil$ pointers, but not less than two pointers unless the tree consists of only one node.

12.20 Why are the leaf nodes of a B⁺ tree chained together? [2007, In-course 2, 2005. Marks: 1]
OR, Why are nodes of a B⁺ tree at the leaf level linked? [2002. Marks: 2]

The leaf nodes of a B⁺ tree are chained together to allow for efficient *sequential processing* of the file.

12.21 What are the differences between B⁺ tree structure and in-memory tree structure? [2007; In-course 2, 2005; Marks: 2]

B ⁺ Tree Structure	In-Memory Tree Structure (Binary Tree)
1. Each node is large – typically a disk block – and a node can have a large number of pointers – 200 to 400.	1. Each node is small and has at most 2 pointers.
2. Fat and short.	2. Thin and tall.
3. If there are K search-key values in the file, the path for a lookup is no longer than $\lceil \log_{\lceil n/2 \rceil}(K) \rceil$.	3. In a balanced binary tree, the path can be of length $\lceil \log_2(K) \rceil$.

12.22 What are the differences between B-tree and B⁺ tree? [2002, Marks: 3. In-course 2, 2005, Marks: 4]

B ⁺ Tree	B-Tree
1. Some search key values may appear twice.	1. Search key values appear only once.
2. Contains redundant storage of search-key values.	2. Eliminates redundant storage.
3. Takes more space than a corresponding B-tree.	3. Takes less space than a corresponding B ⁺ tree.
4. No additional pointer for search key is needed.	4. As search-keys in non-leaf nodes appear nowhere else in the B-tree, an additional pointer field for each search key in a non-leaf node must be included.

12.23	<p>What are the causes of bucket overflow in a hash file organization? [2006, Marks: 1. 2005, 2004, 2003, Marks: 2]</p> <p>The causes of bucket overflow are :</p> <ol style="list-style-type: none"> 1. Insufficient buckets. Our estimate of the number of records that the relation will have was too low, and hence the number of buckets allotted was not sufficient. 2. Skew in the distribution of records to buckets. This may happen either because there are many records with the same search key value, or because the hash function chosen did not have the desirable properties of uniformity and randomness.
12.24	<p>What can be done to reduce the occurrence of bucket overflows? [2006. Marks: 1]</p> <p>To reduce the occurrence of overflows, we can:</p> <ol style="list-style-type: none"> 1. Choose the hash function more carefully, and make better estimates of the relation size. 2. If the estimated size of the relation is n_r and number of records per block is f_r, allocate $(n_r / f_r) \times (1 + d)$ buckets instead of (n_r / f_r) buckets. Here d is a fudge factor, typically around 0.2. Some space is wasted: about 20% of the space in the buckets will be empty. But the benefit is that some of the skew is handled and the probability of overflow is reduced.
12.25	<p>How bucket overflows are handled? [2003. Marks: 3]</p> <p>Bucket overflows can be handled using two techniques:</p> <ol style="list-style-type: none"> 1. Closed Hashing. If records must be inserted into a bucket and the bucket is already full, they are inserted into overflow buckets which are chained together in a linked list. 2. Open Hashing. In this technique, the set of buckets is fixed, and there are no overflow chains. Instead, if a bucket is full, the system inserts records in some other bucket in the initial set of buckets. When a new entry has to be inserted, the buckets are examined, starting with the hashed-to slot and proceeding in some <i>probe sequence</i>, until an unoccupied slot is found. The probe sequence can be any of the following: <ol style="list-style-type: none"> a. Liner probing. The interval between probes is fixed (usually 1). b. Quadratic probing. The interval between probes increases by some constant (usually 1) after each probe. c. Double hashing. The interval between probes is computed by another hash function.
12.26	<p>What are the causes of skew? [In-course 2, 2005. Marks: 2]</p> <p>Skew can occur for two reasons:</p> <ol style="list-style-type: none"> 1. Multiple records may have the same search key. 2. The chosen hash function may result in non-uniform distribution of search keys.
12.27	<p>For a customer relation, $n_{\text{customer}} = 40000$ and $f_{\text{customer}} = 50$. If the fudge factor is 0.2, how many buckets will be required to reduce bucket overflow? [2007, In-course 2, 2005 (similar) Marks: 2]</p> <p>Given,</p> $n_{\text{customer}} = 40000$ $f_{\text{customer}} = 50$ <p>Fudge factor, $d = 0.2$</p> $\therefore \text{Number of buckets required} = (n_{\text{customer}} / f_{\text{customer}}) \times (1 + d)$ $= (40000 / 50) \times (1 + 0.2)$ $= 960$ <p>Answer: 960.</p>

12.28	<p>Why is hash structure not the best choice for a search key on which range queries are likely? [2006. Marks: 1]</p> <p>A range query cannot be answered efficiently using a hash index; we will have to read all the buckets. This is because key values in the range do not occupy consecutive locations in the buckets; they are distributed uniformly and randomly throughout all the buckets.</p>								
12.29	<p>Give a comparison of static hashing and dynamic hashing. [2004. Marks: 3]</p> <p>In static hashing, the set of bucket addresses is fixed. As databases grow or shrink over time, use of static hashing results in degradation of performance or wastage of space.</p> <p>In dynamic hashing, the hash function can be modified dynamically to accommodate the growth or shrinkage of the database.</p>								
12.30	<p>Compare closed and open hashing. [2007, In-course 2, 2005. Marks: 2]</p> <table border="1" data-bbox="193 600 1508 981"> <thead> <tr> <th data-bbox="193 600 847 651">Closed Hashing</th> <th data-bbox="847 600 1508 651">Open Hashing</th> </tr> </thead> <tbody> <tr> <td data-bbox="193 651 847 808">1. On bucket overflow, records are inserted into overflow buckets which are chained together in a linked list.</td> <td data-bbox="847 651 1508 808">1. Records are inserted in some other bucket in the initial set of buckets using a probe sequence (linear probing, quadratic probing etc.).</td> </tr> <tr> <td data-bbox="193 808 847 860">2. Deletion under closed hashing is simple.</td> <td data-bbox="847 808 1508 860">2. Deletion under open hashing is troublesome.</td> </tr> <tr> <td data-bbox="193 860 847 981">3. Preferable in database systems as insertion-deletion occurs there frequently.</td> <td data-bbox="847 860 1508 981">3. Preferable in compilers and assemblers as they perform only lookup and insertion operations in their symbol tables.</td> </tr> </tbody> </table>	Closed Hashing	Open Hashing	1. On bucket overflow, records are inserted into overflow buckets which are chained together in a linked list.	1. Records are inserted in some other bucket in the initial set of buckets using a probe sequence (linear probing, quadratic probing etc.).	2. Deletion under closed hashing is simple.	2. Deletion under open hashing is troublesome.	3. Preferable in database systems as insertion-deletion occurs there frequently.	3. Preferable in compilers and assemblers as they perform only lookup and insertion operations in their symbol tables.
Closed Hashing	Open Hashing								
1. On bucket overflow, records are inserted into overflow buckets which are chained together in a linked list.	1. Records are inserted in some other bucket in the initial set of buckets using a probe sequence (linear probing, quadratic probing etc.).								
2. Deletion under closed hashing is simple.	2. Deletion under open hashing is troublesome.								
3. Preferable in database systems as insertion-deletion occurs there frequently.	3. Preferable in compilers and assemblers as they perform only lookup and insertion operations in their symbol tables.								
12.31	<p>What are the limitations of hashing? [In-course 2, 2005. Marks: 2]</p> <p>The limitations of hashing are:</p> <ol style="list-style-type: none"> 1. Hash function must be chosen when the system is implemented and it cannot be changed easily thereafter if the file being indexed grows or shrinks. 2. Since the hash function maps search-key values to a fixed set of bucket addresses, space is wasted if the set of buckets is made large to handle further growth of the file. 3. If the set of buckets is too small, they will contain records of many different search-key values and bucket overflow can occur. As the file grows, performance suffers. 								
12.32	<p>Discuss the use of the hash function in identifying a bucket to search. [2002. Marks: 3]</p> <p>Since it cannot be known at design time precisely which search-key values will be stored in the file, such hash function should be chosen that assigns search-key values to buckets in such a way that the distribution has these qualities:</p> <ol style="list-style-type: none"> 1. Uniform: The distribution is uniform. The hash function assigns each bucket the same number of search-key values from the set of all possible search-key values. 2. Random: The distribution is random. In the average case, each bucket will have nearly the same number of values assigned to it regardless of the actual distribution of the search-key values. <p>Hash functions require careful design. A bad hash function may result in lookup taking time proportional to the number of search keys in the file. A well-designed function gives an average-case lookup time that is a small constant, independent of the number of search-keys in the file.</p>								