© ALL RIGHTS RESERVED WITH THE PUBLISHERS

(All rights resolved, no part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written permission of the publisher. Any violation thereof shall lead to legal recourse without any further notice.)

OUR MAJOR BOOKSELLERS :

Kewal Book Depot, Tilak Nagar
Bharat Book Depot, Jawahar Nagar
Jain Book Shop, Karampura
Madaan Book House, Nai Sarak
Pandey Book Stall, Laxmi Nagar
Lamba Book Depot, Tilak Nagar
Aggarwal Book House, Rohini
Bhatia Book Depot, Kalkaji
Jagdish Book Shop, Bersarai
Rakesh Book Shop, Yamuna Vihar
Satija Book Depot, Pritampura
Vijay Book Depot, Paschim Vihar
Raj Book Agency, Dwarka
Madaan Book Collection , Pritampura
Satija Book Centre, Rohini
Jatin Book Shop , Hari Nagar
Aadesh Book Shop, Geeta Colony
Prakash Brothers, Gole Market

Published by :
AB Publishers
4278/3, Ansari Road, Darya Ganj,
New Delhi-110002
Ph. : 0-9818257423, 9971086337

Note : As publishers, we are committed to serve the student community with the best of our resources. We take every care to eliminate errors during the course of editing printing of books. Therefore, we state that authors, publishers and sellers should not be held responsible for any mistake that might have crept in advertently.

Printed at : Delhi

SYLLABUS

[Academic Session 2015-16]

DATABASE MANAGEMENT SYSTEM (ETCS-208)

Maximum Marks : 75

Instruction to Paper Setters:

1. *Question No. 1 should be compulsory and cover the entire syllabus. This question should have objective or short answer type questions. It should be of 25 marks.*
2. *Apart from Q. No. 1 rest of the paper shall consist of four units as per the syllabus. Every unit should have two questions. However, student may be asked to attempt only 1 question from each unit. Each question should be 12.5 marks.*

UNIT–I

Introductory Concepts of DBMS: Introduction and application of DBMS, Data Independence, Database System Architecture – levels, Mapping, Database users and DBA, Entity – Relationship model, constraints, keys, Design issues, E-R Diagram, Extended E-R features- Generalization, Specialization, Aggregation, Translating E-R model into Relational model.

[T1, T2] [No. of Hours: 10]

UNIT–II

Relational Model: The relational Model, The catalog, Types, Keys, Relational Algebra, Fundamental operations, Additional Operations-, SQL fundamentals, DDL,DML,DCL PL/SQL Concepts, Cursors, Stored Procedures, Stored Functions, Database Integrity – Triggers.

[T2, R3] [No. of Hrs. 10]

UNIT–III

Functional Dependencies, Non-loss Decomposition, First, Second, Third Normal Forms, Dependency Preservation, Boyce/Codd Normal Form, Multi-valued Dependencies and Fourth Normal Form, Join Dependencies and Fifth Normal Form.

[T2, R1, R3][No. of Hrs. 10]

UNIT–IV

Transaction Management: ACID properties, serializability of Transaction, Testing for Serializability and concurrency control, Lock based concurrency control (2PL, Deadlocks), Time stamping methods, Database recovery management.

Implementation Techniques: Overview of Physical Storage Media, File Organization, Indexing and Hashing, B+ tree Index Files, Query Processing Overview, Catalog Information for Cost Estimation, Selection Operation, Sorting, Join Operation, Materialized views, Database Tuning.

[T1, T2, R2][No. of Hrs. 12]

Scanned by CamScanner

YOU KNOW WHAT I'VE NOTICED, HOBBES? THINGS DON'T BUG YOU IF YOU DON'T THINK ABOUT THEM.

SO FROM NOW ON, I SIMPLY WON'T THINK ABOUT ANYTHING I DON'T LIKE, AND I'LL BE HAPPY ALL THE TIME!

DON'T YOU THINK THAT'S A PRETTY SILLY AND IRRESPONSIBLE WAY TO LIVE?

WHAT A PRETTY AFTERNOON.

Scanned by CamScanner

FIRST TERM EXAMINATION
(B.TECH.)—2012
DATABASE MANAGEMENT SYSTEM

Time : 3 hrs. **M.M. : 75**

Note: ***Attempt any five questions including Q.No 1 which is compulsory. Select one question from each unit.***

Q.1. (*a*) What is the reason for having null values in a relation? **[2]**

Ans. Nulls may be introduced into the relation because the actual value is either unknown or does not exist.

For example,

1. An employee whose address has changed and whose new address is not yet known should be retained with a null address.
2. If employee types have a composite attribute dependents, and a particular employee has no dependents, then that tuple's dependents attribute should be given a null value.

Q.1.(b) Differentiate between primary key and super key? **[2]**

Ans. Refer Q. No.3.a. of first trem 2014

Q.1.(c) Differentiate between primary key and super key? **[2]**

Ans. Refer Q. No.5.a. of End trem 2012

Q.1.(*d*) What are composite and multi valued attributes in E-R model? **[2]**

Ans. Composite attribute: A composite Attribute is defined as an attribute composed of multiple components each with an independent existence. In this case, some attributes can be further divided to yield smaller components with an independent existence of their own.

For example, the attribute name is a composite attribute which can be further subdivided into fname, mname and lname *i.e.*, first name, middle-name and last-name.

Multi-valued Attribute: It is defined as an attribute that multiple values for a single entity. For example, for a student entity, if one of the attributes is 'Hobby', then 'Hobby' can have multiple values be reading, dancing, gardening and so on.

Q.1.(*e*) Select and where in SSQL are analogous to which operations in relational algebra? **[2]**

Ans. For example:

Original Table:

EMP

emp-id	ename	Age	Salary
1	Amit	18	8000
2	Rajat	20	9000
3	Kamal	25	10000

If we find out all those employee which have age less that 19.

Scanned by CamScanner

In SQL: Select * from EMO
WHERE AGE < 19

In Relational Algebra: σAGE < 19 (EMP)

Output:

emp-id	ename	Age	Salary
1	Amit	18	8000

Q.2.(a) Distinguish between logical and physical data independence? Which one is more difficult to achieve and why? **[6]**

Ans. Logical data independence is **difficult** to achieve that the **physical data independence.** This is because logical data independence requires flexibility in the design of database and the programmer has to foresee the future requirements or modifications in the design.

Let us tabulate the differences between the two.

S. No.	Physical data Independence	Logical Data Independence
1.	It is the ability to modify the physical schema without causing application programs to be written.	It is the ability to modify the logical schema without causing application programs to be rewritten.
2.	Modification at the physical level are necessary to improve performance.	Modification at the logical level are necessary whenever the logical structure of the database is altered.
3.	It is not difficult to be achieved.	It is more difficult to achieve as application programs are dependent on the logical structure of the data that they access.
4.	It provides immunity of conceptual or external schemes.	It provides immunity of external schemes or application programs.
5.	The conceptual scheme insulate the users form changes in physical storage of data.	The users are shielded from changes in the logical structure of the data or changes in choice of relations to be stored.

Q.2.(b) What are the responsibilities of DBA? **[4]**

Ans. DBA has central control over the system. The database administrator is responsibles for following functions:

(*i*) Schema Design and maintainance,
(*ii*) Physical shema and organisation Modification,
(*iii*) Authorization and secutiry,
(*iv*) Integrity constraint specification,
(*v*) Recovery from failure and give backup,
(*vi*) Database upgradation,
(*vii*) Granting access to users,
(*viii*) Physical database design.

DBA is the central controller of the database system who manages all resources like database. DBMS and related software. A DBA provides the necessary technical support for implementing policy decision of databases.

Q.3. Construct an ER diagram for a department store. This database contains information about customers, products they order, departments in the store, employees, and suppliers of products. For customers we record their name, address, phone number, balance and orders they have placed. The orders are for certain products and should include a description of the product, product number and quantity. Products are usually being supplied by more than one supplier at different prices. The store needs to keep contact information about the suppliers. The store is divided into departments. An employee cannot work in more than one departments of the store. Each department is managed by a manager. For employees the store needs to keep their name, address phone number and salary. **[10]**

Ans.

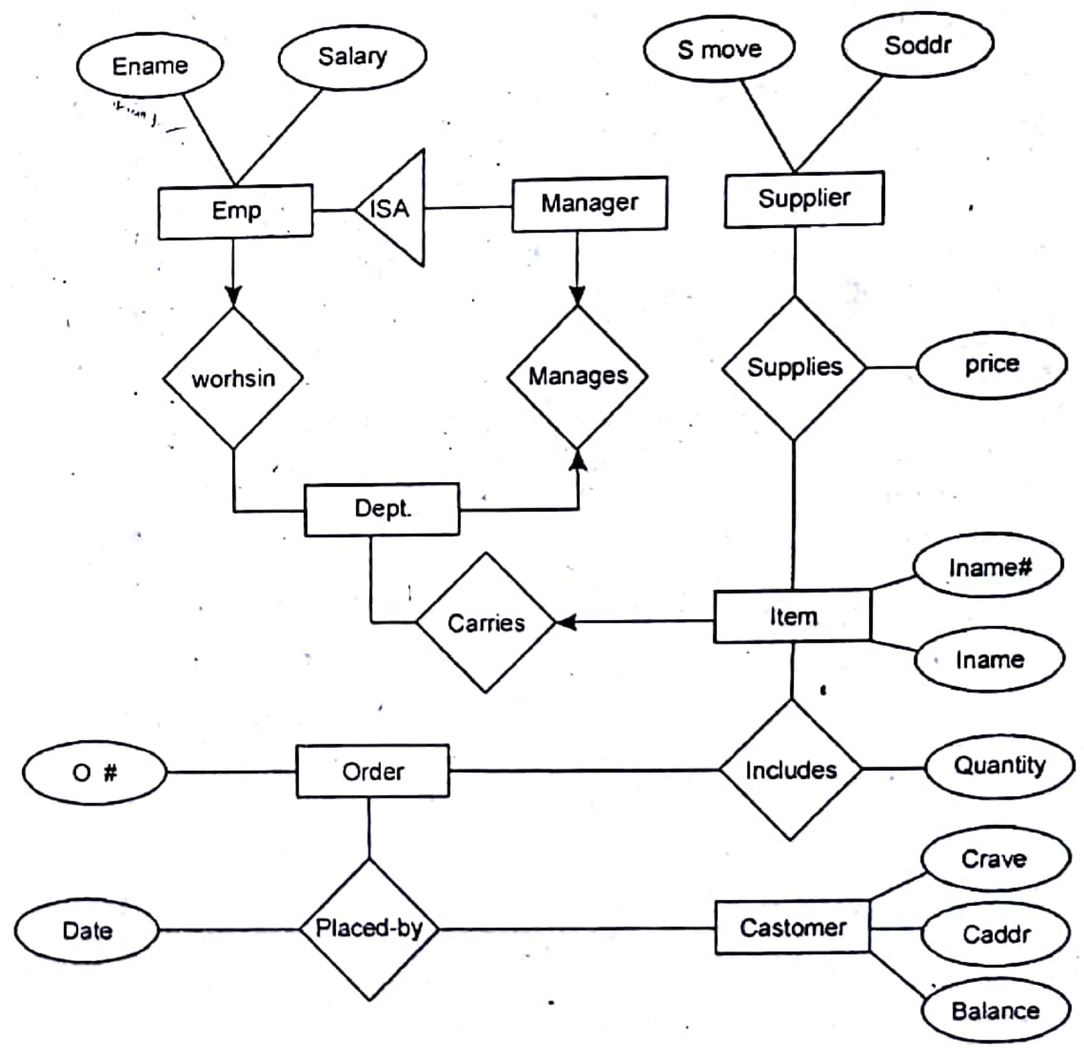


Fig. E-R Diagram for Department store

Scanned by CamScanner

Q.4. Consider the following relational schema: **[10]**

Employee(Emp_number, Emp–name, salary)

Project(Proj_ number, Proj_name, Proj_number)

For each of the following queries given an expression in SQL and relational algebra:

(*a*) Find the name of the projects at Delhi.

Ans. In SQL:

```
SELECT EMPLOYEE.Emp_name from
Employee, Project, works_on
WHERE Employee, Emp_number = works_on. Emp_number
and Project. Proj_number = works_on Proj_number
and Proj_location = 'Delhi';
```

In Relational Algebra:

```
πEMPLOYEE.Emp_name (σProj-location = 'Delhi'
(σEMPLOYEE.  Emp_number = works_on. Emp_number ^
Project. Proj_number = works_on. Proj_number_
(EMPLOYEE × WORKS_ON × PROJECT)))
Replace the last selection condition σ by a natural join
(EMPLOYEE ⋈ WORKS_ON ⋈ PROJECT)
```

(*b*) Find the name and salary of employees working on Proj number 10 at Noida.

Ans. In SQL:

SELECT EMPLOYEE.Emp_name, EMPLOYEE.SALARY
from EMPLOYEE, PROJECT, WORKS_ON
WHERE PROJECT Proj_number = '10' and
Employee.Emp_number = works_on.Emp_number
and PROJECT Proj_location = 'Noida';

In Relational Algebra:

```
πEMPLOYEE_Emp_name,EMPLOYEE.SALARY
(σPROJECT.Proj_number = '10' PROJECT.Proj_location='NOIDA'
(EMPLOYEE ⋈ PROJECT ⋈ WORKS_ON))
```

SECOND TERM EXAMINATION (B.TECH.)—2012

DATABASE MANAGEMENT SYSTEM

Time : 3 hrs. **M.M. : 75**

Note: ***Attempt any five questions including Q.No 1 which is compulsory. Select one question from each unit.***

Q.1. (*a*) What do you mean by partially committed state of a transaction? **[2]**

Ans. When the final statement has been executed, we say, transaction is in a partially committed state. At this time, the transaction has completed its execution but it is still possible that it may have to be aborted as the actual output may still be temporarily residing in the main memory. DBMS then writes information to disk that even if the system failure occurs, the updates performed by *Ti* can be can be recreated when the final or last of this information is written out, the transaction enters the committed state.

Q.1.(*b*) What is dependency preservation property of a decomposition? **[2]**

Ans. The dependency preservation decomposition is another property of decomposed relational database scheme, *D* in which each *FD*, $x \rightarrow y$, specified in *F* either apperared directly in one of the relation schemes *Ri* in the decomposed *D* or could be inferred form the dependencies that appear in some *Ri*.

The Decomposition, $D = \{R_1, R_2 --- R_m\}$ of *R* is said to be dependency preserving with respect of *F* if the union of the projections of *F* on each *Ri* in *D* is equivalent to *F*.

That is,

$R \subset$ join of R_1, R_1 over x

Q.1.(*c*) Explain the pseudo transitive inference rule? **[2]**

Ans. If $\propto \rightarrow \beta$ holds and $\upsilon\beta \rightarrow \delta$ holds, then $\propto \upsilon \rightarrow \delta$ holds.

For example:

$$A \rightarrow C$$
$$CG \rightarrow I$$
$$\therefore \quad AG \rightarrow I \text{ Pseudo transitivity}$$

Q.1.(d) What is the difference between implicit and explicit local? **[2]**

Ans. NOT IN NEW SYLLABUS

Q.1.(*e*) Given *R* (*A, B, C, D, E, F, G*) and a set of functional dependencies? *f* = {*A* → *B, BC* → *DE, AEF* → *G*} Find {*A, C*}⁺? **[2]**

Ans.

$$X^{(0)} = \{A, C\}$$
$$X^{(1)} = \{A, B, C\}$$
$$X^{(2)} = \{A, B, C, D, E\}$$
$$X^{(3)} = \{A, B, C, D, E\}$$
$$X^{(2)} = X^{(3)}$$

Scanned by CamScanner

Hence. $\{A, C\}^+ = \{A, B, C, D, E\}$

Q.2.(a) Define two phase locking? What benefits does strict two phase locking provide?

Ans. Refer Q. No.1.d. of second trem 2016

Q.2.(b) Which of the following schedules are conflict serializable? Also, for each conflict serializable schedule find the equivalent serial schedule? [5]

(1) $r1(x)$; $r3(x)$; $w1(x)$; $r2(x)$; $w3(x)$

(2) $r3(x)$; $r2(x)$; $w3(x)$; $r1(x)$; $w1(x)$

(where ri and wi denotes the read and write operations of ith transaction)

Ans.2.b. Refer Q. No.1; pg 31-2016

Q.3. Given the relation schema [10]

```
Car_Sale (car#, salesman #, Datesold,
Commission%,Discount_amt)withfunctional dependencies:
Datesold →Discount_amt
Salesman # →Commission %
Car # →Datesold
```

(a) Find the primary key for the relation?

(b) Comment on the present state of the relation and normalize till BCNF.

Ans. (a) Assume that car # and salesman # is the primary key.

(b) Given the relation scheme Car_sale with the functional dependencies that relation satisfies 1NF but not 2NF (Car # → Date_sold and car#→disscount_) so these two attributes are not FFD on the primary key and not in 3 NF to normalize.

2NF:

```
Car_Sale1 (Car#, Date_sold, Discount_ant)
Car_sale2 (Car#, Salesman#)
Car_sale3 (Salesman#, Commission%)
```

3NF

```
Car_sale1-1 (Car#m Date_sold)
Car_Sale1-2 (Date_sold, Discount_ant)
Car_sale2 (Car#,Salesman#)
Car_sale3 (Salesman#, Commission%)
```

Q.4.(a) What is Thoma's write rule. Explain [5]

Ans. Thoma's write rule which states that "in order to increase the concurrency and to reduce the rollback of time stamp based protocols, obsolete write operations can be ignored under certain circumstances."

So, the protocol rules for read operations remain unchanged whereas for write operations, they are slightly different from our timestamp ordering protocol.

To understand the concept of Thomas write rule let as consider the following schedule:

T_1	T_2
read(A,a)	
	write(A,a)
write(A,a)	

Apply the timestamp ordering protocol on above schedule, since T_1 starts before T_2, so $TS(T_1) < TS(T_2)$. The read (A, a) operation of T_1 succeeds, as does the write (A, a) operation of T_2. When T_1 attempts its write (A, a) operations, we find that $TS(T_1) < w$ – timestamp (A), since w – timestamp $(A) = TS(T_2)$. Thus write (A, a) by T_1 is rejected and T_1 is required by the timestamp ordering protocol it is unnecessary. Since T_2 has already written A, the value of T_1 is attempting to write is one that will never need to be read. Any transaction Ti with $TS(Ti) < TS(T_1)$ that attempts a read (A) will be rolled back, since $TS(Ti) < w$ – timestamp (A) and any transaction Tj with $TS(Tj) > TS(T_2)$ must read value of A written by T_2, rather than the value written by T_1.

Q.4.(b) What is difference between immediate and deferred database modification with respect to recovery? [5]

Ans.

S.No.	Deferred Database	Immediate Database
1.	It ensures transaction atomicity by recording all database modification in the log, by deferring the execution of all write operations of a transaction until the transaction partially commits.	1. It allows database modification to be output to the database while the transaction is still in the active transactions are called "uncommitted modification"
2.	The format of log Record: <*Ti, Xj, Vnew*> Where *Ti* is transaction identifier *xj* is the data item, *Vnew* is the modified data item *Xj*.	2. Format of log Record: <*Ti, Xj, Vold, Vnew*> Where *Ti* is transaction identifier, *Xj* is the data item, *Vold* is the old value of data item *Vnew* is new value of *Xj*.
3.	It will be done by Undo (*Tj*) only.	3. It will be done by Undo (*Tj*) as well as Redo (*Tj*).
4.	This approach save the stable storage space & reduce the compexity of log record.	4. Using the log system can handle any failure that result in the loss of info. of volatile storage.

Scanned by CamScanner

END TERM EXAMINATION
(B.TECH.)—2012
DATABASE MANAGEMENT SYSTEM.

Time : 3 hrs. **M.M. : 75**

Note: ***Attempt any five questions including Q.No 1 which is compulsory. Select one question from each unit.***

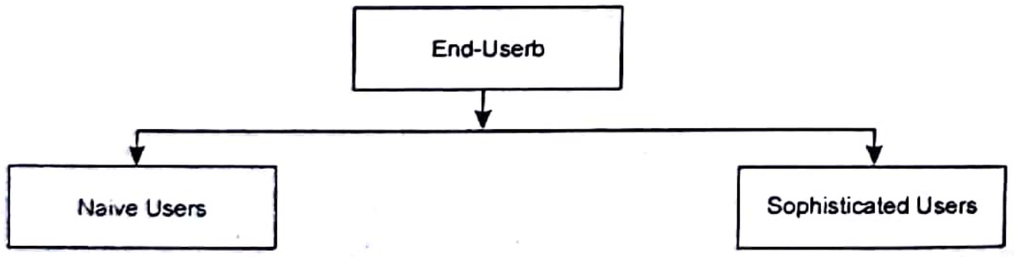
Q.1. (*a*) What do you understand by conceptual model of a database? [2]

Ans. The term conceptual model may be used to refer to models which are formed after a conceptualization process in mind. Conceptual models represent human intentions or semantics. Conceptualization from observation of physical existence and conceptual modelling are necessary means human employ to think and solve problems. Concepts are used to convey semantics during various natural languages based communication. Since a concept might map to multiple semantics by itself, an explicit formalization is usually required for identifying and locating the intended semantic from several candidates to avoid misunderstandings and confusions in conceptual models.

"Conceptual modelling is the activity of formally describing some aspects of the physical and social world around us for the purposes of understanding & communication"

Q.1.(*b*) Differentiate the terms Database user and Database administrator? [2]

Ans. Database Users: The clients for the database are the end-users. End-users are classified as follows:



Database Users

Naive users: They are usually unware of the DBMS. They access the database by using specially written application program.

Sophisticated Users: The sophisticated end-user is familiar with the structure of the database and the facilities that DBMS provides.

Database Administrator: A DBA is an individual person or group of persons with an overview of one or more databases so that he/she can control the design and the use of these databases. A DBA is the highest salary paid person in an organisation. A DBA provides the necessary technical support for implementing policy decision of databases.

The main functions of a DBA are follows:

1. Defining conceptual Schema.
2. Physical database design.
3. Security and Integrity checks.
4. Give backup and recovery strategies.
5. Granting access to users.

Q.1.(c) Give example of Generalization and Aggregation in E-R diagram ? [2]

Ans. Refer Q. No.3.b. of End Term 2015 ; Q.No.1.c. of First term 2014

Q.1.(*d*) Define Multivalued attribute and composite attribute in and composite attribute in ER - Diagram? [2]

Ans. Refer to FTE (Sep. - 2012) in Q. No. 1(d).

Q.1.(*e*) List the reasons for introduction a null value in a database? [2]

Ans. Refer to ETE (Dec., – 2012) in Q.No. 3(a)(iii).

Q.1.(*f*) What is division operator and how it is useful? [2]

Ans. The division operator divides a dividend relation, A, of degree (m + n) by a divisor relation, B, of degree, n, and produces a resultant relation of degree, m.

For example: Relation A has 1-table, as follows.

S.No	P.No
S1	P1
S1	P2
S1	P3
S1	P4
S1	P5
S1	P6
S2	P1
S2	P2
S3	P2
S4	P2
S4	P4
S4	P5

Relation B has 3-tables as follows:

P.No
P1

P.No
P2
P4

P.No
P1
P2
P3
P4
P5
P6

Then, A DIVIDE By B gives the following resultant tables for all the three, cases:

Scanned by CamScanner

Case 1

S.No
S1
S2

Case 2

S.No
S1
S4

Case 3

S.No
S1

Q.1.(g) Differentiate natural join and outer join operations? **[2]**

Ans. Natural Join: Natural join between two relations is written as (R ⋈ S). The result of the natural join is the set of all combinations of tuples in R and S that are equal on their common attribute names.

Outer Join: An outer join operations returns all the rows returned by an inner join plus all the rows from one table that do not match any row form the other table. An outer join can be left, right or both tables are retained.

For Example: Consider the tables Employee and department and their Natural, Outer join.

Department

Dept Id	Dept Name
d–01	Finance
d–02	Sales
d–03	Marketing

Employee

Name	Emp Id	Dept Id
Khanak	e–01	d–01
Kavita	e–02	d–02
Alpesh	e–03	d–01
Aruna	e–04	d–02

Result of Natural join: Employee ⋈ Department

Name	Emp id	Dept Id	Dept Name
Khanak	e–01	d–01	Finance
Kavita	e–02	d–02	Sales
Alpesh	e–03	d–01	Finance
Aruna	e–04	d–02	Sales

Result of Outer join: Employee ⋈ Department

Name	Emp Id	Dept Id	Dept Name
Khanak	e – 01	d – 01	Finance
Kavita	e – 02	d – 02	Sales
Alpesh	e – 03	d – 01	Finance
Aruna	e – 04	d – 02	Sales
NULL	NULL	d – 03	Marketing

Q.1.(h) What is query correlation? **[2]**

Ans. 1(h) Query Correlation: In a SQL database query, a correlated query (also known as a synchronized sub query) is a subquery (a query nested inside another (very) that uses values from the outer query. The subquery is evaluated once for each row processed by the outer query.

Q.1.(i) Define Functional Dependency? **[2]**

Ans. Yes, we can infer a functional dependancy from a given relational state.

"An attribute in a relational model is said to be functionally dependent on another attribute in the table if it can take only one value for a given value of the attribute upon which it is functionally dependent."

Functional Dependency (FD): A functional dependency is an association between two attributes of the same table (Relation). One of the attributes is called as the **determinant** and other attributed is called as the **determined.** Note here that for each value of the determinant there is associated one and only one value of the determined.

For example:

Supplier Table (SP)

S.No.	Sname	Status	City
S_1	Ajay	11	Rohtak
S_2	Mayur	12	Delhi
S_3	Rahul	13	Rohtak

Here, sname is FD on S_{no}, why? This is because Sname can take only one value for the given value of S_{no} (e.g. S_2 for Mayur). We can also say that there must be one Sname for supplier number, S_2.

∴ The FD is represented as :

$$S_{no} \rightarrow Sname$$

Here, FD is shown by an arrow (→) which shows that Sname is functionally dependent on Sno.

Similarly, city and status are also *FD* on *S*no, because for each value of S_{no}, there will be only one city and status. These FDs are represented as follows :

$$SNo \rightarrow City$$
$$Sno \rightarrow Status$$

We can combine these FDs also and write them as follows :

$$SP.Sno \rightarrow SP\ (Sname, city, status)$$

Q.1.(j) Define Atomicity, Durability, Isolationand Consistency ? **[2]**

Ans. Refer Q. No.4.b. : Second Term -2016

Scanned by CamScanner

Q.1.(k) What is the need for checkpoint? [2]

Ans. Typical, I/O operations use buffers to match the CPU speed and the fast main memory. The buffer manager component of the OS does the assignment and management of memory blocks. It is responsible for the efficient management of the database buffers that are used to flush pages between buffer and secondary storage. The buffer management provides a temporary copy of a database page.

Non-Volatile Storage Failure: The basic technique for non-volatile storages is to dump the entire content of the database to stable storage periodically (say, once per day). For example, we may dump the database to one or more magnetic tapes. If a failure occurs that results in the loss of physical database blocks, then the most recent dump is used in restoring the database to a previous consistent state. Once this restorating a accomplished, the system uses the log to bring the database system to the most recent consistent state. To recover from the loss of non-volatile storage, we restore the database to disk using the most recent dump. Then the log is consulted and all the transactions that have committed since the most recent dump occured are done. No undo operations need to be executed because dump already contains the old values of data items. Also note that a dump of the database contents is called as an archival dump because we can archieve the dumps and use them later to examine old states of the database. Dumps of a database and checkpoining of buffers are similar.

Checkpoints can be helpful in Recovery: The checkpoint is defined as the point of synchronisation between the database and the transaction log file. As we know that the general method of database recovery is using information in the transaction log. So, we need to search the entire log. This is time consuming. So, we use checkpoints. A checkpoint is like a snapshot of the DBMS state. By taking checkpoints periodically, DBMS can reduce the amout of work to be done during restart the amount of work to be done during restart in the event of a system crash. The approach is to find a point that is suficiently far back to ensure that any item written before that point has been done correctly and stored safety. This method is called as checkpointing. The following actions are performed when a system issues a checkpoint :

- Output onto stable storage, all the records currently residing in main memory.
- Output on the disk, all modified buffer blocks.
- Output on the stable storage, a log record <checkpoint>

Q.1.(l) When is a log record written to log file ? [2]

Ans. Refer Q.No.3.b. of Second Term 2013

UNIT

Q.2(a) Differentiate between hierarchial and network model of a database by taking a suitable example. [6.5]

Ans. NOT IN NEW SYLLABUS

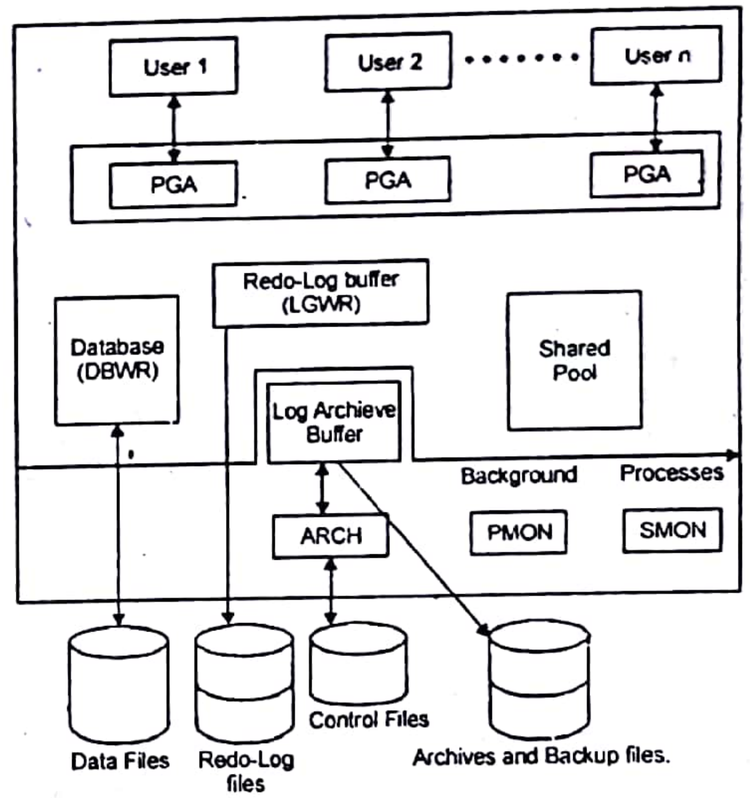
Q.2.(b) What are different subsystems in DBMS Architecture and what functionalities they provide? [6]

Ans. When oracle uses the names like Oracle 8 server, Oracle 8i server or Oracle 8i server to a specific database offering, it refers to what was formerly known as work group server.

The set of processes that constitute an instance of the server's operation is called as oracle instance. It consists of

1. System Global Area (SGA)
2. Set of background processes.

1. System Global Area (SGA): Oracle uses an area of memory'to store database information which is shareable and it assigns it when an instance states. This area of memory is called as SGA.



Block Diagraim Architecture of Oracle 8

SGA has various memory structure like:

(i) **Database buffer cache:** It is cache memory that is used to store the most frequently accessed data blocks from the database. So, now the I/O activity can be significantly reduced.

(ii) **Reado log Buffer:** A buffer for the redo log file. It is used for recovery purposes.

(iii) **Shared Pool:** It contains shared memory constructs like shared SQL areas. These areas contain the parse trees of SQL queries as well as their execution plans.

2. Set of Background Processes:

(a) **User Processes:** Each user process corresponds to the execution of some applications or some tool.

(b) **Program Global Area (PGA):** It is a memory buffer that contains data and control information for a server process. It is created by oracle when a server process is started.

(c) **Oracle Processes:** A process is a thread of control or a mechanism in OS to execute a series of steps. A process has its own private memory area where it runs.

Scanned by CamScanner

Q.3.(a) In E–R Diagram what can be different types of cardinalities? Give an example of each. **[6]**

Ans. Connectivity or Cardinality: The cardinality of a relationship describes the mapping of associated entity instances in the relationship. The values of connectivity are "one" or "many". While the actual count of elements associated with the connectivity is called as the cardinality of the relationship connectivity.

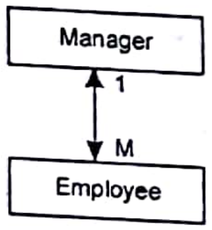
There are three basic types of connectivity for relations and they are as follows:

(a) One-to-one (1 : 1): A 1 : 1 relationship is when at most one instance of entity (say, A) is associated with one instance of another entity (say, B).

1 : 1 Relationship

(b) One-to-May (1 : M): A 1 : M is when for one instance of entity A there are zero, or many instances of entity B.

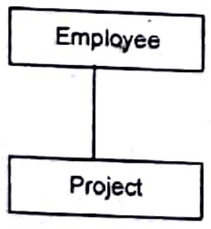
For example:



1 : M Relationship

(c) Many-to-Many: A M : N relationship is when for one instance of entity A, there are zero, one or many instances of entity B.

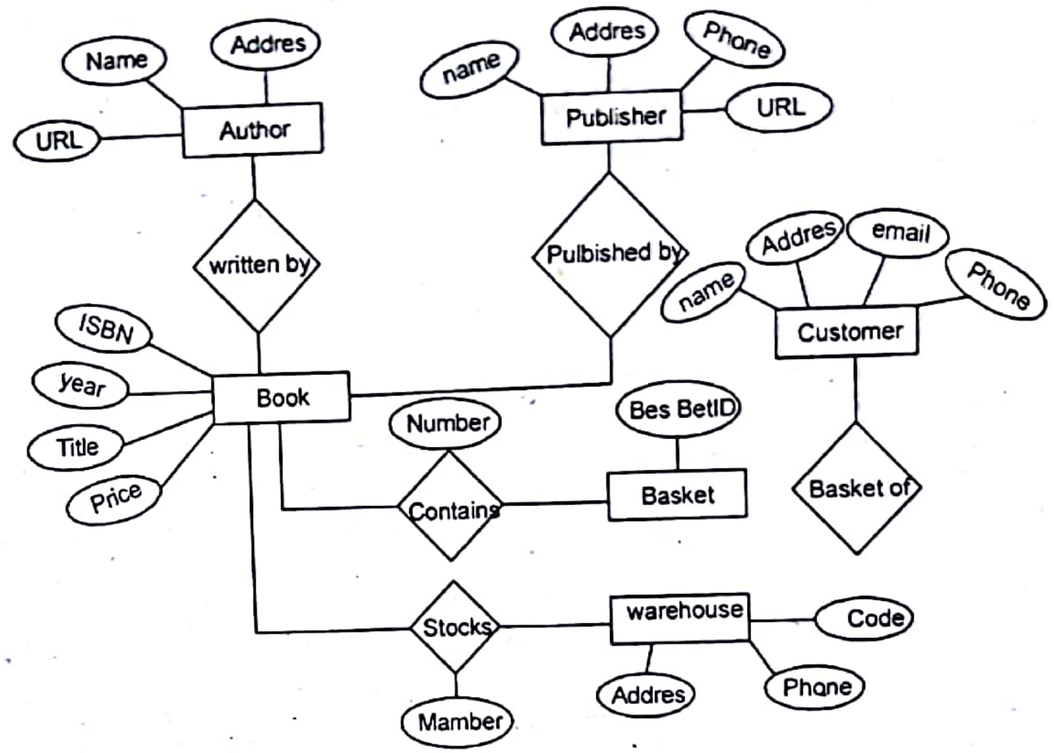
For example:



M : N Type Relationship

Q.3.(b) Draw an E–R diagram for a book store which maintains data of different books, customers suppliers and Publishers. Books are classified as scientific, engineering & medical. Customers can be students, faculty or Teaching Institutes. Based on category of customer appropriate discount is offered. Also, books are promoted for sole by supplier or individual publishers by contacting faculty or institutes. (Assume missing data if any). **[6.5]**

Ans.



E – R Diagram for a Book Store

Q.4.(a) Relation *R* (*A, B, C, D, E, F*) satisfies the following functional dependencies: $AB \rightarrow C, C \rightarrow A, BC \rightarrow D, ACD \rightarrow B, BE \rightarrow C, CE \rightarrow FA, CF \rightarrow BD, D \rightarrow EF$. Find out the closure of FD's. **(5)**

Ans.2.b. Refer Q. No.3; pg 32-2016

Q.4.(b) What is log based database recovery technique? Explain **[5]**

Ans. Refer Q.No.3.b. of Second Term 2013

Q.5.(a) Explain giving example the terms Integrity constraints and domain constraints. How are they enforced in oracle? **[6]**

Ans. Integrity Constraints: Integrity constraints are used to apply business rules for the database tables. The constraints available in SQL are Foreign key, Not NULL, Unique, Check.

Constraints can be defined in two ways :

1. The constraints can be specified immediately after the column defination. this is called **column - level defination.**

2. The constraints can be specified after all the columns are defined. This is called Table-evel defination.

(i) SQL Primary Key : This constraint defines a column or combination of columns which uniquely identifies each row in the table.

Scanned by CamScanner

Example : CREATE TABLE employee
(id number (S) PRIMARY KEY,
name char (10))

(ii) SQL Foreign Key: This constraint identifies any column referencing the PRIMARY KEY in another table. It establishes a relationship between two columns in the same table.

Example : CREATE TABLE PRODUCT
(product_id number(s) CONSTRAINT
pd_id_pk PRIMARY KEY,
product_name char(20));

Domain Constraints: The domain constraints integrity states that every elemant from a relation should respect the type and restrictions of its corresponding attribute. A type can have a variable length which needs to be respected, restrictions could be the range of values that the element can have, the default value if none is provided, and if the element can be NULL.

Oracle enforces data integrity and domain: Oracle enables you to define and enforce each type of data integrity rule. Most of these rules are easily defined using integrity constraints or database triggers.

Integrity constraint description: An integrity constraint is a declarative method of difining a rule for a column of a table. Oracle supports the following integrity constraints.

- NOT NULL constraints for the rules associated with nulls in a column.
- UNIQUE key constraints for the rule associated with unique column values..
- PRIMARY KEY constraints for the rule associated with primary identification values.
- CHECK constraints for complex integrity rules.

Domain database triggers : Oracle also lets you enforce integrity rules with a non-declarative approach using database triggers (stored database procedures automatically invoked on insert, update, or delete operations).

Q.1.(c) What are database triggers? How they are created? **(6.5)**
Ans. Refer Q. No.8.d. of End Term 2015

UNIT – II

Q.6.(a) Compute the closure of the following set f of functional dependencies for relation schema R = (A, B, C, D, E)

Fig. 1

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

List the candidate keys for R. **[8.5]**

Ans. Given: $A \rightarrow BC$, So $A \rightarrow B$ and $A \rightarrow C$

Since $A \rightarrow B$ and $B \rightarrow D$, So $A \rightarrow D$ (Transitivity rule)
Now, $A \rightarrow CD$ and $CD \rightarrow E$, $A \rightarrow E$
(Union, decomposition, transitivity rule)
Since $A \rightarrow A$, we have (reflexity Rule)
$A \rightarrow ABCDE$ (UNION Rule)
Since, $E \rightarrow A$, $E \rightarrow ABCDE$ (Transitivity Rule)
Since $CD \rightarrow E$, So, $CD \rightarrow ABCDE$ (Transitivity Rule)
Since, $B \rightarrow D$ and $BC \rightarrow CD$
So, $BC \rightarrow ABCDE$ (Aujumentation Rule & Transitivity)

Also, $C \rightarrow C$, $D \rightarrow D \rightarrow D$, $B\,D \rightarrow D$ and so on Therefore, we find that any FD with A, E, BC or CD on the L.H.S. of arrow is in F^*, in the FD. say,* represents any set of attributes in R, then F^* is

$$BD \rightarrow B$$
$$BD \rightarrow D$$
$$C \rightarrow C$$
$$D \rightarrow D$$
$$BD \rightarrow BD$$
$$B \rightarrow D$$
$$B \rightarrow B$$
$$B \rightarrow BD$$

and all FDs of the form

$$A^* \rightarrow \alpha$$
$$BC^{**} \rightarrow \alpha$$
$$CD^* \rightarrow \alpha$$
$$E^* \rightarrow \alpha$$

Where α is any subset of {A, B, C, D, E}.

The candidates keys are A, B, C, CD and E.

Q.6.(b) "Every schema that is in BCNF must also be in 3NF". Justify this statement. **[4]**

Ans. A relation in 3NF with a unique key is always in BCNF because BCNF deals with dependencies among the candiate keys. Since we have only one condiate key over here hence relation is also in BCNF.

Proof: Suppose that a relation $R(A_1, A_2 \ldots An)$ is in BCNF but it still has a transitive dependency $X \rightarrow Y$, $Y \rightarrow X$, $Y \rightarrow Ai$ for some $i(i = 1, 2, \ldots n)$, where X is a key and Ai is not in X since Ai is not a key attribute. Ai *is* not in F since otherwise the dependency $Y \rightarrow Ai$ would have been trival, contrary to the defination of transitive dependence. The existence of a non-trivial functional dependency $Y, \rightarrow Ai$ and the assumption that *R is* in *BCNF* imply that $Y \rightarrow Ai$ for ally = 1, 2,..., n. But then $Y \rightarrow X$ contray to the assumption about the existence of the transitive dependancy $X \rightarrow Y$, $Y \rightarrow X$ and $Y \rightarrow Ai$. This proves that a transitive dependense cannot exist in a relation which is in BCNF and thus such a relation is also in 3NF.

Q.7.(a) Write an algorithm for calculating canonical cover. Using the functional depedencies of Fig. 1, work out the steps to calculate canonical cover Fc. **[8.5]**

Scanned by CamScanner

Ans. Algorithm for computing the canonical Cover:

Step 1: For each FD $A \to B \in F$ perform a **Left Reduction:** Check for all $a \in A$ whether the attribute a is extraneous *i.e.*, whether $B \subseteq$ Attribute closure $(F, A - \{a\})$ holds. If this is the case, replace $A \to B$ by $(A - \{a\}) \to B$

Step 2: For each remaining FD $A \to B \in F$ perform the **right reduction:** check for all $b \in B$, whether the attribute b is extraneous *i.e.*, whether $b \in$ Attrclosure $(F - (A \to B) \cup (A \to (B - \{b\})))A)$ holds. If this is the case, replace $A \to B$ by $A \to (B - \{b\})$

Step 3: Remove the FDs of the form $A \to \phi$ which perhaps have been produced in the previous step.

Step 4: By using the union rule replace all FDs of the form $A \to B_1, \ldots A \to B_n$ by $A \to B_1 \cup \ldots \cup B_n$.

Example:

$$\left.\begin{array}{l} A \to BC \\ CD \to E \\ B \to D \\ E \to A \end{array}\right\}$$

Fig. 1.

Here, L.H.S (determinant) of all FDs is unique. Also, none of the attributes in the left or right side of any FDs is extraneous. Therefore, the canonical cover fc is equal to F.

Q.7.(*b*) List the physical storage. Media available on the computers you use routinely. Give the speed with which data can be accessed on each medium. **[4]**

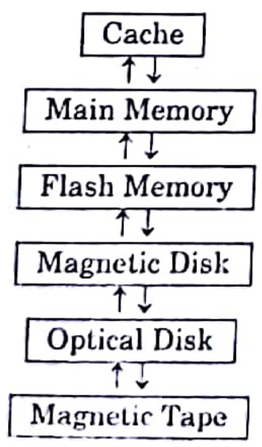
Ans. Several type of data storage exist in most computer systems. They very in speed of access, cost per unit of data & reliability.

- **Cache :** Most costly and fastest form of storage. Usually very small and managed by the operating system.
- **Main Memory :** The storage area for data available to be operated on.
- **Flash Memory :** (EEPROM)

Data in flash memory survive from power fax.

- **Magnetic - disk storage :** Primary medium for long-term storage.

– Typically the entire database is tored on disk.



- **Optical storage :** CD-ROM (Compact Disk Read only Memory), worm (Write-one read many) disk and Juke Box.

- **Tape storage :** Used primarily for backup archival data and protection from disk failures.

UNIT – IV

Q.8. Consider the following two transactions:

T_{31} : **read (A);**
read (B);
if A = 0 then B = B + 1;
write (B).

T_{32} : **read (B);**
read (B);
if B = 0 the A = A + 1;
write (A);
write (A).

Add lock and unlock instructions to transaction T_{31} and T_{32}, so that they observe the two-phase locking protocol. Can the execution of these transactions result in a deadlock? **[12.5]**

Ans. Adding lock and unlock instructions

T_{31} : lock – S(A);
read (A);
lock – X(B);
read (B);
if A = O
then B: = B + 1;
write (B);
Unlock (A);
Unlock (B);

T_{32} : Lock – S(B);
read (B);
Lock – X(A);
read (A);
if B = O
then A = A + 1;
write (A);
Unlock (B);
Unlock (A)

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

T_{31}	T_{32}
Lock – S(A);	
	Lock – S(B); read (B);
read (a); lock – X(B);	
	Lock – X(A);

Scanned by CamScanner

These transactions, T_{31} and T_{32} are now deadlocked.

Q.9.(a) How is database recovered after a crash using log file?

Ans. Refer Q.No.3.b. of Second Term 2013

Q.9.(b) What is time stamp Protocol and when it is used? [4]

Ans. This protocol works on a principle that "we must have a prior knowledge about the order in which the transactions will be accessed".

How to do it: To order the transactions, we associate a unique fixed timestamp, denoted by TS(Ti) for each transaction, Ti. This time stamp is an integer (unique) value. It is assigned by DBMS before the transaction, Ti starts execution.

1. **Uniqueness:** No equal timestamp value exist.
2. **Monotonicity:** Timestamp value always increase

Scheme Implementation Methods: There are two common methods for its implementation:

1. Using the value of system clock as the timestamp that is
   Transactions' Timestamp value = value of system clock
   when transactions enter in the system.
2. Using a logical counter that is incremented after a new timestamp has been assigned. That is,
   Transaction's timestamp value = value of the counter
   when the transaction enters the system.

Q.9.(c) Explain the cost based query optimization. [4.5]

Ans. Our objective is to choose the most efficient way of implementing the relational algebra operations at the lowest possible cost. *This method of optimizing the query by choosing a strategy that results in the minimum cost is called as cost – based Query Optimization.* The cost functions used in query optimization are just hte estimates. The cost of an algorithm depends on the cardinality of its input and also of its output because in the query tree, this forms the input to the next operation in the tree.

The cost of executing the query includes the following costs:

1. **Cost of access to secondary storage:** The cost of searching for reading and writing data blocks that reside on secondary storage like disk is called as an access cost.
2. **Cost of storage:** It is the cost of storing any intermediate relations that are generated by the execution strategy for the query.
3. **Cost of computation:** It is the cost of performing in – memory operations on hte data buffers during query execution.
4. **Cost of Memory Usage:** It is the cost pertaining to the no. of memory buffers needed during query execution.
5. **Cost of Communication:** It is the cost of transferring query and its results from the database site to site of query origination.

FIRST TERM EXAMINATION
(B.TECH)
DATABASE MANAGEMENT SYSTEM–2013

Time : 1 ½ hrs. **M.M. : 30**

Note: ***Q.No. 1 is Compulsory. Attempt any two more Question from remaining.***

Q.1(a) Explain the concept of age egation in E-R diagram . Give an example of where there concept is useful.

Ans.1.a. Refer Q.No.1.c. of First Term 2014

Q.1. (b) Why would you choose a database system instead of simply storing data in operating system files?

Ans. A database is an integrated collection of data, usually so large that it has to be stored on secondary storage devices such as disks or tapes. This data can be maintained as a collection of operating system files or stored in a database. The advantages of using a DBMS are:

- Data Independence and efficient access.
- Reduced application development time.
- Data Integrity and Security.
- Concurrent Access and Crash Recovery.
- Data Administration

Q.1. (c) List the advantages and disadvantages of two tier architecture of DBMS.

Ans. Advantages:

- Understandability
- Maintainability
- It will provide a user-friendly interface.

Disadvantages:

- Performance.
- It can support only a few users and allow data to be shared within a homogeneous environment.
- Scalability
- Poor Logic Sharing
- Remote usage

Q.1. (d) For the following expression in relational algebra, write an equivalent SQL query:

$\Pi_{A1, A2...An}(\sigma_p (r1 \times r2 > ....x.... rm))$

Ans. SQL Query:

Select A1, A2, ..., A_n
From r1, r2 ..., r_m
Where P;

Q.1. (e) List two reasons why null values might by introduced into the database. **(2 × 5 = 10)**

Ans. Nulls may be introduced into the database because the actual value is either unknown or does not exist. For example, (1) An employee whose address has changed and whose new address is not yet known should be retained with a null address.

Scanned by CamScanner

(2) If employee tuples have a composite attribute dependents and a particular employee has no dependents, then that tuple's dependents attribute should be given a null value.

Q.2. (a) Explain the following with reference to E-R diagram.

(i) Descriptive attributes.

(ii) Weak entity sets.

(iii) Total participation **(3)**

Ans. Descriptive Attributes: A descriptive attributes specifies a property of a dimension attribute, to which it is related by an one-to-one dimension. It is not used for aggregation; they are always leaves of their hierarchy and are graphically represented by horizontal lines.

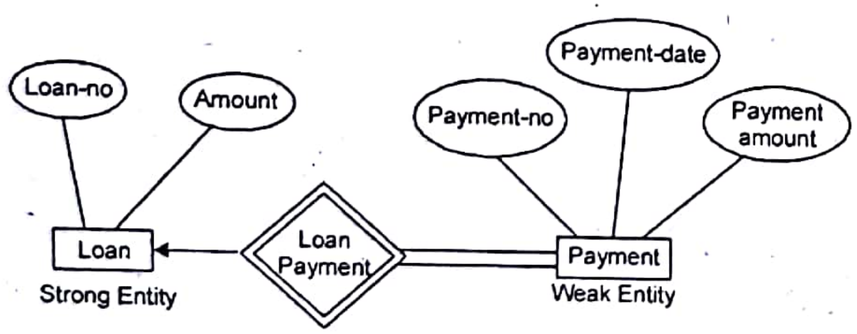
For example: Consider a birth date attribute. Many people could have the same birth date and so different people could have the same birth date, birth date would not be a key attribute for a person entity type, it would be a non-key attribute of person. Student entity type attributes such as: Stu_id, First Name, Last Name, Gender, Home No. ..... Most of these attributes will be descriptive.

(ii) Weak Entity Sets: (i) The entity set which does not have sufficient attributes to form a primary key.

(ii) Each weak set must be a part of 1 : M relationship set.

(iii) A member of weak entity set is called as a subordinate entity.

(iv) A weak entity set is represented by double outlined box and the corresponding identifying relation by a doubly outlined diamond.



(iii) Total Participation: Total participation constraints require the participation of every entity the relationship (displayed by double Line). It is also called **existence dependency.**

Q.2. (b) Construct an E-R diagram for a company having following description:

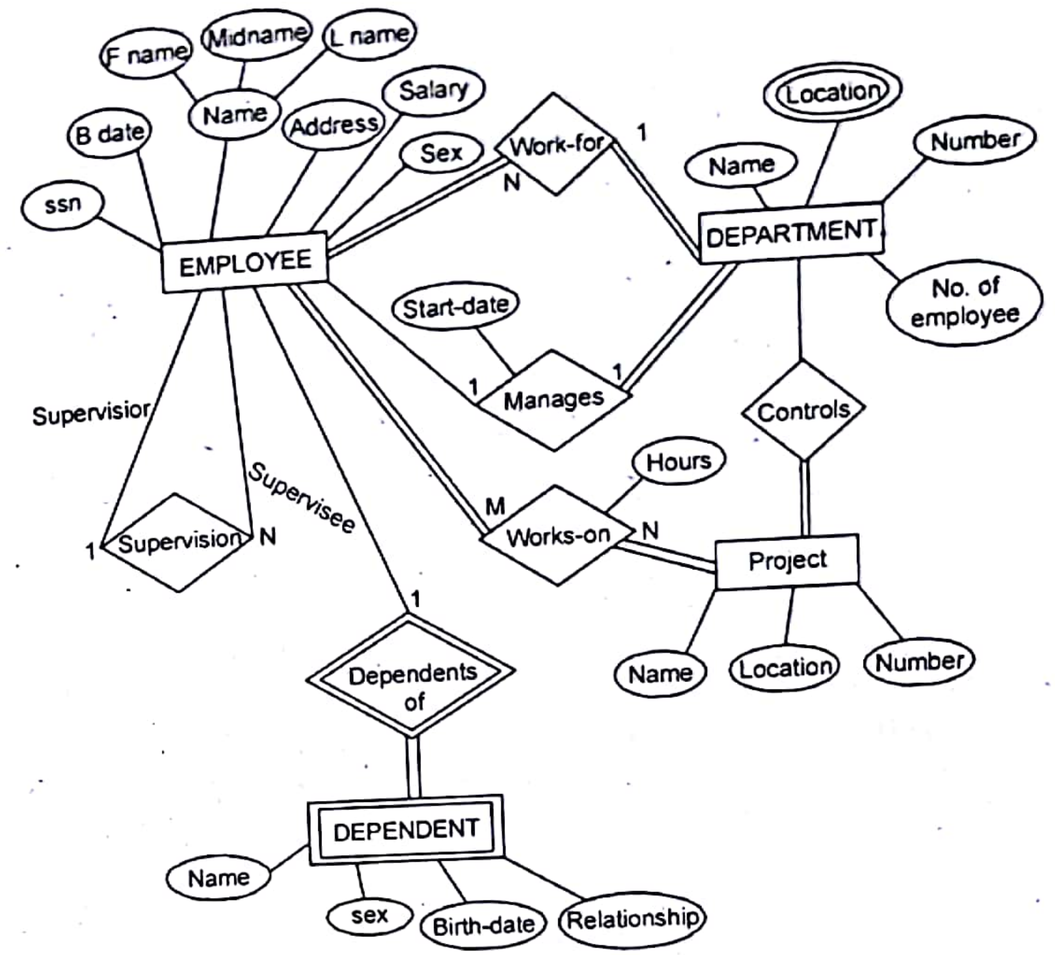
(i) There are several departments in a company and each department play have several locations.

(ii) A manager controls a particular department.

(iii) Each department is associated with number of projects.

(iv) An employee work in only one department but can work on several projects. **(4)**

Ans.



Q.2.(c) Define data abstraction. Explain the different levels of abstraction. **[3]**

Ans. Refer Q.No. 3 ; page 32-2016

Q.3.(a) Differentiate between natural join and outer join. Support your answer with an example. **[4]**

Ans. Refer Q.No.1.g. of End Term 2012

Q.3.(b) List two reasons why we may choose to define a view. **[2]**

Ans. Refer Q.No. 1(i) of End term 2015

Q.3.(c) Explain the following giving an example for each one of them

(i) Super Key

(ii) Primary Key

(iii) Foreign Key

(iv) Alternate Key

Ans.3(c) (i) and (ii) Refer Q.No. 3(a) of first term 2014

(iii) Foreign Key: It says that the column in the child table that references a primary key of the parent table is called as a Foreign key. They are used to link together two or more different tables which have some form of relationship with each other.

For Example: Consider the following parent Master table STD-ADD with the following attributes.

Scanned by CamScanner

STD-ADD (Roll-no, Name, Addr, Place, Pin)

Now consider another transaction/child table :STD-Marks with the following attributes:

STD-MARKS (Roll-no, Subject, Marks, Date)

Then, Roll-no column of STU-ADD table is the primary key whereas the Roll-No column of STD-Marks table is Foreign Key.

***(iv)* Alternate Key:** The alternate keys of any table are those candidate keys which are not currently selected as the primary key. So we can say that.

Alternate key (Ak) = Candidate key – Primary Key

$AK = CK - PK$

For Example: If Roll No. and Name combination is the candidate key, then if Roll-No is the primary key then Name becomes the alternate key.

Q.4. (*a*) Given the following relations: **(6)**

Vehicle (reg no, make, colour)

Person (eno, name, address)

Owner (eno, reg, no)

Write expressions in relational algebra or SQL to answer the following queries:

(*i*) List the names of persons who do not own any car.

(*ii*) List the names of persons who own only Maruti Cars.

Ans. (*i*) SLECT PERSON. name FROM
VEHICLE, PERSON, OWNER
WHERE VEHICLE. reg-no = OWNER.reg. no
AND PERSON. eno = OWNER.eno
AND VEHICLE.reg.no <> 'NOTHING'.

(*ii*) SELECT PERSON. name FROM
VEHICLE, PERSON, OWNER
WHERE VEHICLE. reg no = OWNER.reg. no
AND PERSON.eno = OWNER.eno
AND VEHICLE. reg-no = 'MARUTI';

Q.4. (*b*) Discuss division operation in relational algebra giving an example. **(4)**

Ans. Division: The division operator divides a dividend relation, A, of degree (m + n) by a divisior relation, B, of degree, n, and produces a resultant relation of degree, m.

For Example: Consider the following relation.

Relation–A

Sno	Pno
S1	P1
S1	P2
S2	P1
S2	P2

Relation B

Pno
P_1

Result

A Divide by B

Sno
S1
S2

SECOND TERM EXAMINATION
(B.TECH)
DATABASE MANAGEMENT SYSTEM–2013

Time : 1 ½ hrs. **M.M. : 30**

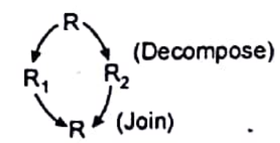
Note: ***Q.No. 1 is Compulsory. Attempt any two more Question from remaining.***

Q.1. (*a*) What is the lossless Decomposition? Explain with suitable example?

Ans. Lossless Decomposition: A relational table is decomposed or factored into two or more smaller tables in such a manner that it is possible to get back the original table by joining these decomposed tables. This is called as **Loss Less Decomposition.**

- Please note that all decompositions must be loss less, that is, there should not be any loss of information.
- The lossless join decomposition is a property of decomposition which ensures that no spurious tuples are generated when a natural join operation is applied to the relations in the decomposition.

For Example: The decomposition of R(A, B, C) into R, (A, B) and R_2(A, C) is loss less if for attributes A (common to R_1 and R_2) either $X \rightarrow Y$ or $Y \rightarrow Z$.



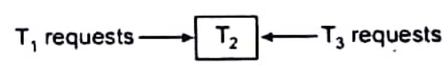
Q.1.(b) Describe briefly any two desirable properties of a transaction?

Ans. Refer Q.No.4.b. Second Term 2016

Q.1. (*c*) What is the difference between What die and Wound wait scheme?

Ans. Wait Die Scheme: If Ti has a higher priority, it is allowed to wait else it is aborted. This means that when Ti request for a data-item that is currently being held by T_j, then T_i is allowed to wait only if it has a timestamp smaller than thatof T_j is older than T_j otherwise, T_i is rolled back (dies).

For Example: Say, T_1 requests for a data item held by T_2 and T_3 also requests for data item held by T_2 i.e.



Wound-Wait Method: If Ti transaction has a higher priority then we abort Tj else Ti wais. It means that when Ti requests a data-item currently held by Tj, then Ti is allowed to wait only if it has a Ts (Ti) > Ts (Tj). Else Tj is rolled back.

For Example: If T_1 requests a data-item held by T_2 then the data-item will be presnpted from T2 and T2 will be rolled back i.e.,

T1 → T2 (Rollbacks)

Now, if T3 requests a data-item held by T2, then T3 waits

T2 ← T3 (waits)

Wait-die Scheme	Wound-Wait Scheme
(*i*) This scheme is Non-Preemptive	(*i*) This scheme is Preemptive.
(*ii*) Lower priority transactions can never wait for higher transactions.	(*ii*) The higher priority transaction never wait for lower priority transaction.

Scanned by CamScanner

Q.1. (*d*) What are the different types of schedules explain with suitable example? **(2)**

Ans. Complete Schedule: A schedule that contains either an abort or a commit for each, transaction whose actions are listed in it is called as a complete schedule.

(*ii*) Serial Schedule: Each serial schedule consists of a sequence of instructions from various transactions, where the instructions belonging to one single transaction appear together in that schedule.

$T1$	$T2$
$A = A \times 10$ $B = B - 10;$	
	$A = A/100;$ $B = B \times 4$

Schedule 1: Serial Schedule

(*iii*) Non-Serial Schedule: A schedule in which the operations from a set of concurrent transactions are interleaved is called as non-serial schedule.

T_1	T_2
$A = A \times 10$	
	$A = A/100;$
$B = B - 10$	
	$B = B \times 4$

Schedule 2: Non-Serial Schedule

(*iv*) Equivalent Schedule: Two schedules are said to be equivalent schedules if they produce the same result on any database state.

For Example: The Schedule 2 (given above) is an equivalent schedule to schedule 1.

(*v*) Serializable Schedule: A non-serial schedule that is equivalent to some serial execution of transactions is known as serializable schedle.

Q.1. (*e*) Explain the difference between 3NF and BCNF?

Ans.

3NF	BCNF
1. A relation R is said to be in 3NF iff the following conditions are simulationeously satisfied: (*i*) R is already in 2NF. (*ii*) No non-prime attribute is transiviely dependent on the key.	1. A relation R is said to be in BCNF iff every determinant is a candidate key. Here, determinant is a simple. attribute or composite attribute on which some other attribute is fully functionally dependent.
2. Suppose a FD, A→B, 3NF allows this in a relation if B is a PK and A is not a candidate key.	2. BCNF insists that A→B dependency to remain in a relation, A must be a candidate key.

Q.2. (*a*) Compute the closure of the following set F of functional dependencies for relation schema R = (A, B, C, D, E): **(5)**

A->BC

CD->E

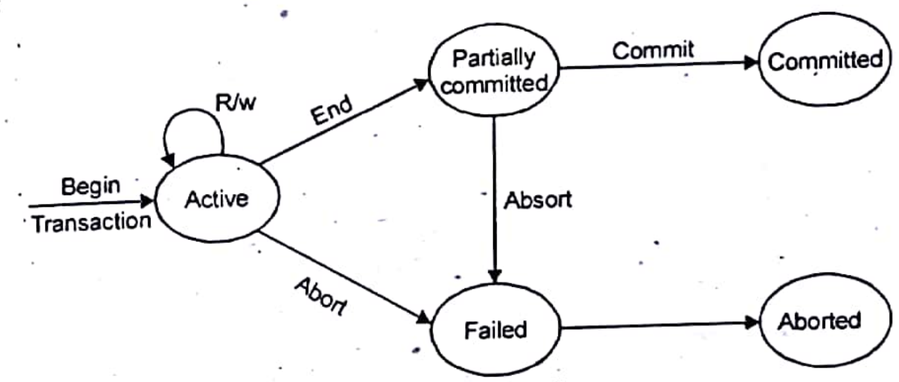
B->D

E->A

Ans. Refer to End Term Examination, Dec-2012 in Q.No. 6. (*a*).

Q.2. (*b*) Draw the State Transition Diagram of Transaction. Explain each of them? **(5)**

Ans. States of a Transaction: A transaction can be in various states as shown below:



Transaction State diagram

Transaction state diagram

(*i*) **Activie state:** It is the initial state. The transaction stays in this state while it is executing.

(*ii*) **Partially committed:** When the final statement has been executed, we say transaction is in a partially committed state.

(*iii*) **Failed:** When the normal execution can no longer proceed, we say that the transaction is in a failed state.

(*iv*) **Aborted:** A transaction is aborted if the transaction has been rolled back and the database has been restored to its state prior to the start of the transaction.

(*v*) **Committed:** When a transaction has successfully been completed.

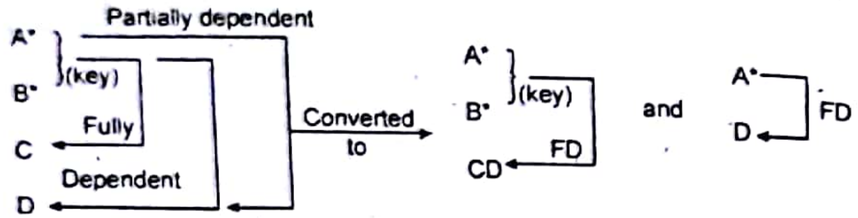
Explanation: A transaction starts in the **Active State.** When it finishes its last statement, it enters the **partially committed state.** At this time, the transaction has completed its execution but it is still possible that it may have to be aborted as the actual output may still be temporarily residing in the main memory. DBMS then writes infomation to disk that even if the system failure occurs, the updates performed by Ti can be recreated when the final or last of this information is written out, the transaction enters the committed state.

Q.3. (*a*) Explain the 2NF and 3NF normal form with suitable example? (5)

Ans. 2NF: "A relation R is in second Normal Form (2NF), if and only if it is in 1NF and every non-key attribute is fully dependent on the primary key".

If A, B, C and D are four fields or attributes and if A and B are Pks. Now A and B attributes determine all other attributes. C and D are non-key (or non-prime) attributes. Attribute C is fully dependent on the key. Attribute D is partially dependent on the key, because we need only attribute A to functionally determine it. Here, the rule is

Scanned by CamScanner



These resultant two-relations are both fully dependent on the PK now. So,

$$(A^*, B^*) \rightarrow C$$
$$A^* \rightarrow D$$

Here, '*' denotes that it is a PK. So, we observe that the first new relation has 3 attributes. A*, B* and C where (A*, B*) is the PK of this relation and the second relation has A* and D as its 2 attributes. So, A* is the PK of second relation and D is now fully dependent on the key.

3NF: A relation R is said to be in 3NF iff the following conditions are simultaneously satisfied:

1. R is already in 2NF.
2. "No non-prime attribute is transitively dependent on the key or we can say that no non-prime attribute functionally determines any other non-prime attribute."

So, we observe that the objective of transforming relations into 3NF is to remove transitive dependencies.

For Example: Consider a relation (R) that has a set of these attributes –A, B and C. Also, assume that the following FDs for this R, exists.

A→B, B→C and thus A→C.

Note that C→B is neither required nor prohibited. And if all these conditions are true, we say that the attribute C is TD on attribute A. These FDs determine the conditions for having a TD of attribute – C on A. This is shown in Fig.

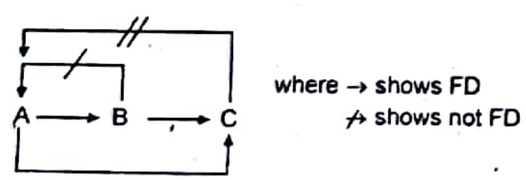


Fig.

i.e.,

$$A \rightarrow B$$
$$B \rightarrow C$$
$$\boxed{A \rightarrow C}$$

Q.3. (b) What is log based recovery? How it is different from shadow paging Scheme? **(5)**

Ans. Log Based Recovery : It is the most common structure used for recording any modification made to the database. In this recovery procedure, a Log-life. A log record maintains a record of all operations of the database. There are four types of log record, namely

(*i*) < Start > Log Record.
(*ii*) < Update > Log Record.
(*iii*) < Commit > Log Record.
(*iv*) < Abort > Log Record.

(*i*) < Start > Log Record : It contains information about the start of each transaction. It has a transaction identifier, which is always unique. It's syntax is

< Ti, Start >

(*ii*) < Update > Log Record : It describes a single database write.

It's synatx is

< Ti, Xj, V_1, V_2 >

Ti – Transaction indentifier

Xj – Data Item

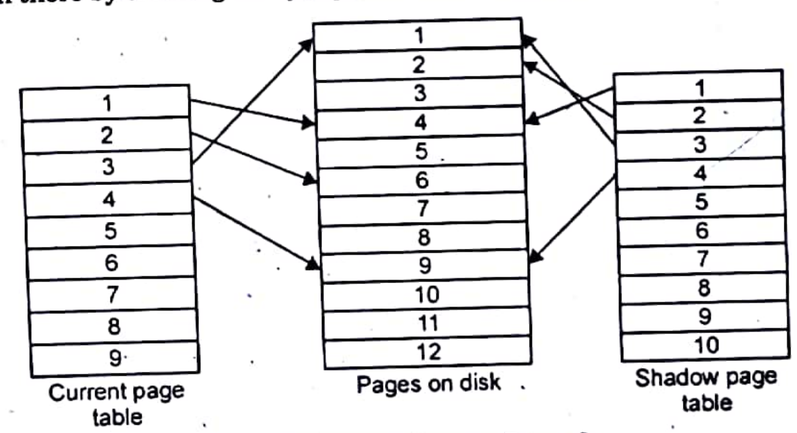
V_1 – Old data-item value

V_2 – New data-item value

(*iii*) < Commit > Log Record : When a transaction, Ti is sucessfully committed or completed, a < Ti, Commit > Log record is stored in the log file.

(*iv*) < Abort > Log Record : When a transaction Tj is aborted due to any reason, a < Ti, about > log record is stored in the log file.

Log based recovery is different from shadow paging. If we compose shadow paging with log-based recovery we find that in log–based method the system writes the data item in the same location (over writing). So, only a sinlge copy of each item is maintained on disk. On the other hand, in shadowing, the system writes a new data item at a different disk loction there by creating many copies of the data item. This is shown :



Writ operation on Page–2

Use of shadow table when system crashes : One simple method is to choose one fixed location in stable storage that contains the disk address of the shadow page table. After a crash, we copy the shadow page table into the main memory and then use it for subsequent transaction processing. The shadow page table will point to be database pages corresponding to the state of the database prior to any transaction that was active at the time of the crash. This is the reason as to why aborts are automatic.

Q.4. (a) What is time stamp? Discuss the working of time stamp ordering protocol with suitable example? **(5)**

Ans. Timpestamp: This protocol works on a principle that –"we must have a prior knowledge about the order in which the transactions will be accessed". In graph based protocol, we require the knowledge of ordering of data items in the transactions.

How to do it?

To order the transactions, we assosiate a unique fixed timestamp, denoted by TS(Ti) for each transaction, Ti. This time stamp is an integer (unique) value. It is assigned by DBMS before the transaction Ti, starts execution.

Scanned by CamScanner

Timestampes have 2 properties:

1. Uniqueness: This property assures that no equal timestamp values can exist.

2. Monotonicity: It means that timestamp values always increase.

Timestamp–Ordering Protocol: It ensures that any conflicting read and write operations are executed in timestamp order. This protocol operates as follows.

1. Suppose that transaction Ti issue read (Q): (a) If TS (Ti) < w-timestamp (Q), then Ti needs to read a value of Q that was already overwritten. Hence the read operation is rejected. and Ti is rolled back.

(*b*) If TS (Ti) >= w-timestamp (Q), then the read operation is executed, and R-timestamp (Q) is set to the maximum of R-timestamp (Q) i.e. TS (Ti).

2. Suppose that transaction Ti issues write (Q):

(a) If Ts (Ti) < R-timestamp (Q), then the value of Q that Ti is producing was needed previously and the system assumed that value would never be produced. Hence, the write operation is rejected, and Ti is rolled back.

(*b*) If TS (Ti) < w-timestamp (Q), then Ti is attempting to write an absolute value of Q, Hence the write operation is rejected and Ti is rolled back.

Q.4. (*b*) What is deadlock? GIve a mechanism to detect deadlock? (5)

Ans. Deadlock: Deadlock are preferable to there inconsistent states because they can be handled by rolling back of transactions. The inconsistent states may lead to real world problems that cannot be handled by DBMS.

The rule is –"When a transaction requests a Lock on a data item in a particular mode and no other transaction has a lock on the same data item in a conflicting mode, then the lock can be granted."

Deadlock Detection: It is defined as a periodic check mechanism by DBMS to determine whether a deadlock has occured or not. Actually, the frequency of deadlocks depends upon the query load and the physical organization of database.

In 1981, Gray proposed that the deadlock frequency (deadlock per second) is:

(*a*) Square of he degree of multi-programming (DOM).

(*b*) Fourth power of the transaction size. For this a system must maintain information about the current allocation of data items to transactions provide on appropriate algorithm to find if the system has enered a deadlock state and to recover from deadlock if it exists.

Wait-For Graph (WFG) for Deadlock Detection: A very simple way to detect a state of deadlock is to draw wait-for graph. This graph is constructed and maintained by the system.

Notations Used:

1. One node is created in the wait-for graph for each transaction that is currently executing.

2. Whenever a transaction, Ti, is waiting to Lock an item—A that is currently locked by a transaction, Tj a directed edge (Ti → Tj) is created in the wait-for graph. When Tj releases the lock on the items hat Ti is waiting for, the directed edge is dropped from the wait for graph.

END TERM EXAMINATION
(B.TECH)
DATABASE MANAGEMENT SYSTEM–2013

Time : 3 hrs. **M.M. : 75**

Note: ***Attempt any five questions including Q.no. 1 which is compulsory. Select one question from each unit.***

Write short answers of any five of the following:

Q.1. (*a*) Represent the natural join operation of the relational algebra as a combination of Cartesian product, selection and projection operation.

Ans. $\Pi_{<Att.\ Name}(\sigma_{<Att.\ Name>}(A \times B))$

(OR)

$A \bowtie B$

It will performs the joining of two tables A and B.

Q.1. (*b*) What do you mean by instance and schema? Explain the different between these.

Ans. Schemas: The overall description of the database is called as the database schema. Actually, DBMS has a three level architecture and accordingly these are three different types of schemes in the database. A schema is defined as an outline or a plan that describes the records existing at a particular level. The schema diagram is used to show he database schema.

STU-ADD	Roll.No.	Name	Address	Place

Schema diagram

Instance: Databases change over time when you insert or delete the information. The collection of information stored in the database at a particular moment is called as an instance.

Roll-No	Name	Addr	Place
10	Shilpa	A-2/7, Saket	Delhi
20	Rajiv	B-5, Noida	Delhi
30	Amit	G-8, Janakpuri	Delhi

Instance of STU-ADD table

The **schema** is sometimes called as the intension of the database while an instance is called as an extension of the database.

Q.1. (*c*) Explain Authorization in SQL with example.

Ans. Authorization includes primarily two processes:

- Permitting only certain users to access, process or deterdata.
- Applying varying Limitations on users access or actions. The limitations placed on users can apply to objects, such as schemes, tables or raws, or to resources, such as time (CPU, connect or idle times).

Authorization in SQL:

Privileges

Scanned by CamScanner

Roles

Profiles

Resource Limitations

Who Can grant Schema Object Privileges: A user automatically has all object privileges for schema objects contained in his or her schema. A user can grant any object privilege on any schema object he or she owns to any other user or rote.

For Example: Assume user SCOTT has a table named t2:

SQL > GRANT grant any object privilege to U1:

SQL > Connect u1/u1.

Connected.

SQL > GRANT select on scott. t2/To U2.

SQL > SELECT GRANTEE, OWNER, GRANTOR, PRIVILEGE, GRANTABLE FROM DBA_TAB-PRIVS WHERE TABLE-NAME = employees';

Q.1. (*d*) Explain Candidate Key and Alternate Key.

Ans. Alternate Key : Refer Q.No.3.c (iv) of First term 2013
Candidate Key : Refer Q.No.3.a. First Term 2014

Q.1. (*e*) List two reasons why you may choose to define system.

Ans. Refer to First Term Examination, Sept-2012 in Q.No. 1 (a).

Q.1. (*f*) Under what circumstances multivalued dependency is known as functional dependency?

Ans. In database, theory multivalued depdency is a fully constraint between two sets of attributes in a relation.

In contrast to he functional dependency, the mulivalued dependency requires that certain tuples be present in a relation. Therefore, a multivalued dependency is a special case of tuple-generating dependency. The multivalued dependency plays a role in the 4NF database normalization.

A multivalued dependency is a special case of a join dependency, wih only two sets of values involved, i.e. it is a 2-ary join dependency.

Intersting Properties Related to FD:

- If $\alpha \rightarrow\rightarrow \beta$ then $\alpha \rightarrow\rightarrow R-\beta$
- If $\alpha \rightarrow\rightarrow \beta$ and $\gamma \subseteq \delta$ then $\alpha\delta \rightarrow\rightarrow \beta\nu$
- If $\alpha \rightarrow\rightarrow \beta$ and $\beta \rightarrow\rightarrow \nu$ then $\alpha \rightarrow\rightarrow \nu - \beta$

The following also involve FDs.

- If $\alpha \rightarrow \beta$ then $\alpha \rightarrow\rightarrow \beta$
- If $\alpha\rightarrow\rightarrow\beta$ and $\beta\rightarrow\nu$ then $\alpha\rightarrow\nu - \beta$

Q.1. (*g*) What is the need of the normalization?

Ans. Need of Normalization:

(*i*) They should have a minimal number of attributes necessary to suport the data requirements of the enterprise.

(*ii*) The attributes with a close logical relationship are found in the same relation.

(*iii*) It has a minimal redundancy, which requires that the values of no attribute of a database instance is replicated except where tuples are linked by foreign keys.

Q.1. (*h*) Differentiate DML and DDL.

Ans. DD: "Data Defination Language".

DDL is used to create and remote database objects.

For Example:

- Create
- ALTER
- DROP

DML: "Data Manipulation Language".

DML is used to manipulate data in the database. Some commands are:

- INSERT
- DELETE
- UPDATE
- SELECT

UNIT–I

Q.2. (*a*) What are the problems with traditional file processing system?

Ans. Problems:

(*i*) **Data Redundancy:** Since these system used a decentralized approach, so each department used its own application programs and files. This resulted in data redundancy or duplicacy of information in several files.

(*ii*) **Poor Data Control:** Since these systems are decentralized in nature, so these was no centralized control on the fields. This made field-names ambiguous-having many meanings. This result in poor control.

(*iii*) **Poor Data Manipulation Capabilities:** Since these systems do not provide strong connections between data in different files so it's data manipulation capability is low.

(*iv*) **Data Dependence:** In these systems, files and records were described by specific physical formals that were coded into the application program.s

(*v*) **Incompatible file formats:** As we know that the structure of files is embedded in the application programs, so the structures are dependent on the application programming language.

(*vi*) **Security Problems:** Every user of the database system shiould not be allowed o access all data. Since application programs are added to the file-oriented systems in adhoc manner, it was difficult to enforce such security system.

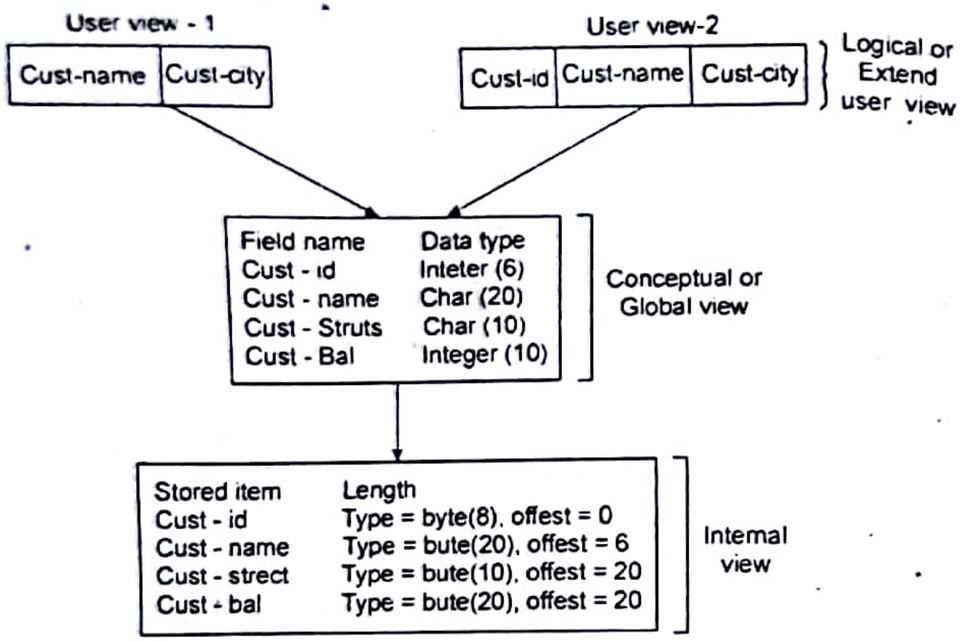
Q.2. (*b*) Explain with suiable example the physical, logical and view level of database.

Ans. External Level or View Level: It is the user's of the database. It is the highest level of obstraction. It deals with the way in which data is viewed by the individual users.

Conceptual or Logical Level: It is the global view of the database. This level discribes what data is stored in the database and the relationships among the data. It is the middle level in this 3-tier architectiure. This level contains the logical structure of the existise database as seen by the DBA.

Internal Level or Storage Level: It is the physical representation of the database on the computer. This level describes how the data is stored in the database. It concerns with the way the data are physically stored on the hardware.

Scanned by CamScanner



Note the following from the above diagram.

(a) External View: This view depend upon the who is accessing the database.

(b) Conceptual View: This view contains the logical view of the whole database. It represents the data ype of each required field.

(c) Internal View: This view represents physical location of each element on the disk of the server as will as how many bytes of storage each element needs.

OR

Q.3. Draw the E-R diagram for the following problem definition:

The book club has members. The book club sells books to its members. The members place orders for books, which the book club fulfils. Each order contains one or more than one books. The books are written by author(s). The publisher publishes are book. An author can write more than one book and a book can have more than one author. A book is published by a publisher, but a publisher publishes many books. A member can place more than one order. The member also can choose not to place an order. The book club sells many books.

Ans. *Refer to End Term Examination , Dec–2012 in Q.No. (3) (b).*

UNIT–II

Q.4. Consider the following database schema:

STUDENT (name, sroll_number, class, major)

COURSE (course_number, course_name, credit_hour, department)

SECTION (Sectionid, course_number, semester, year, instructor)

GRADE REPORT (sroll_number, sectioned, grade)

PREEQUISITE (course_number, prerequisite_no)

Specify the following queries in SQL for above schema:

(a) Retrieve the name of students majoring in ECS (elecronics)

Ans. SLEECT STUDENT. name
FROM STUDENT, GRADE REPORT, SELECTION COURSE
WHERE STUDENT. sroll nmber = GRADE REPORT.
sroll_number AND GRADE REPORT, sectioned=SECTION.sectioned AND
COURSE. Course_number = SECTION. Course_number
AND Course. course_name = 'ECS';

(b) Name of all courses taught by professor Sharma in the year 2005 and 2006.

Ans. SELECT Course. Course-name FROM
COURSE, SECTION WHERE
COURSE. course-number = SECTION. course-number
AND SECTION. Instructor = 'PROFESSOR-SHARMA'
AND SECTION.Year = '2005' AND SECTION. YEAR = '2006'.

(c) For each section taught by professor Sharma, find the course number, semester, year and number of students who took the section.

Ans. R: SELECT SECTION. course-number, SECTION. semester, SECTION. year FROM SECTION, GRADE_REPORT, STUDENT WHERE SECTION. Instructor = 'PROFESSOR SHARMA' AND STUDENT.sroll_number = GRADE-REPORT.small number AND SECTION sectionid = GRADE-REPORT. sectiond; SELECT COUNT (sroll-number) FROM R;

(d) Retrieve the name and transcrip of each senior student (class=4). Transcript conains course number, course name, credit hourse, grade, year for each course completed.

Ans. SELECT COURSE. course-number, COURSE.course-name, COURSE. credit-hour, SECTION. year, GRADE REPORT. goode FROM SECTION, COURSE, GRADE REPORT, STUDENT WHERE SECTION sectionid = GRADE REPORT. sectionid AND SECTION. Course-number = COURSE.course-number AND STUDENT. sroll-number = GRADE REPORT. sroll-number AND STUDENT. Class = '4'.

(e) Retrieve the names and major department of all A grade students i.e students who have grade A in all courses taken by them.

Ans. SELECT STUDENT . name, STUDENT. major FROM SECTION STUDENT, COURSE, GRADE REPORT WHERE

STUDENT. sroll-number = GRADE REPORT. sroll-number AND SECTION. course-number = COURSE. course-number and GRADE REPORT. Grade = 'A' AND SECTION. sectionid = GRADE REPORT. sectiond;

OR

Q.5. Using the above database schema explains the following relational algebra operations by writing suitable queries:

(a) Set difference

Ans. The difference of relations R and S, defined as

$R - S = \{(x_1, \ldots, x_n) \in R : (x_1, \ldots x_n) \notin S\}$

It is the set of types which are in R but not in S. It is required that R and S have the same degree.

Scanned by CamScanner

(b) Set interaction

Ans. It can be defined as:

$$R \cap S = R - (R - S)$$

Thus its functionality is provided by difference, that is why it is not considered to be recognised as a basic operation.

(c) Create view

Ans. $\Pi_{emp\text{-}no}$, e-name, d-name (EMP ⋈ DEPT)

(d) Natural join

Ans. Natural Jain: It is a special case of the join. It is applicable only when both R and S have colums that are named by attributes. Assuming that $<A_1, A_2 ... A_k>$ are all the attributes names used for both R and S, then the natural Join is defined as:

$R \times S = \Pi_{i1, i2 ... im}\, \sigma_{R.A1 = S.A_1 \wedge ... \wedge R.A_k = S.A_k}(R \times S)$ such that $i_1 ... i_m$ is the list of all components of $R \times S$, in order, except the components $S.A_1 .... S.A_k$.

(e) Union: The UNION of relations R and S is defined as

$RUS = \{<x_1 ... x_n> : <x_1 .... x_n> \varepsilon R \; V <x_1 .. x_n> \varepsilon S\}$

It is the set of types that are in R or S. This operation may be applied to the relations of the same degree only, so all the tuples in the result have the same number of components (values).

UNIT-III

Q.6(a) What do you mean by decomposition? Explain the desirable properties of decomposition ?

Ans. NOT IN NEW SYLLABUS

Q.6. (b) Discuss of role of functional dependency (FD) in Normalization. Provide an example to understand the FD.

Ans. Yes, we can infer a functional dependancy from a given relational state.

"An attribute in a relational model is said to be functionally dependent on another attribute in the table if it can take only one value for a given value of the attribute upon which it is functionally dependent."

Functional Dependancy (FD): A functional dependency is an association between two attributes of the same table (Relation). One of the attributes is called as the **determinant** and other attributed is called as the **determined**. Note here that for each value of the determinant there is associated one and only one value of the determined.

For example:

Supplier Table (SP)

S.No.	S name	Status	City
S_1	Ajay	11	Rohtak
S_2	Mayur	12	Delhi
S_3	Rahul	13	Rohtak

Here, sname is FD on S_{no}. why? This is because Sname can take only one value for the given vlaue of S_{no} (e.g. S_2 for Mayur). We can also say that there must be one Sname for supplier number, S_2.

∴ The FD is represented as:

$$\boxed{S_{no} \rightarrow Sname}$$

Here, FD is shown by an arrow (→) which shows that Sname is functionally dependent on Sno.

Similarly, city and status are also *FD* on *Sno*, because for each value of S_{no}, there will be only one city and status. These FDs are represented as follows:

$$\boxed{\begin{array}{l} Sno \rightarrow City \\ Sno \rightarrow Status \end{array}}$$

We can combine these FDs also and write them as follows:

$$\boxed{SP.Sno \rightarrow SP(Sname, city, status)}$$

Q.7. (a) Explain the concepts of Object Oriented OBMS.

Ans. Object Oriented Database: Software engineering models the dynamic aspects of software whereas the database technology model the static aspects of information storage. OODB are extensible meaning that they are capable of defining new data types as well as operations on them.

Advantages of Object Oriented Databases: There are number of advantages such as:

- Enriched modefing capabilities
- More expressive query language
- Support for schema evolution.
- Support for long-duration transaction.
- Improved performance.
- Improved Security.
- Applicability to advanced database applications.

Q.7.(b) What mechanism are used for (i) Database Protection (ii) Accessing shared database ?

Ans. NOT IN NEW SYLLABUS

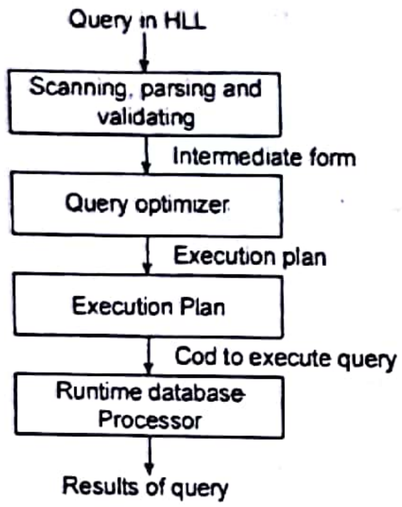
UNIT-IV

Q.8. (a) What is meant by the term Query Processing? What are the various steps involved in this process?

Ans. The procedure of selecting the best plan or strategy to be used in responding to a database request and executing the plain to generate the response is known as query processing. It is a stepwise process. The SQL query is first of all scanned, parsed and validated. Let us draw a flow chort now.

The scanner identifies the language takens such as SQL key words, attribute names and relation names in the text of query. The parser cheks the query syntax to determine whether it is formulated according to the syntax rules of the query language. The query must also be validated by checking that all attributes and relation names are valid. An internal representation of the query is then created usually as a tree data structure called as a query-tree.

Scanned by CamScanner



Flowchoots for query processing

Q.8. (*b*) Discuss the advantages of distributed system.

Ans. Advantages:

- Better communication services to provide access to remote sites and to transfer queries and data among multiple sites.
- Better distributed system catalogue management.
- Better distributed query processing.
- Better security control to maintain access privileges.
- Better distributed transaction management.
- Better concurrency control for data consistency.
- Better recovery services during system failures.
- Support for gateways.
- Support for fragmentation independence.
- It should be hardware and OS indepdent.

OR

Q.9. (*a*) Explain the process of Deadlock-Detection and Recovery.

Ans. It is defined as a periodic check mechanism by DBMS to determine whether a deadlock has occured or not. Actually, the frequency of deadlocks depends upon the query load and the physical organisation of a database.

Wait for graph for deadlock Detection: A very simple way to detect a state of deadlock is to draw wait-for-graph. This graph is constructed and maintained by the system.

Notations Used:

1. One node is created in the wait for Graph for each transaction that is currently executing.

2. Whenever a transaction Ti, is waiting to lock an item-A that is currently locked by a transaction, T_j, a directed edge ($T_i \rightarrow T_j$) is created in the wait-for graph. When T_j releases the lock on the items that T_i is waiting for, the directed edge is dropped from the wait-for graph.

Deadlock Recovery: Whenever our detection algorithm finds that the deadlock exists then we must recover from it. The common solution to this is to roll-back one or more transactions to break the deadlock. This processes of choosing which transactions to abort is known as victin selection. We should rollabck those transactions only that involve minimum cost. For this, the criteria used as follows:

1. Abort the transaction that has fewest locks.
2. Abort the transaction that has done the least work.
3. Abort the transaction that is farthest from completion.

Once we have decided that a particular transaction must be rolled back, we must find out that how far it should be rolled back.

Q.9. (*b*) Why a concurrency control needed? Explain lost update, inconsistent retrievals and uncommitted dependency anomalies.

Ans. Concurrent execution of transaction means executing more than one transaction at the same time. We use the technique of interleaving execution of two transactions.

(*i*) Lost update Problem: "A lost update problem occurs when two transactions that access the same database items have their operations in a way that makes the value of some database item incorrect. In other words, if the transactions, T_1 and T_2 both read a record and then update it, the effects of the first update will be overwritten by the second update."

For example:

Transection *A*	Time	Transection *B*
=		=
FETCH *R*	t_1	=
=		=
=	t_2	FETCH *R*
=		=
UPDATE *R*	t_3	=
=		=
=	t_4	UPDATE *R*

Update at time t_3 lost at t_4

To understand the above situation, assume that there is a Record *R*, with a field, say Amt having value 1000 before time t_1.

- Both transactions *A* & *B* fetch this value at t_1 and t_2 respectively.
- Transaction *A* updates the Amt field in *R* to 800 at time t_3.
- Transaction *B* updates the Amt field in *R* to 1200 at time t_4.

Thus after time t_4, the Amt value in record *R* has value 1200. Update by transaction *A* at time t_3 is over-written by the transection *B* at time t_4.

(*ii*) **Unrepeatable Read (or Inconsistent Retrievals):** This problem occurs when a transaction reads several values from a database while a second transaction update some of them. Actually, this problem occurs when a transaction calculates some aggregate function over a set of data while other transactions are updating the data. In an unrepeatable read, the transactions, T_1 reads a record and then does some other processing during which the transaction T_2 updates the record. Now, if T_1 rereads the record, the new value will be inconsistent with the previous value."

For example:

Scanned by CamScanner

T_1	T_2
—	Read X
Read Y	$X + 1 = X$
—	Write X
Read X	—
$X + Y = X$	—
—	Roll back.
Write X	—

This inconsistency occurs because T_1 reads several values from a database while a second trnasaction. T_2 updates some of them.

FIRST TERM EXAMINATION
FOURTH SEMESTER [2014]
DATABASE MANAGEMENT SYSTEM [ETCS-208]

Time : 1.30 hrs. **M.M. : 30**

Note: Note: *Attempt Q.No. 1 which is compulsory and any two more questions from remaining.*

Q.1.(a) Define the terms: **(5×2=10)**

(i) Data Independence (ii) Views

Ans. (i) Data independence is the type of data transparency that matters for a centralized DBMS. It refers to the immunity of user applications to changes made in the definition and organization of data.

Physical data independence deals with hiding the details of the storage structure from user applications. The application should not be involved with these issues, since there is no difference in the operation carried out against the data.

The ability to modify schema definition in one level without affecting schema definition in the next higher level is called data independence. There are two levels of data independence, they are Physical data independence and Logical data independence.

1. Physical data independence is the ability to modify the physical schema without causing application programs to be rewritten. Modifications at the physical level are occasionally necessary to improve performance. It means we change the physical storage/ level without affecting the conceptual or external view of the data. The new changes are absorbed by mapping techniques.

2. Logical data independence is the ability to modify the logical schema without causing application program to be rewritten. Modifications at the logical level are necessary whenever the logical structure of the database is altered (for example, when money-market accounts are added to banking system). Logical Data independence means if we add some new columns or remove some columns from table then the user view and programs should not change. It is called the logical independence. For example: consider two users A & B. Both are selecting the empno and ename. If user B add a new column salary in his view/table then it will not effect the external view user; user A, but internal view of database has been changed for both users A & B. Now user A can also print the salary. It means if we change in view then program which use this view need not to be changed.

(ii) Views: In database theory, a **view** is the result set of a *stored* query on the data, which the database users can query just as they would in a persistent database collection object. This pre-established query command is kept in the database dictionary. Unlike ordinary *base tables* in a relational database, a view does not form part of the physical schema: as a result set, it is a virtual table computed or collated dynamically from data in the database when access to that view is requested. Changes applied to the data in a relevant *underlying table* are reflected in the data shown in subsequent invocations of the view. In some NoSQL databases, views are the only way to query data.

Views can provide advantages over tables:

- Views can represent a subset of the data contained in a table. Consequently, a view can limit the degree of exposure of the underlying tables to the outer world: a given user may have permission to query the view, while denied access to the rest of the base table.

Scanned by CamScanner

- Views can join and simplify multiple tables into a single virtual table.
- Views can act as aggregated tables, where the database engine aggregates data (sum, average, etc.) and presents the calculated results as part of the data.
- Views can hide the complexity of data. For example, a view could appear as Sales2000 or Sales2001, transparently partitioning the actual underlying table.
- Views take very little space to store; the database contains only the definition of a view, not a copy of all the data that it presents.

Depending on the SQL engine used, views can provide extra security.

Q.1.(*b*) List any two differences between a File Processing System and a Database Management System.

Ans. Difference between File processing system and DBMS:

1. 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.

2. 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.

Redundancy is control in DBMS, but not in file system.

Q.1.(*c*) Define the concept of aggregation in E-R diagrams. Give any one example where this concept is useful.

Ans. Consider a data box with information about employes who work on a particular project and use number of machines doing that work. We get the E.R. diagram shown in fig.

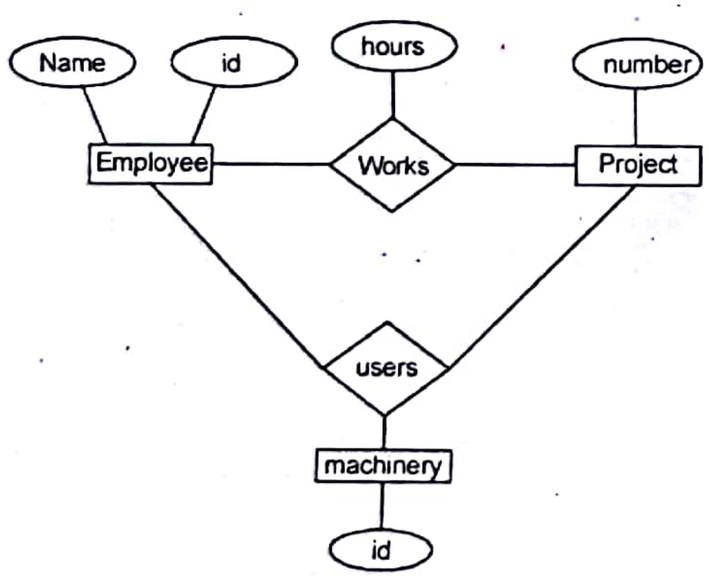


Fig. E.R. diagram with redundant relationship

The solution is to use aggregation : An abstraction through which relationship are treated as higher level entities.

For an example we treat the relationship set works and the entity sets employee and project as a higher level entity set called works.

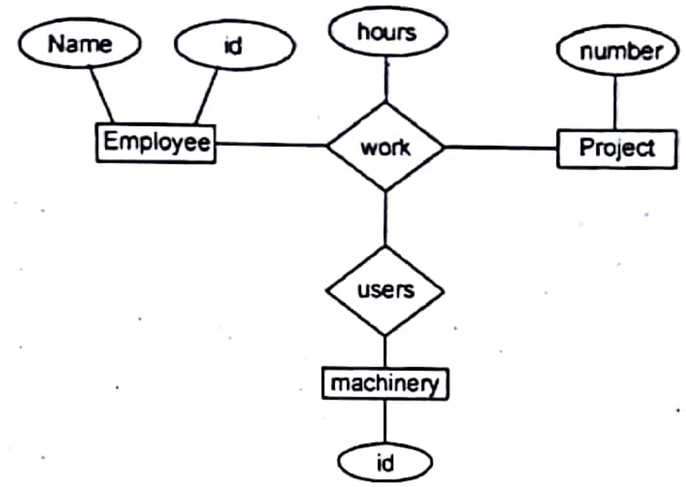


Fig. E.R. diagram with aggregation

Transforming an E.R diagram with aggregation into tabuler form is easy. We create a table for each entity and relationship set as before. The table for relationship set users contain a column for each attribute in the primary key of machinery and works.

Q.1.(*d*) Explain why navigationis simpler in relational data model than in hierarchical data model.

Ans. A hierarchical database model is a data model in which the data is organized into a tree-like structure. The data is stored as **records** which are connected to one another through **links**. A record is a collection of fields, with each field containing only one value. The **entity type** of a record defines which fields the record contains.

A record in the hierarchical database model corresponds to a row (or tuple) in the relational database model and an entity type corresponds to a table (or relation).

The hierarchical database model mandates that each child record has only one parent, whereas each parent record can have one or more child records. In order to retrieve data from a hierarchical database the whole tree needs to be traversed starting from the root node. This model is recognized as the first database model created by IBM in the 1960s

. A relational database allows the definition of data structures, storage and retrieval operations and integrity constraints. In such a database the data and relations between them are organized into tables. A table is a collection of records and each record in a table contains the same fields.[1] The contents of a table can be permanently saved for future use.

Data is presented as a collection of relations. Each relation is depicted as a table.

Columns are attributes that belong to the entity modeled by the table (ex. In a student table, you could have name, address, student ID, major, etc.). Each row ("tuple") represents a single entity (ex. In a student table, John Smith, 14 Oak St, 9002342, Accounting, would represent one student entity). Every table has a set of attributes that taken together as a "key" (technically, a "superkey") uniquely identifies each entity (Ex. In the student table, "student ID" would uniquely identify each student – no two students would have the same student ID).

Thus through the key it is very easy to navigate as compared to hierarchical database model.

Scanned by CamScanner

Q.1.(e) Write an equivalent relational algebra expression for the following SQL query:

Select $A_1, A_2 \ldots\ldots A_n$

From $r_1\ r_2 \ldots\ldots r_m$

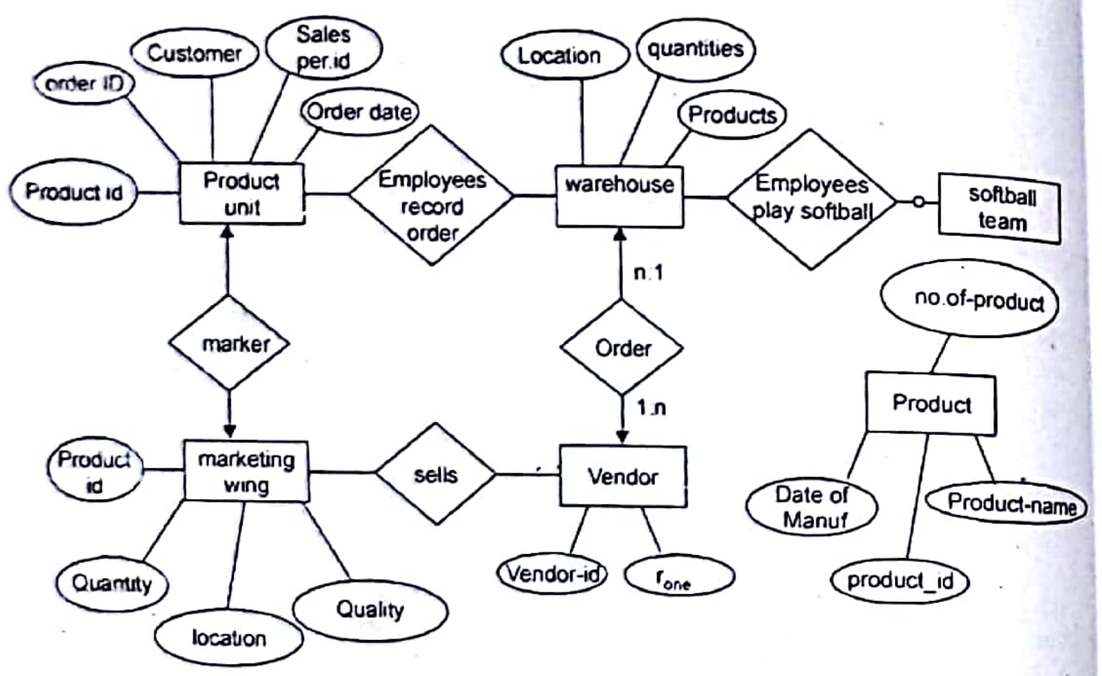
where P

Ans. $\pi_{A_1 \ldots A_n}$

$(\sigma_P (R_1 \times R_2 \ldots R_m))$

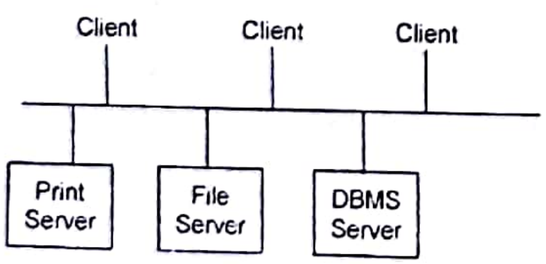
Q.2.(a) Draw an E-R diagram for a garment manufacturing company. The entities include warehouse, production units, marketing wing, vendor and product types. Define the relationships betwen each of these entities. Choose the attribute so that they define a particular entity properly. (4)

Ans.



Q.2.(b) Explain the two tier and three tier architecture of databases with proper digrams. (4)

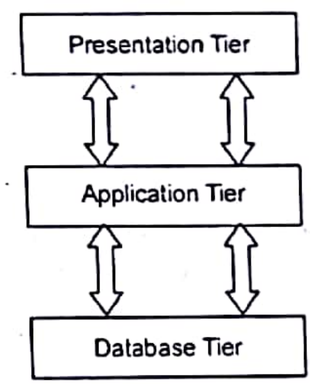
Ans. The two-tier architecture



If the architecture of DBMS is 2-tier then must have some application, which uses the DBMS. Programmers use 2-tier architecture where they access DBMS by means of application. Here application tier is entirely independent of database in term of operation, design and programming.

3-tier architecture: Most widely used architecture is 3-tier architecture. 3-tier architecture separates it tier from each other on basis of users. It is described as follows:

- **Database (Data) Tier:** At this tier, only database resides. Database along with its query processing languages sits in layer-3 of 3-tier architecture. It also contains all relations and their constraints.
- **Application (Middle) Tier:** At this tier the application server and program, which access database, resides. For a user this application tier works as abstracted view of database. Users are unaware of any existence of database beyond application. For database-tier, application tier is the user of it. Database tier is not aware of any other user beyond application tier. This tier works as mediator between the two.
- **User (Presentation) Tier:** An end user sits on this tier. From a users aspect this tier is everything. He/she doesn't know about any existence or form of database beyond this layer. At this layer multiple views of database can be provided by the application. All views are generated by applications, which resides in application tier.



Q.2.(c) Explain tuple variables in SQL. (2)

Ans. SELECT CUSTOMER_NAME, T.LOAN_NUMBER, S.AMOUNT
FROM BORROWER AS T, LOAN AS S
WHERE T.LOAN_NUMBER = S.LOAN_NUMBER

So the tuple variables here are T and S?

They are useful for saving typing, but there are other reasons to use them:

- If you join a table to itself you must give it two different names otherwise referencing the table would be ambiguous.
- It can be useful to give names to derived tables, and in some database systems it is *required*... even if you never refer to the name.

Q.3.(a) Differentiate betwen primary key, super key and candidate key. Support your answer with examples. (3)

Ans. A candidate key is the most minimal subset of fields that uniquely identifies a tuple. For example if you have a candidate key on the column "user_id" and "pet_id" you'll never have more than 1 tuple with the same user_id and pet_id and neither user_id nor pet_id individually will work as a unique identifier for the tuple.

- A super key is a set of fields that contains a key. Using the above example where the combination of "user_id" and "pet_id" uniquely identifies a tuple if we added "pet_name" (which is not key because we can have multiple pets named "fluffy") it would be a super key. Basically it's like a candidate key without the "minimal subset of fields" constraint.

Scanned by CamScanner

A primary key is a candidate key that you tell the DB to optimize on. There might be multiple ways of referring to a unique tuple (ie. multiple candidate keys) but you can specify one when you're creating the table that you will use the most frequently.

Q.3.(b) Why sometimes weak entity sets are needed in a database? (2)

Ans. Weak entity set:

It inherits at least of it's primary key from the entity to which it is related.

It cannot exist at least part of it's primary Key from the entity to which it is relatyed.

Thus when we need to inherit the properties of primary key we use the weak entity set.

Q.3.(c) Consider the following scheme for institute library. (5)

Student (Roll No., Name, Father_Name, Branch)

Book (ISBN, Title, Author, Publisher)

Issue (Roll No., ISBN, Data_of_Issue)

Write the following queries in relational algebra.

(i) List Roll Number and Name of all students of the branch 'CSE'.

(ii) Find the name of students who have issued a book published by 'ABC' publisher.

(iii) List title of all books and their authors issued by a student 'XYZ'.

(iv) List title of all boks issued on or before Jan 1, 2014.

(v) List all books published by publisher 'ABC'.

Ans. (i) $\sigma_{S.\text{roll no.}, S.\text{name}}$ (student) $\pi_{s.\text{branch}}$ = "CSE"

(ii) $\sigma_{S.\text{name}}$ (student $\bowtie$ Book B) $\pi_{B.\text{publisher}}$ = "ABC"

(iii) $\sigma_{B.\text{title}, B.\text{author}}$ (Book B $\bowtie$ Issue I $\bowtie$ student S)

$\pi_{s.\text{name}}$ = "xyz"

(iv) $\sigma_{B.\text{ISBN}, I.\text{Date_of_Issue}}$ (Book B $\bowtie$ Issue I)

$\pi_{I.\text{Date_of_Issue}}$ = 'Jan-1-2014'

(v) $\sigma_{B.\text{publisher}, B.\text{Title}}$ (Book B)

$\pi_{B.\text{publisher}}$ = 'ABC'.

Q.4.(a) BOOK (Bookid, Title, Publisher Name)

BOOK_Authors Bookid, Author Name)

Publisher (Name, Address, Phone)

Book_Copies (Bookid, Branchid, No_of_copies)

Book_loans (Bookid, Branchid, Card No. Date Out, Due date)

Library_Branch (Branchid, Branch Name, Address)

Borrower (Card No., Name, Address, Phone)

Now, write SQL queries for the following:

(i) How many copies of the book titled 'The Lost Tribe' are owned by the library branch whose name is "Sharpstown"? (2)

(ii) Retrieve the names of all borrowers who do not have any books checked out. (1)

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

(iv) Retrieve the names, address and number of books checked out for all borrowers who have ore than five books checked out. (2)

Ans. (i) Select No_of_copies

from books-copies, library-Branch, Book.

where Titte = 'The lost Tribe' and branch name = (Shopshown)

and books copies.bookid = book.Bookid and

book_copies.Branch_id = library_Branch_Branch_id:

(ii) Select No_of_copies from Book_copies

where Books_copies.No. of copies is Null.

(iv) Select branch_name, address, count (Book_id)

From. Book_loans, library_Branch

group-by card No.

having count (book_id)≥5

(iii) Select brand name, count (book_id)

from library_branch, Book_loans

group by book_id, branch.name

Q.4.(b) Explain the following:

(i) Natural Join (ii) Outer Join

Ans. (i) Natural join is a join that creates an implicit join clause on the common columns in the two tables being joined.

(ii) The full outer join keyword sets on all rows from the left table (table 1) and from the right table (table 2)

Scanned by CamScanner

SECOND TERM EXAMINATION
FOURTH SEMESTER [2014]
DATABASE MANAGEMENT SYSTEM [ETCS-208]

Time : 1.30 hrs. **M.M. : 30**

Note: *Attempt Q.No. 1 which is compulsory and any two more questions from remaining.*

Q.1.(a) Define functional dependency. Explain the property of augmentation. **(5×2=10)**

Ans. Functional dependency is a relationship that exists when one attribute uniquely determines another attribute. If R is a relation with attributes X and Y, a functional dependency between the attributes is represented as X->Y, which specifies Y is functionally dependent on X. Here X is a determinant set and Y is a dependent attribute. Each value of X is associated precisely with one Y value.

Augmentation rule in Functional dependencey states that

when X -> Y , then XZ->YZ.

Q.1.(b) Why are certain functional dependencies called trivial dependencies.

Ans. A trivial functional dependency occurs when you describe a functional dependency of an attribute on a collection of attributes that includes the original attribute. This type of functional dependency is called trivial because it can be derived from common sense. It is obvious that if you already know the value of B, then the value of B can be uniquely determined by that knowledge. Thus certain functiona;l dependencies arte called trivial dependenrcies.

Q.1.(c) Differentiate between 3NF and BCNF.

Ans. A relation, R, is in 3NF iff for every nontrivial FD (X → A) satisfied by R at least ONE of the following conditions is true:

(a) X is a superkey for R, or

(b) A is a key attribute for R

BCNF requires (a) but doesn't treat (b) as a special case of its own. In other words BCNF requires that every nontrivial determinant is a superkey even its dependent attributes happen to be part of a key.

A relation, R, is in BCNF iff for every nontrivial FD (X->A) satisfied by R the following condition is true:

(a) X is a superkey for R

BCNF is therefore more strict.

Q.1.(d) When do two operations in a schedule are said to conflict?

Ans. Condition for the conflict that can occur:

Conflict Serializability: Instructions Ii and Ij, of transactions Ti and Tj respectively,conflict if and only if there exists some item Q accessed by both Ii and Ij, and at least one of these instructions wrote Q.

1. Ii = **read**(Q), Ij = **read**(Q). Ii and Ij don't conflict.
2. Ii = **read**(Q), Ij = **write**(Q). They conflict.

3. Ii = **write**(Q), Ij = **read**(Q). They conflict.
4. Ii = **write**(Q), Ij = **write**(Q). They conflict.

Intuitively, a conflict between Ii and Ij forces a (logical) temporal order between them. If Ii and Ij are consecutive in a schedule and they do not conflict, their results would remain the same even if they had been interchanged in the schedule.

Q.1.(e) Write an algorithm for unlocking operation for two mode (read-wirte or shared exclusive) locks.

Ans. The following code performs the unlock operation:

```
    if LOCK (X) = "write-locked" then
begin LOCK (X) ¬ "unlocked";
      wakes up one of the transactions, if any
end
else if LOCK (X) ¬ "read-locked" then
     begin
      no_of_reads (X) ¬ no_of_reads (X) -1
      if no_of_reads (X) = 0 then
      begin
          LOCK (X) = "unlocked";
          wake up one of the transactions, if any
      end
      end;
```

The following code performs the unlock operation:

```
    if LOCK (X) = "write-locked" then
begin LOCK (X) ← "unlocked";
      wakes up one of the transactions, if any
end
else if LOCK (X) ← "read-locked" then
     begin
      no_of_reads (X) ← no_of_reads (X) -1
      if no_of_reads (X) = 0 then
      begin
          LOCK (X) = "unlocked";
          wake up one of the transactions, if any
      end
      end;
```

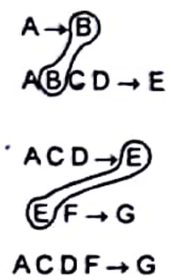
Q.2.(a) Given the following set of functional dependencies on scheme R (A, B, C, D, E, F, G)

A → B, ABCD → E, EF → G **(3)**

Determine if ACDF → G holds on R.

Scanned by CamScanner

Ans.



Thus ACDF → G holds on R.

Q.2.(b) Suppose a relational scheme R = (A, B, C, D, E) is decomposed into (A, B, C) and (A, D, E). Prove that this decomposition is a lossless join decomposition, if the following FD' hold on R: (3)

A → BC

CD → E

B → D

E → A

Ans. A decomposition $\{R_1, R_2\}$ is a loss less 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 candidate key. Therefore $R_1 \cap R_2 \rightarrow R_1$.

Q.2.(c) Consider the relation $R = \{A, B, C, D, E, F, G, H, I, J\}$ and set of functional dependencies $F = \{\{A, B\} \rightarrow \{C\}, \{A\} \rightarrow \{D, E\}, \{B\} \rightarrow \{F\}, \{F\} \rightarrow \{G, H\}, \{D\} \rightarrow \{I, J\}.\}$ What is the key for R?

Decompose R into 2 NF and 3 NF relations.

Ans. A minimal set of attributes whose closure includes all the attributes in R is a key. Since the closure of $\{A, B\}$, $\{A, B\}^+ = R$, one key of R is $\{A, B\}$ (in this case, it is the only key).

To normalize R intuitively into 2NF then 3NF, we take the following steps: First, identify partial dependencies that violate 2NF. These are attributes that are functionally dependent on either parts of the key. {A} or {B} alone. We can calculate the closures $\{A\}^+$ and $\{B\}^+$ to determine partially dependent attributes:

$\{A\}^+ = \{A, D, E, I, J\}$. Hence $\{A\} \rightarrow \{D, E, I, J\}$ ($\{A\} \rightarrow \{A\}$ is a trival dependency)

$\{B\}^+ = \{B, F, G, H\}$, hence $\{A\} \rightarrow \{F, G, H\}$ ($\{B\} \rightarrow \{B\}$ is trivial dependency)

To normalize into 2NF, we remove the attributes that are functionally dependent on part of the key (A or B) from R and place them in seperate relations R_1 and R_2, along with the part of the key they depend on (A or B), which are copied into each of these relations but also remains in the original relation, which we call R_3 below:

$R_1 = \{A, D, E, I, J\}$, $R_2 = \{B, F, G, H\}$, $R_3 = \{A, B, C\}$

The new keys for R_1, R_2, R_3 are underlined. Next, we look for transitive dependencies in R_1, R_2, R_3. The relation R_1 has the transitive dependency $\{A\} \rightarrow \{D\} \rightarrow \{I, J\}$, so we remove the transitively dependent attributes $\{I, J\}$ from R_1 into a relation R_{11} and copy the attribute D they are dependent on R_{11}. The remaining attributes are kept in a relation R_{12}. Hence, R_1 is decomposed into R_{11} and R_{12} as follows:

$R_{11} = \{D, I, J\}$, $R_{12} = \{A, D, E\}$

The relation R_2 is similarly decomposed into R_{21} and R_{22} based on the transitive dependency.

$\{B\} \rightarrow \{F\} \rightarrow \{G, H\}$:

$R_{21} = \{F, G, H\}$, $R_{22} = \{B, F\}$

The final set of relations in $3NF$ are $\{R_{11}, R_{12}, R_{21}, R_{22}, R_3\}$

Q.3.(a) What is meant by the concurrent xecution of database transactions in a multiuser system? Discuss why concurrency control is needed with examples. (5)

Ans. Concurrent execution of database transactions in a multi-user system means that any number of users can use the same database at the same time. Concurrency control is needed in order to avoid inconsistencies in the database.

Here is an example of how this scenario can lead to an inconsistency:

Assume we have two users, Alice and Bob, who both have access to the same bank account. Alice wants to deposit \$100 and Bob wants to withdraw \$200. Assuming there is \$500 in the account, here is how the execution *might* take place if they perform their actions at the same time:

1. Alice gets initial amount (x = \$500)
2. Bob gets initial amount (x = \$500)
3. Alice deposits \$100 (x + 100) = \$600
4. Bob withdraws \$200 (x – 200) = \$300
5. Alice saves the new balance (\$600)
6. Bob saves the new balance (\$300)

New balance after both actions should be \$400. Now the database is in an inconsistent state.

Concurrency control can prevent inconsistencies by providing Alice with a temporary "lock" on the database until she is done with her action.

Q.3.(b) What are transaction commit points and why are they important? (2)

Ans. Transaction is a sequence of operations that performs logical unit of operations on a database which include one or more operations as insert, delete, update and modification, Transaction is a executing program, the boundaries of a transaction is define as the explicit begin of transaction and end of transaction. Read only Transaction performs only retieve data from the data base and the read–write operations includes both retrieving and updating transactions.

Transaction is the executing program that executes the operations on the database as read(x), write (x), so the transactions plays a vital role among database operations.Thus transaction commit points is very essential inorder to record the details of the current transaction.

Q.3.(c) Define the following:

(i) Cascading roll back (ii) Serilizable schedule

Ans. (i) Cascading rollback:

(i) In database technologies, a **rollback** is an operation which returns the database to some previous state. Rollbacks are important for database integrity, because they mean that the database can be restored to a clean copy even after erroneous operations are performed. They are crucial for recovering from database server crashes; by rolling back any transaction which was active at the time of the crash, the database is restored to a consistent state. The phenomenon where a single transaction failure leads to a series of transaction rollbacks is known as Cascading rollback

Scanned by CamScanner

(ii) Serializable Schedule: In concurrency control of databases,[1][2] transaction processing (transaction management), and various transactional applications (e.g., transactional memory[3] and software transactional memory), both centralized and distributed, a transaction schedule is serializable if its outcome (e.g., the resulting database state) is equal to the outcome of its transactions executed serially, i.e., sequentially without overlapping in time. Transactions are normally executed concurrently (they overlap), since this is the most efficient way. Serializability is the major correctness criterion for concurrent transactions' executions. It is considered the highest level of isolation between transactions, and plays an essential role in concurrency control. As such it is supported in all general purpose database systems. *Strong strict two-phase locking* (SS2PL) is a popular serializability mechanism utilized in most of the database systems (in various variants) since their early days in the 1970s.

Q.4.(a) Describe the shadow paging recovery technique. Under what circumstances does it not require a log? (3)

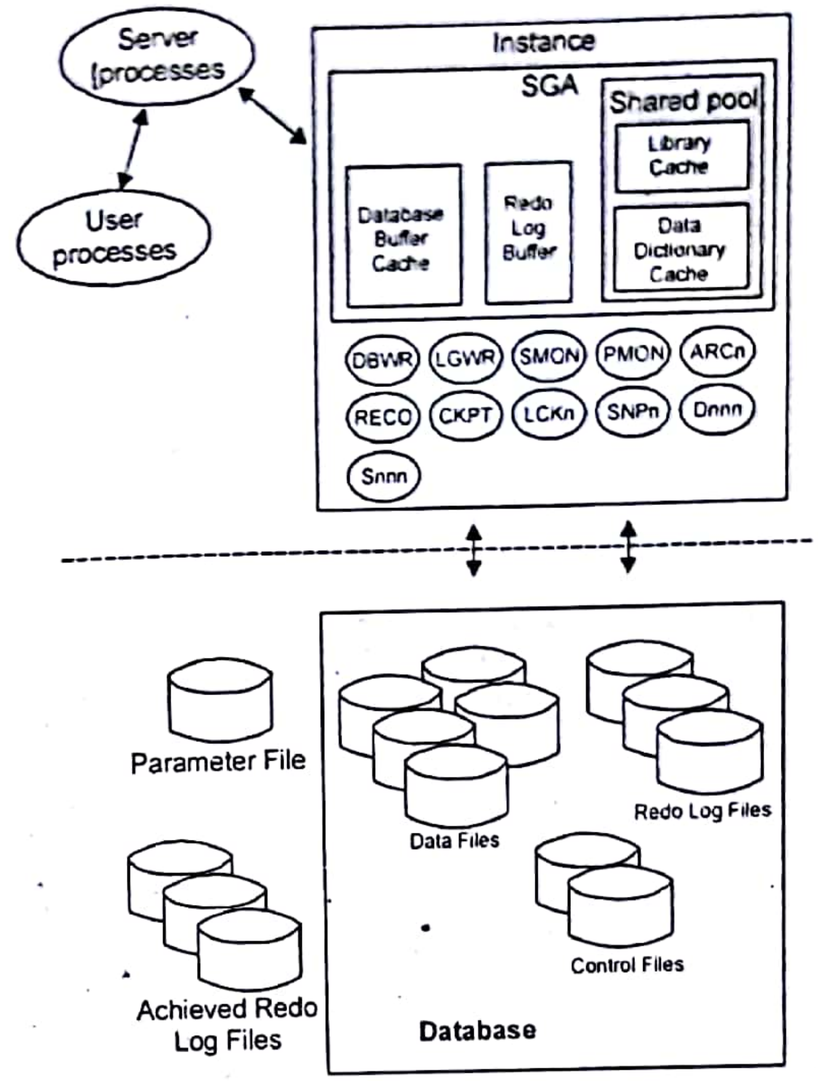
Ans. Shadow paging recovery scheme does not require the use of a log in a single-user environment. In a multiuser environment, a log may be needed for the concurrency control method. Shadow paging considers the database to be made up of a number of fixed-size disk pages (or disk blocks)—say, n—for recovery purposes. A directory with n entries is constructed, where the ith entry points to the ith database page on disk. The directory is kept in main memory if it is not too large, and all references—reads or writes—to database pages on disk go through it. When a transaction begins executing, the current directory—whose entries point to the most recent or current database pages on disk—is copied into a shadow directory. The shadow directory is then saved on disk while the current directory is used by the transaction.

During transaction execution, the shadow directory is *never* modified. When a write_item operation is performed, a new copy of the modified database page is created, but the old copy of that page is *not overwritten*. Instead, the new page is written elsewhere—on some previously unused disk block. The current directory entry is modified to point to the new disk block, whereas the shadow directory is not modified and continues to point to the old unmodified disk block. For pages updated by the transaction, two versions are kept. The old version is referenced by the shadow directory, and the new version by the current directory.

To recover from a failure during transaction execution, it is sufficient to free the modified database pages and to discard the current directory. The state of the database before transaction execution is available through the shadow directory, and that state is recovered by reinstating the shadow directory. The database thus is returned to its state prior to the transaction that was executing when the crash occurred, and any modified pages are discarded. Committing a transaction corresponds to discarding the previous shadow directory. Since recovery involves neither undoing nor redoing data items, this technique can be categorized as a NO-UNDO/NO-REDO technique for recovery.

Q.4.(b) With a neat diagram, explain the architecture of oracle 8. (5)

Ans. The Oracle8 database is comprised of two parts, the Oracle instance and the Oracle database. The Oracle instance is the software that manipulates the database. The instance refers to the memory structure and a set of background processes. These processes share the memory structure known as the SGA (System Global Area). The SGA is composed of various caches. The DBMS (Database Management System) really resides on the hard drive. The instance is created when the SGA has been allocated and the processes are copied into memory, as shown below. When the instance is shutdown, the SGA is deallocated from memory.



Q.4.(c) Differentiate between time stamping techniques and locking techniques with reference to deadlocks. (3)

Ans. This is the minimum time elapsed between two adjacent timestamps. If the resolution of the timestamp is too large (coarse), the possibility of two or more timestamps being equal is increased and thus enabling some transactions to commit out of correct order. For example, assuming that we have a system that can create one hundred unique timestamps per second, and given two events that occur 2 milliseconds apart, they will probably be given the same timestamp even though they actually occurred at different times.

Even though the technique time stamp locking is a non-locking one, in as much as the Object is not locked from concurrent access for the duration of a transaction, the act of recording each timestamp against the Object requires an extremely short duration lock on the Object or its proxy.

Scanned by CamScanner

MODEL TEST PAPER
(According to New Syllabus)
END TERM EXAMINATION
FOURTH SEMESTER [B.TECH]
DATABASE MANAGEMENT SYSTEM [ETCS-208]

Time : 3 hrs. **M.M. : 75**

Note: *Attempt one question from each unit in which Q.No. 1 is compulsory.*

Q.1.(a) Why we need of data base systems?

Ans. (*i*) A database can be defined as a collection of non-redundant data which can be shared by different systems.

(*ii*) A database implies separation of physical storage from use of the data by an application program to achieve program/data impedance.

(*iii*) Using a database system, the user or programmer need not know the details of how the data are stored and such details are usually "transparent" to the user.

Q.1.(b) Explain the terms Data, Database, Hardware and Software?

Ans. Data: The data is the raw facts and figure. In other words, meaningless information is known as Data.

For example: 56 that means we do not know about the 56 is marks or Age.

Database: A database is a mechanized, formally defined, centrally controlled collection of data in an organization. The data records are physically organized and stored so as to promote availability, evolvability and integrity which are some of the important objectives.

Hardware: The hardware consists of the secondary storage volumes—disks, drums etc. which the database resides, together with the associated devices, control units, channels and so forth.

Software: Between the hardware and the users there is a layer of software, usually called the database management system or DBMS. All requests from users for access to the database are handled by the DBMS.

Q.1.(c) What are the properties of Relation?

Ans. • In a table, the order of rows is immaterial.

• Order of columns is immaterial.

• In a table, there are no duplicate rows.

• Each row of the table contains only one value.

• In a table, the values in each column must come from the same attribute domain.

Q.1.(d) What do you mean by Functional Dependency?

Ans. Functional Dependency: A functional dependency, denoted by $x \rightarrow y$, between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuples that can form a relation state r of R. The constraint is that, for any two tuples t_1 and t_2 in r that have $t_1[x] = t_2[x]$, they must also have $t_1[y] = t_2[y]$.

This means that the values of the y component of a tuble in r depend on, or are determined by, the values of the X component; alternatively the values of the X component of a tuble uniquely (or functionally) determine the values of the Y component. The set of attributes X is called the left-hand side of the FD, and Y is called the right-hand side.

Q.1.(e) What is the difference between a unique constraint and a primary key?

Ans. Primary Key Constraint:

• A primary key is a single field or combination of fields that uniquely defines a record.

• None of the fields that are part of the primary key can contain a null value.

Unique Key Constraint:

• A unique constraint is a single field or combination of fields that uniquely defines a record.

• Some of the fields can contain null values as long as the combination of values is unique.

Q.1.(f) What are the different rules for strict two phase locking?

Ans. There are number of rules for strict two phase locking such as:

(*i*) When operation access data item within transaction.

(*ii*) If item is not locked, then server locks and proceeds.

(*iii*) If site is held in a conflicting lock by another transaction, transaction must wait till lock released.

(*iv*) If item is held by non-conflicting lock, lock is shared and operation proceeds.

(*v*) If item is already locked by same transaction, lock is promoted if possible.

Q.1.(g) What are the differences between Relational Algebra and Relational Calculus?

Ans.

Relational Algebra	Relational Calculus
• Relational Algebra operations manipulate some relations and provide some expression in the form of queries.	• Relational calculus are formed queries on the basis of pairs of expressions.
• RA have operator like join, Union, Intersection, Division, Difference, Projection, Selection etc.	• RC has tuples and domain oriented expressions.
• RA is procedural language.	• RC is non-procedural query system.
• RA queries are more powerful.	• RC queries are less powerful.

Q.1.(h) What do you mean by Deadlock and Starvation?

Ans. Deadlock: Although two phase locking ensure serializability, they may lead to a deadlock. Deadlock occurs when each transaction T in a set of two or more transactions is waiting for some item that is locked by some other transaction T' in the set.

Starvation: Starvation occurs when a transaction can not proceed for an indefinites period of time while other transactions in the system continue normally.

Q.1.(i) What are different commands used for DCL. Explain one of them?

Ans. DCL commands: COMMIT, ROLLBACK, GRANT and REVOKE.

Commit command: It is the transactional command used to save changes invoked by a transaction to the database. It saves all transactions to the database since the last COMMIT or ROLLBACK command Syntax is

Scanned by CamScanner

COMMIT [WORK];

Where COMMIT is the keyword and work is optional keyword and it is used to make the command more user friendly.

Example DELETE FROM EMP

WHERE BASIC > 7000;

COMMIT;

After deletion, a COMMAND is issued to save the changes to the database, completing the transaction.

Q.1.(j) Explain GROUP by CLAUSE and HAVING CLAUSE with syntax?

Ans. Group by Clause: The Group by Clause can be used in a SELECT statement to collect data across multiple records and group the results by one or more columns.

Syntax: SELECT column 1, ... column *n*, aggregate-function (expression) FROM tables.

WHERE Predicates

GROUP by Column 1, Column 2, ... Column n;

Having Clause: It is used in combination with the Group By Clause. It can be used in a SELECT statement to filter the records that a GROUP by returns.

Syntax:

GROUP BY Column 1, Column 2 ..... Column-n

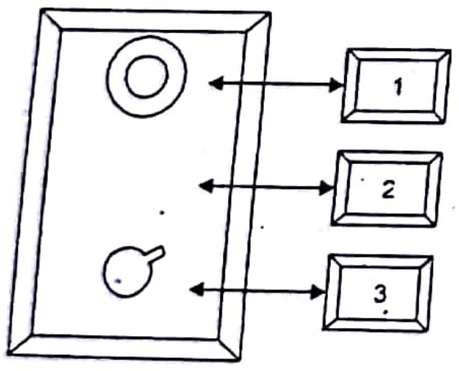
HAVING condition 1 ..... condition-n;

UNIT-I

Q.2.(a) With the help of mean and clean diagram explain various client server architecture of database management system.

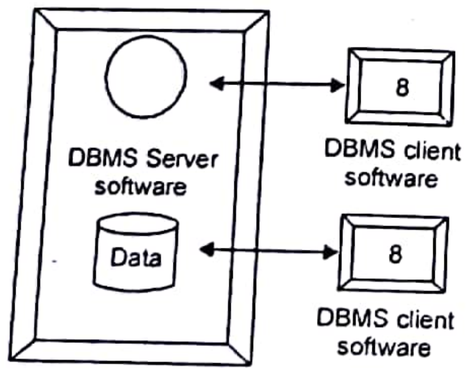
Ans. Database client server architecture are partitioned into 3 types:

(1) Single Tier: In single tier system all software are present on a single machine (server system), which are accesed by terminals.



Server single tier

Fig. (A)



Two tier system

Fig. (B)

(2) Two Tier: The application is divided into a component that besides at the client machine, which invokes database system functionally at the server machine through query language statement (Fig. B) API (Application program Interface) standards like ODBC and JDBC are required.

(3) Three Tier: This system adds middle ware which provide a way for the clients of one DBMS to access data from another DBMS on other machine (Fig. C). This system is more suitable for very large applications.

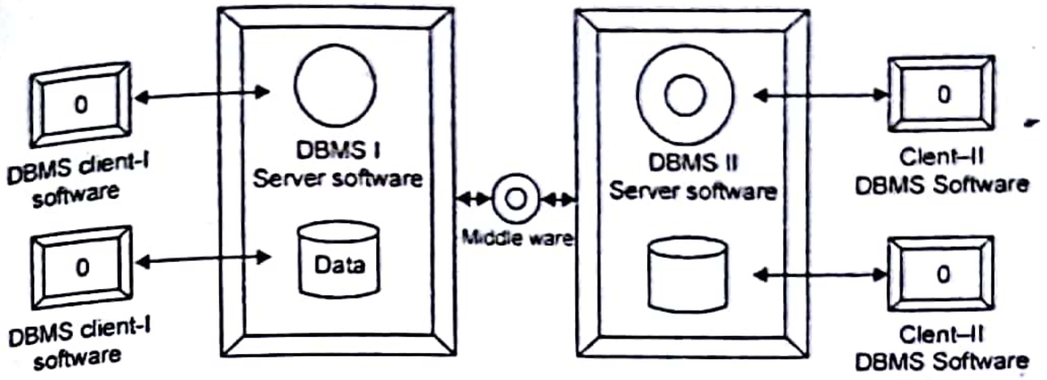


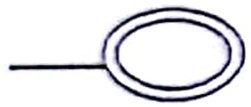
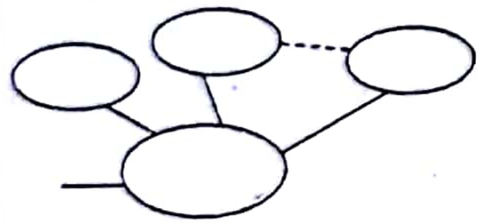
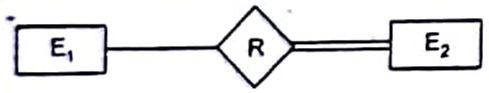
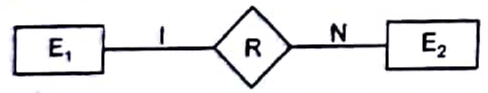
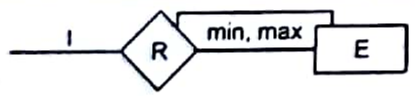
Fig. (c) Three-Tier System

Q.2.(b) Draw various symbols used for Entity-Relationship diagram.

Ans.

Symbol	Meaning
	Entity Type
	Weak entity Type
	Relationship Type
	Identifying Relationship Type
	Attribute
	Key Attribute

Scanned by CamScanner

	Multivalued Attribute
	Composite Attribute
	Derived Attribute
	Total Participation of E_2 in R
	Cardinality Ratio 1 : N for $E_1 : E_2$ in R.
	Structural constraints (min, max) on participation of E in R.

Q.3. Describe the various relationship constraints by giving suitable example?

Ans. There are two types of relationship constraints:

1. Participation constraints
2. Cardinality Ratio

1. Participation Constraints: It specifies whether the existence of an entity depends on other entity via the relationship type. There are two types of participation constraints.

1.(a) Total Participation: In Total Participation every entity in entity set must be depend on another entity. For example, if a organisation policy states that every employee must work for a department. Then an employee entity can exist only if it is related to a department entity via WORKS-FOR relationship.

Total participation is also known as existence dependency. In ER diagrams, it is represented as a double line connecting the participating entity type to the relationship as shown in fig. 1. (Total participation of E_2 in R).

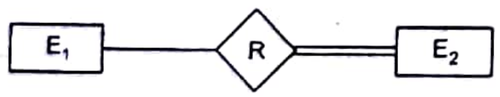


Fig. 1

1(b) Partial Participation: In Partial Partition, some entity in entity set are depend on another entity. For example, it is not necessary that all employee manage some department, so the participation of Employee in the Manages relationship Type is partial.

2. Cardinality Ratio: Cardinality ratio for a binary relationship specifies the number of relationship instances that an entity can participate in a relation set. There are four types of cardinality Ratio.

2(a) One-to-One (1 : 1): The relationship between a Department and a Manager is usually one-to-one as shown in fig. (2).

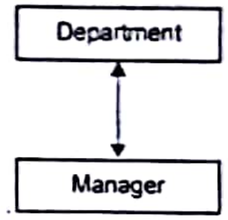


Fig. 2

Fig. (3) shows some instances of these relationships.

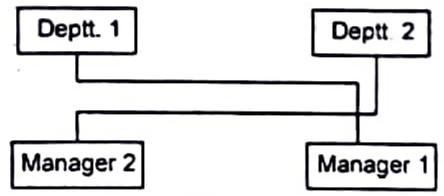


Fig. 3

2(b) One-to-Many (1:N): The relationship between manager and employee is one-to-many as shown in fig. (4) because there are several employees reporting to the manager.

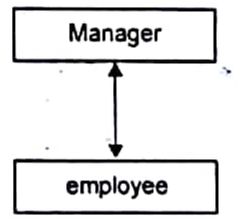


Fig. 4

Fig. (5) shows some instances of these relationship.

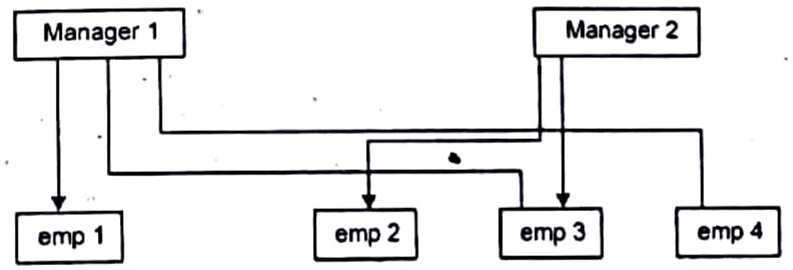


Fig. 5

2(c) Many-to-one (N : 1): The relationship between Dependent and employee is many-to-one as shown in fig. (6).

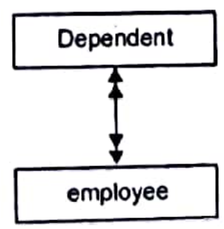


Fig. 6

Scanned by CamScanner

2(*d*) Many-to-Many (M:N): The relationship between Employee and Project is many-to-many as shown in fig. (7). Because each employee could be involved in a number of different projects and a number of employees could be working on a given project.

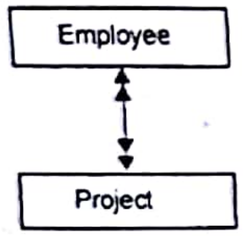


Fig. 7

Fig. (8) given below shows some instances of these relationships.

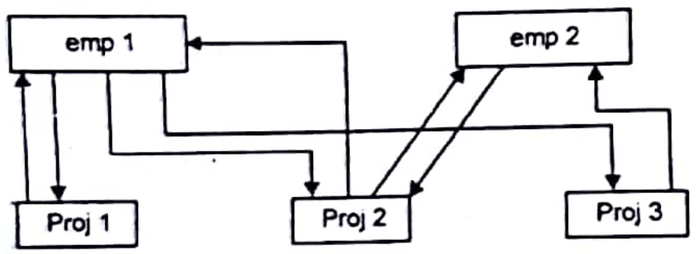


Fig. 8

Structural Constraints: Cardinality ratio and participation constraints are together framed as structural constraints on a reaction.

UNIT-II

Q.4.(*a*) Given the following relations:

Vehicle (Reg No., Make, Colour)

Person (ENO, Name, Address)

Owner (ENO, Reg No.)

Write expressions in relational algebra or SQL to answer the following queries.

(*i*) List the names of persons who do not own any car.

(*ii*) List the names of persons who own only Maruti Cars.

Ans. Refer to Q.No. 4(*a*) of First Term Examination 2013.

Q.4.(*b*) Discuss division operation in relational algebra giving an example?

Ans. Refer to Q.No. 4(*b*) of First Term Examination 2013.

Q.5.(*a*) Given the following relations:

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

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

Account (Account-number, Branch-name, Balance)

Depositor (Customer-name, Account-number)

(*i*) Find all loan numbers for loans made at the ABC branch with loan amounts greater that $ 1200.

(*ii*) For all customers who have a loan from the bank, find their names, loan numbers and loan amount.

(*iii*) Find the names of all customers whose street address includes the substring 'Main'.

(*iv*) To list in alphabetic order all customers who have a loan at the ABC Branch.

(*v*) Find the average account balance at each branch.

(*vi*) Find all customers who do have a loan at the bank, but do not have an account at the bank.

Ans.

```
(i)   SELECT    Loan_number
      from      Loan
      WHERE     Branch-name = 'ABC' and amount > 1200.
(ii)  SELECT    Customer_name, borrower.loan_number,
      amount    from borrower, loan
      WHERE     borrower. loan_number = loan, loan_number
(iii) SELECT    Customer_name
      from      Customer
      WHERE     Costumer-street like '% Main %"
(iv)  SELECT    distinct customer_name
      from      borrower, loan
      WHERE     borrower, loan-number = loan. Loan-number and
                branch_name = 'ABC'
      ORDER     by Customer_name.
(v)   SELECT    Branch-name, AVG (Balance)
      from      account
      GROUP     BY Branch_name
(vi)  SELECT    distinct customer_nam
      from      Borrower
      WHERE     Customer_name NOT IN (SELECT Customer_name from
                Depositer).
```

UNIT-III

Q.6. Write down the procedure or Algorithm to finding a Minimal Cover F for a set of FD with suitable example?

Ans. Algorithm:

(*i*) Set F : = E.

(*ii*) Replace each functional dependency $x \rightarrow \{A_1, A_2, ..A_n\}$ in F by the n functional dependencies $x \rightarrow A_1, x \rightarrow A_2, ... x \rightarrow A_n$.

(*iii*) For each functional dependency $x \rightarrow A$ in F-for each attribute B that is an element of X if $\{\{F - \{X \rightarrow A\}\} \cup \{(X - \{B\}) \rightarrow A\}\}$ is equivalent to F,

then replace $X \rightarrow A$ with $(X - \{B3) \rightarrow A$ in F.

(iv) For each remaining functional dependency $X \rightarrow A$ in F.

if $\{F - \{X \rightarrow A\}\}$ is equivalent to F, then remove $X \rightarrow A$ from F.

For example: Let the given set of FDs be E: $\{B \rightarrow A, D \rightarrow A, AB \rightarrow D\}$. We have to find the minimum cover of E.

All above dependencies are in canonical form: so we have completed step of Algorithm and can proceed to step (ii). In step (ii) we need to determine if $AB \rightarrow D$ has any redundant attribute on the left hand side that is can it be replaced by $B \rightarrow D$ or $A \rightarrow D$?

• Since $B \rightarrow A$, by augmenting with B on both sides we have $BB \rightarrow AB$, or $B \rightarrow AB$ (a). However, $AB \rightarrow D$ as given (b).

Scanned by CamScanner

• Hence by the transitive Rule, we get from (a) and (b), $B \rightarrow D$. Hence $AB \rightarrow D$ may be replaced by $B \rightarrow D$.

• We now have a self equivalent to original E, say E': $\{B \rightarrow A, D \rightarrow A, B \rightarrow D\}$. No further reduction is possible in step (ii) since all FDs have a single attribute on the left hand side.

• In step (iii) we look for a redundant FD in E_1. By using the transitive rule on $B \rightarrow D$ and $D \rightarrow A$, we derive $B \rightarrow A$. Hence, $B \rightarrow A$ is redundant in E' and can be eliminated.

$\Rightarrow$ Hence the minimum cover of E is $\{B \rightarrow D, D \rightarrow A\}$

Q.7.(a) Define Join Dependencies and fifth Normal Form. Why is SNF also called Project-Join Normal Form (PJNF)?

Ans. Join Dependency: A Join Dependency (JD) denoted by JD $(R_1, R_2, \ldots, R_n)$, specified on relation scheme R, specifies a constraint on the states r of R. The constraint states that every legal state r of R should have a non additive join decomposition into $R_1, R_2, \ldots R_n$ that is, for every such row have

$$(\pi R_1(r), \pi R_2(r), \ldots, \pi R_n(r)) = r$$

Notice that an MVD is a special case of a JD where $n = 2$. That is, a JD denoted as JD (R_1, R_2) implies an MVD $(R_1 \cap R_2) \rightarrow (R_1 - R_2)$ A join Dependency JD $(R_1, R_2, \ldots R_n)$, specified on relation scheme R, is a trivial JD if one of the relation schemes R_1 in JD $(R_1, R_2 \ldots R_n)$ is equal to R. Such a dependency is called trivial because it has the non-additive join property for any relation state r of R and hence does not specify any constraint on R.

Fifth Normal Form: A relation scheme R is in Fifth Normal Form (SNF) [or Project Join Normal Form (PJNF)) with respect to a set F of functional, multivalued and join dependencies, if for every non-trivial join dependency JD $(R_1, R_2, \ldots R_n)$ in F^+ (that is, implied by F), every R_i is a super key of R.

For example: Consider the SUPPLY all-key relation of Figure 1. Suppose that the following additional constraint always holds: whenever a supplier s supplies part p, and a project j uses part p, and the supplier s supplies at least one part to project j, then suppliers s will also be supplying part p to project j: This constraint can be restarted in other ways and specifies a join dependency JD (R_1, R_2, R_3) among the three projections R_1 (Sname, Part-name), R_2 (Sname, Proj-name) and R_3 (Part-name, Proj-name) of SUPPLY. If this constraint holds, the tuple below the dotted line in Figure 1 must exist in any legal state of the SUPPLY relation that also contains the tuples above the dotted line. Figure 2 shows how the supply relation with the join dependency is decomposed into three relations R_1, R_2 and R_3 that are each in SNF.

Supply

Sname	Part-name	Proj-name
Smith	Bolt	Proj X
Smith	Nut	Proj Y
Adanisky	Bolt	Proj Y
Walton	Nut	Proj Z
Adamsky	Nail	Proj X
Adamsky	Bolt	Proj X
Smith	Bolt	Proj Y

Fig. 1

R1

Sname	Part-name
Smith	Bolt
Smith	Nut
Adamsky	Bolt
Walton	Nut
Adamsky	Nail

R2

Sname	Proj-name
Smith	Proj X
Smith	Proj Y
Adamsky	Proj Y
Walton	Proj Z
Adamsky	Proj X

R3

Part-name	Proj-name
Bolt	Proj X
Nut	Proj Y
Bolt	Proj Y
Nut	Proj Z
Nail	Proj X

Fig. 2

UNIT-IV

Q.8.(a) Explain Replication, Distribution and External data?

Ans. DB2 replication is a product in the DB2 family that provides replication capabilities among DB2 other relational Data sources such as Oracle, Microsoft SQL server, Sybase SQL server and Informix, as well as non-relational data sources such as IBM's IMS. It consists of capture and apply components, which are controlled by administration interfaces. The change capture mechanisms are either "Log-based" for DB2 tables or "trigger-based" in the case of other data sources. The captured changes are stored in temporary staging table areas under the control of DB2 Replication. These staged intermediate tables with changes are then applied to destination tables using regular SQL statements: inserts, updates, and deletes. SQL based transformations can be performed on the intermediate staging tables by using filtering conditions as well as aggregations. The resulting rows can be applied to one or more target tables. All of these actions are controlled by the administrator facility.

In version 8.2, DB2 supports a new feature called queue replication Queue (Q) replication creates a queue support mechanism using IBM's message-queue product to ship captured log records as messages. These messages are extracted from the queues at the receiving end and applied against targets. The apply process can be parallelized and allow for user-specified conflict resolution rules.

Another member of the DB2 family is the DB2 information integrator product, which provides federation, replication (using the replication engine described above), and search capabilities. The federated edition integrates tables in remote DB2 or other relational database into a single distributed database. Users & developers can access various non-relational data sources in tabular format, using wrapper technology. The federation engine provides a cost-based method for query optimization across the different data sites.

Scanned by CamScanner

DB2 supports user-defined table functions that enable, access to non-relational and external data sources. User-defined table functions are created by using the create function statement with the clause return table. Using these features, DB2 is able to participation in the OLE DB protocols.

Finally, DB2 provides full support for distributed transaction processing using the two-phase commit protocol. DB2 can perform all stages of the two-phase commit protocol. As a participant, DB2 can interact with any commercial distributed transaction manager.

Q.9.(a) Explain the difference between three storage type-Volatile, non-volatile and stable?

Ans. Volatile storage: Information residing in volatile storage does not usually survive system crashes. Examples of such storage are main memory and cache memory. Access to volatile storage is extremely fast, both because of the speed of the memory itself, and because it is possible to access and data item, in volatile storage directly.

Non-volatile: Information residing in nonvolatile storage survives system crashes. Examples of such storage are disk and magnetic tapes. Disks are used for on line storage, whereas tapes are used for archival storage. Both, however, are subject to failure. Which may result in loss of information. At the current state of technology, non volatile storage is slower than volatile storage by several orders of magnitude. This is because disk and tape devices are electromechanical, rather than based entirely on chips, as is volatile storage.

Stable storage: Information residing in stable storage is never last (never should be taken with a grain of salt, since theoretically never cannot be guaranteed-for example, it is possible, although extremely unlikely, that a black hole may envelop the earth and permanently destroy all data) although stable storage is theoretically impossible to obtain, it can be closely approximated by technique that makes data loss extremely unlikely.

Q.9.(b) Explain Multiversion two-phase locking using certify locks?

Ans. In this scheme, there are three locking modes for an item-read, write and certify. Hence the state of lock (X) for an item X can be one of road-locked, write locked, certify locked or unlocked. Certify lock is also exclusive lock. We can describe the relationship between read, write and certify locks by means of the lock compatibility table shown in fig. 12.12. Here yes means 'T' can obtain the lock and 'No' means T' must wait until T release the lock.

The idea behind multiversion 2PL is to allow other transaction T' to read an item X while a single transaction T holds a write locks on X. This is accomplished by allowing two versions for each item X; one version must always have been written by some committed transaction. The second version of X while T holds the write lock. Transaction T can write the value of X as needed, without affecting the value of the committed version X.

(a) read/write locking scheme

		Read	Write
T hold the lock	Read	yes	no
	write	no.	no

(b) read/write/certify locking scheme.

		Read	Write	certify
T hold the lock	Read	yes	yes	no
	write	yes	no	no
	certify	no.	no	no

However, once T is ready to commit, it must obtain a certify lock on all items that it currently holds write locks on before it can commit. The certify lock is not compatible with read locks, so the transaction may have to wait until all its write-locked items are released by any reading transactions in order to obtain the certify locks. Once the certify are locked, the committed X of the data item is set to the value of version X', version X' is discarded, and the certify locks are then released.

FIRST TERM EXAMINATION [FEB. 2015]
FOURTH SEMESTER [B.TECH]
DATABASE MANAGEMENT SYSTEM
[ETCS-208]

Time. 1 Hour **MM: 30**

Note: Attempt Q.1. which is compulsory and any two more questions from remaining.

Q.1. Fill in the blanks. **(10)**

Ans. (a) RDBMS stands for Relational Data Base Mangement system.

Ans. (b) DCL stands for Data control Language.

Ans. (c) A special table to check any function in Oracle is dual table.

Ans. (d) An association between two or more entities is relationship.

Ans. (e) The command to eliminate a table from a database is DROP table <table name>.

Ans. (f) Dtatabase level is closest to the users is External.

Ans. (g) A recursive relationship is a relationship between an entity and Itself.

Ans. (h) In a 1:N relationship, the foreign key is placed in the child table.

Ans. (i) A superclass/subclass hierachy is one in which subclass has/have one superclass.

Ans. (j) A supertype entity related to two or more assosiated entities that each contains specialized attributes that apply to some but not all of the instances of the entity.

Q.2. (a) Describe briefly with an example: derived attribute multivalued attribute, composite attribute, complex, attribute, descriptive attribute. **(5)**

Ans. Composite attribute: A composite Atribute is defined as an attribute composed of multiple components each with an independent existence. In this case, some attributes can be further divided to yield smaller components with an independent existence of their own.

For example, The attribute name is a composite attribute which can be further subdivided into f name, m name and *l* name *i.e.*, first name, middle-name and last-name.

Multi-valued Attribute: It is defined as an attribute that multiple values for a single entity. For example, for a student entity, if one of the attributes is 'Hobby', then 'Hobby' can have multiple values be reading, dancing, gardening and so on.

Derived Attribute: It is defined as an attribute that represented a value that is derivable form the value of related attribute or set of attributes, not essentially in the same entity.

For example: The age attribute can be derived from the data-of-birth attribute and so they are related.

Complex Attribute: A complex attribute that is both composite and multi valued.

Example: A customer can have multiple phone numbers, email id's etc.

Q.2. (b) What is data independence? Explain logical and physical data independence. **(5)**

Ans. Refer to Q.1(a)(i) of First Term Examination 2014.

Scanned by CamScanner

Q.3. (a) Consider the two tables T1 (P,Q,R), T2 (S, T, U) as shown is fig 1 Show the results of the following. (5

Table T1		
P	Q	R
a	1	–
c	2	10
t	2	–
a	–	–

Table 2		
S	T	U
a	1	–
d	3	10
r	3	10
a	–	–

Fig. 1.

(i) $\sigma_p = a\ (T1)$

(ii) $\pi_{P,R}\ (T1)$

(iii) $T1 \cup T_2$

(iv) $T_1 \cap T2$

(v) T1 – T2

(i) Ans.

Table T1		
P	*Q*	*R*
a	1	–
a	–	–

(ii) Ans.

Table T1	
P	*R*
a	–
c	10
t	–
a	–

(iii) **T1 ∪ T2**

P	*Q*	*R*	*S*	*T*	*U*
a	1	–	*a*	1	–
c	2	10	*d*	3	10
t	2	–	*r*	3	10
a	–	–	*a*	–	–

(iv)

T1 ∩ T2

There is no result.

(v)

P	*Q*	*R*
a	1	–
c	2	10
t	2	–
a	–	–

T1 – T2

Q.3. (b) Define the concept of aggregation. Give two example of where this concept useful. (5)

Ans. Refer to Q.No. (1) (c) from First Term Examination, 2014.

Q.4. Emplyee (employee-name, street, city) (5×2)

Works (employee-name, company-name, salary)

Company (company-name, city)

Manages (employee-name, Manager name)

Consider the employee database and give an expression in SQL or Relational Algebra for each of the following queries.

(a) Find the name of all employees who work for SBI.

(b) Find the names and cities of residence of all employees who work for SBI.

(c) Find all employees in the database who do not work for SBI.

(d) Modify the database no that Hari now lives in pune.

(e) Delete all tuples in the works relation for employees of SBI.

Ans. (a) π employee-name (°company-name = "SBI" (works))

(b) π employee-name, city (employee ⋈) (° company-name = "SBI" (works))

(c) π employee - name (°company-name <> "SBI" (works))

(d) UPDATE city

Set City = 'Pune'.

where employee-name = 'Hari';

(e) DELETE FROM WORKS

WHERE company -name = 'SBI';

Scanned by CamScanner

SECOND TERM EXAMINATION [APRIL 2015]
FOURTH SEMESTER [B.TECH]
DATABASE MANAGEMENT SYSTEM
[ETCS-208]

Time. 1 Hour **MM: 30**

***Note:** Attempt Q.1. which is compulsory and any tow more questions from remaining.*

Q.1. Fill in the blanks. **(10)**

Ans. (*a*) BCNF stands for "Boyce codd Normal Form"

Ans. (*b*) PL/SQL stands for "Procedural Language/Structured Query Language".

Ans. (*c*) Every view serializable schedule that is not conflict serializable has involve useless write.

Ans. (*d*) Data item can be locked in Exclusive mode where data item can be both read as well as written.

Ans. (*e*) To avoid cascading roll-back a modified protocol called strict two phase locking is used, where is transaction must held all its exclusive locks till it commits/ aborts.

Ans. (*f*) A serializable schedule is one where, for each pair of transctions Ti and Tj such that Tj read data previously written by Ti, the commit operations of Ti appears before the commit operation of Tj.

Ans. (*g*) w- timestamp (Q) is the Largest time-stamp of any transaction that execuated wrtie (Q) successfully.

Ans. (*h*) An MVD $x \rightarrow\rightarrow y$ in R is called a trivial MVDs. if (*a*) *y* is a subset of X or (*b*) $X \cup Y = R$

Ans. (*i*) **Thomas write rule** makes use of view serializable by, in effect, deleting obsolete write operations from the transactions that issue them.

(*j*) **Ans:** %ISOPEN. always returns FALSE for implicit cursors, because Oracle closes the SQL cursor automatically after executing its assosiated SQL statement

Q.2. (*a*) Given a relation R = (A, B, C, D, E) with the following FD's : {CE → D, D → B, C → A}. **(7)**

(*i*) Find all candidate keys.

(*ii*) Identify the best Normal Form that R satisfies (1NF, 2NF, 3NF, or BCNF)]

(*iii*) If the relation is not is BCNF decompose it until it becomes BCNF.

Ans. (*i*) Candidate keys: CE, CED, BCE, ACE

(*ii*) *R* is in 2NF but not 3NF.

(*iii*) Both D → B and C → A cause BCNF violations. The decomposition: BD, AC, AB (obtaining) by first decomposing to AC, ABC) is BCNF and losslers and join-preserving.

Q.2. (*b*) Write down the syntax for creating a trigger and explain it. **(3)**

Ans. The syntax for creating a trigger is:

Syntax:

CREATE [OR REPLACE] TRIGGER trigger-na

me

{BEFORE/AFTER/ INSTEAD OF} {INSERT (OR)/ UPDATE [OR]}/ DELETE} [OF col-name] ON table-name

[REFERENCING OLD AS O NEW AS N][FOR EACH ROW]

WHEN (Condition)

DECLARE

Declaration- statements

BEGIN Executable - statements

EXCEPTION

Exception - handling-statements

END;

where

⇒ (CREATE [OR REPLACE] TRIGGER trigger-name: create or replaces an existing trigger with the trigger-name.

⇒ {BEFORE/AFTER/INSEAD OF}: This specifies when the trigger would be exceuted. The INSEAD of clause is used for creating trigger on a view.

⇒ {INSERT [OR]/UPDATE [OR]/DELETE}: This specifies the DML operation.

⇒ [OF Col-name]: This specifies the column name that would be updated.

⇒ [ON table-name]: This specifies the name of the table associated with the trigger.

⇒ [REFERNCING OLD AS O NEW AS n]: This allows you to refer new and old values for various DML statements, like INSERT, UPDATE and DELETE.

⇒ [For EACH ROW]: This specifies a row level trigger, *i.e* the trigger would be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.

⇒ WHEN (condition): This provides a condition for rows for which the trigger would fire. This clause is valid only for row level triggers.

Q.3. (*a*) Explain the two deadlock prevention strategies. Which of the two strategies have fewer rollbacks. **(7)**

Ans. Techniques Used: If a transaction, Ti requests a lock and the transaction, Tj, holding a conflicting lock, the lock manager can use one of the following techniques:

(*i*) wait-die.

(*ii*) wound - wait.

Wait Die Scheme: If Ti has higher priority, it is allowed to wait else it is aborted. This means that when Ti request for a data-item that is currently being held by T_j, then T_i is allowed to wait only if it has a timestamp smaller than that of T_j is older than T_j otherwise, T_i is rolled back (dies).

For Example: Say, T_1 requests for a data item held by T_2 and T_3 also requests for data item held by T_2 i.e.

$$T_1 \text{ requests} \longrightarrow \boxed{T_2} \longleftarrow T_3 \text{ request}$$

Wound-Wait Method: If Ti transaction has a higher priority then we abort Tj else Ti wais. It means that when Ti requests a data-item currently held by Tj, then Ti is allowed to wait only if it has a Ts (Ti) > Ts (Tj). Else Tj is rolled back.

For Example: If T1 requests a data-item held by T2 then the data-item will be presnpted from T2 and T2 will be rolled back i.e.,

T1 → T2 (Rollbacks)

Scanned by CamScanner

Now, if T3 requests a data-item held by T2, then T3 waits

T2 ← T3 (waits)

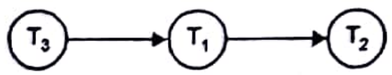
Wait-die Scheme	Wound-Wait Scheme
(*i*) This scheme is Non-Preemptive (*ii*) Lower priority transaction can never wait for higher transactions.	(*i*) This scheme is Preemptive. (*ii*) The higher priority transaction never wait for lower priority transaction.

Q.3. (*b*) Consider the schedule S. Draw the serializability graph for *S* and state whether schedule is seriaizable or not. If schedule is seralizable, write down the equivalent serial schedule. (3)

S:R1(X), R2(Z), R1(Z) R3(X), R3 (Y), W1(X), W3 (Y), R2 (Y) W2(Z), W2(Y)

Ans: It is serializable schedule because

- T1 only reads X (R1(X)), which is not modified either by T2 or T3.
- T3 reads X (R3(X)) before T1 modifies it (W1(X).
- T2 reads Y (R2(Y)) and writes it (W2 (Y)) only after T3 has written to it (W3(Y).
- Thus, the serializability graph is

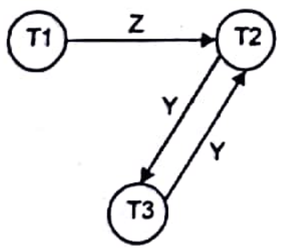


Schedule is not a serializable schedule because

- T2 reads *Y* (R2(Y)), which is then read and modified by T3 (W3(Y))
- T3 reads *Y* (R3 (Y)), which then modified before T2 modifies Y (W2(Y))

T2 modifies Y(W2(Y))

In the above order T3 interferes in the execution of T2, which makes the schedule nonserializable.



Q.4. (*a*) Given the following set of FD's for a relation R (A, B, C, D, E, F), E = {AB → C, DC → AE, E → F}.

Is the decomposition R1 (A, B , C, D) and R2 (B,C,D,E,F) a dependency preserving decomposition? Explain your answer. (7)

Ans. (*i*) F1 the restriction of F to (A, B, C, D) is:

A → BCD	AB → D
A → ABCD	AB → ABC
A → AB	AB → AB
A → CD	AB → A
A → ABC	AB → B
A → B	AB → C
A → C	DC → A
A → A	

B → B
C → C
D → D
AB → C

(*ii*) F_2, the restriction of F to (B, C, D, E, F) is:-

DC → E	E → F
BC → DEF	E → BCDEF
BC → EF	E → CD
BC → D	E → EF
B → CDEF	.
B → CD	.
B → EF	.
B → C	.
B → B	.

Now, $(F_1 \cup F_2)^+$ is easily seen to contains all set of FD's which has given in question. So, set of all FD's is dependency preserving decomposition.

Q.4. (*b*) When a transaction is rolled back under timestamp ordering; it is assigned a new timestamp. Why can it not simply keep its old timestamp. (3)

Ans. A transaction is rolled back because a newer transaction has read or written the data which it was supposed to write. If the rolled back transaction is reintroduced with the same timestamp, the same reason for rollback is still valied and the transaction will have be rolled back again. This will contiune indefinitely.

If a rollback on a transaction occurs, when that transaction is resubmitted, it 's a new transaction. Any automatic processes are going to recognise that and apply new values. If you want a specific value, you should generate it yourself using a different function and then reapply it with the new transaction.

Scanned by CamScanner

END TERM EXAMINATION [JUNE 2015]
FOURTH SEMESTER [B.TECH]
DATABASE MANAGEMENT SYSTEM
[ETCS-208]

Time. 3 Hours **MM: 75**

Note: *Attempt any five questions including Q.No. 1 which is compulsory.*

Q.1. Answer the follwoing question in brief:

Q.1. (*a*) List few application of database Management system. (2.5×10=25)

Ans. Databases are used to hold administrative information and more specialized data, such as engineering data or economic models. Some of the applications are:

- Computerized library system.
- Flight reservation systems.
- Computerized parts inventory system.
- Many content management systems that store websites as collections of webpages in a database.

Q.1. (*b*) Use Armstrong's axioms to prove the soundeness of the pesudo transtivity rule.

Ans. If $\alpha \rightarrow \beta$ holds and $\gamma\ \beta \rightarrow \delta$ holds. then $\alpha\gamma \rightarrow \delta$ holds.

Q.1. (*c*) Write down the functions of DBA.

Ans. The main functions of a DBA are as follows:

- Defining Conceptual schema.
- Physical database design.
- Security and integrity checks.
- Give backup and recovery strategies.
- Granting access to users.

Q.1. (*d*) Describe the principles of Mutual Exclustion?

Ans. Mutual exclustion is applied to critical sections to control the access to shared variables.

- A process that halts in its non-critical section.
- A process requiring access to a critical section not to be delayed indefinitely.
- As long as no other process is in a critical section, a process that requents entry must be permitted to enter without delay
- No assumptions are made about relative process speeds or number of proceeors.
- A process must exit its critical section in a finite amount of time.

Q.1. (*e*) What is query processor?

Ans. Query processing means a range of activites involved in extracting data form a database. It is a procedure of selecting the strategy to be used in responding to a database request. This component of DBMS is called as Query process. It is stepwise process

Q.1. (*g*) Define the term primary key with suitable example?

Ans. Refer to Q.No. (3) (a) from first Term Examination 2014.

Q.1. (*g*) What are integrity constraints?

Ans. Integrity constraints provide a way of ensuring that changes made to the database by authorised users do not result in a loss of data consistency. We saw a form of integrity constraint with E-R models, key declarations, stipulation that certain attributes form a candidate key for the entity set.

Q.1. (*h*) What are the problems caused by deadlocks?

Ans. (*i*) Deadlock can occur in Java because the synchronized keyword causes the executing thread to block while waiting for the lock.

(*ii*) Deadlocks are a common causes of performance problems.

(*iii*) Deadlocks is common probelm in multiprocessing systems, parallel processing.

Q.1. (*i*) What is a view? What are its advantages?

Ans. View: Views are defined as the logical tables of data extracted from existing tables. It can be queried just like a table but does not require any disk spack.

Advantages:

(*i*) View the data without storing the data into the object.

(*ii*) Restrict the view of a table *i.e*, can hide some of columns in the tables.

(*iii*) Join two or more tables and show it as one object to user.

(*iv*) Restrict the access of a table so that nobody can insert the rows into the table.

Q.1. (*j*) What are the various types of failures that can occur in database?

Ans. There are many different types of failure that can affect database processing. Some failures affect main memory only, while others involve non-valitle (secondary) storage. Among the causes of failure are:

→ System Crashes
→ User Error
→ Carelessness
→ Sabotage (international corruption of data)
→ Statement Failure
→ Application Software Errors
→ Network Failure
→ Media Failure
→ Natural Physical Disasters:

Q.2. (*a*) Give the relational schemas: (2.5×4=10)

ENROLL (s#, c#, section)-s# represents student number
TEACH (prof, c#, section) - c# representes course number
ADVICE (prof, s#) - prof is thesis advisor of s#
PRE-REQ (c#, pre_c#) - Pre_c# is prerequisite course
GRADES (S#, C#, grade, year)
STUDENT (s#, sname) - sname is student name.

Give queries expressed in relational algebra or tuple calculs for the following algebra:

(*i*) List all students taking courses with Smith or Jones.

(*ii*) List all students taking at least one course that their advisor teaches.

Scanned by CamScanner

(iii) List those professors who teach more than one section of the same course.

(iv) List the courses that student 'John' can enroll i.e. has passed the neccessary pre-requisite courses but not the course itself.

Ans. (i) π_{sname} (°pre_c# = 'Smith' v 'Jones' (Pre_REQ ⋈ ENROLL ⋈ STUDENT)

(ii) πsname (σprof > 0 (TEACH ⋈ ENROLL ⋈ STUDENT)

(iii) π prof (TEACH ⋈ ADVICE ⋈ ENROLL)

(iv) °sname = 'John' (PRE_REQ ⋈ STUDENT ⋈ ENROLL)

Q.2. (b) Explain the normal forms due to functional dependencies. (2.5)

Ans. First Normal Form (1NF): A relation is said to be in 1NF if and only every entry of the relation has at most a single (or atomic) value only.

Second Normal Form (2NF): A relation R in second Normal Form (2NF), , if and only if it is 1NF and every non-key attribute is fully dependent on the primary key.

Third Normal Form (3NF): A relation R is in Third Normal Form (3NF), if and only if it is 2NF and no non-prime atttribute is transitively dependent on the key or we can say that non-prime attribute functionally determines any other non-prime attributers.

Fourth Normal Form (4 NF): A relation R is in Fourth Normal Form (4NF), if and only if it is in 3NF or BCNF and it contains no multi-valued dependencies.

Fifth Normal Form (SNF): A relation R is in fifth Normal Form (5NF), if and only if it is in 4NF and it cannot be further non-loss decomposed.

Q.3. (a) What is functional dependency? Write down the rules for finding implied functional dependent from a set of given FDs. (7.5)

Ans. Functional Dependency (FD): A functional dependency is an assosiation between two attributes of the same table (relation). One of the attribute is called as the determinant and other attribute is called as the determined.

Note: For each value of the determinant there is assosiated one and only one value of the determined.

A	B
10	70
11	80
12	90
12	90
13	95

Rules:

(i) Reflexivity Rule: If α is a set of attributes and $\beta \le \alpha$ then $\alpha \to \beta$ holds.

(ii) Augmentation Rule: If $\alpha \to \beta$ holds, and w is a set of attributes, then $v\,\alpha \to v\beta$ holds.

(iii) Transitivity Rule: If $\alpha \to \beta$ holds and $\beta \to v$ holds then $\alpha \to v$ holds.

(iv) Complementation Rule: If $\alpha \to \beta$ holds, then $\alpha \to\to R - \beta - \alpha$ holds

(v) Multivalued transtivity rule: If $\alpha \to\to \beta$ holds and $\beta \to\to v$ holds then $\alpha \to\to v$ holds then $\alpha \to\to v - \beta$ holds.

(vi) Replication Rule: If $\alpha \to \beta$ holds then $\alpha \to\to \beta$.

(vii) Coalescence Rule: If $\alpha \to\to \beta$ holds and $v \le \beta$ and there is a s such that $\delta \le R$ and $\delta \cap \beta = \phi$ and $\delta \to v$ then $\alpha \to v$ holds.

(viii) Intersection Rule: If $\alpha \to\to \beta$ holds and $\alpha \to\to v$ holds then $\alpha \to\to \beta \cap v$ holds.

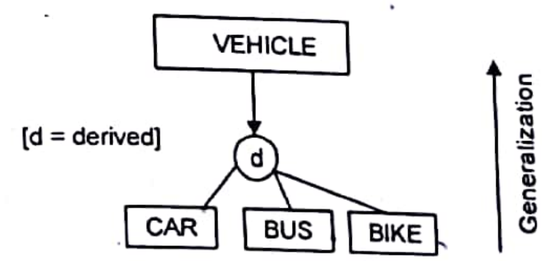
(ix) Difference Rule: If $\alpha \to\to \beta$ holds and $\alpha \to\to v$ holds then $\alpha \to\to \beta - v$ holds and $\alpha \to\to v - \beta$ holds.

Q.3. (b) Explain Generalization and serialization with suitable example. (5)

Ans. Generalization: It is defined as the process of identifying some common characterstics of a collection of entiry sets and creating a new entity set that contains entities processing these common features. It can also be defined as the process of minimising the differences between entities by identifying their common charactertics.

Generalization is a bottom up process *i.e.* say we have three sub-entities as car, Bus Bike Now, these three sub-entities can be generalized into one general super class (or base class) named as VEHICLE.

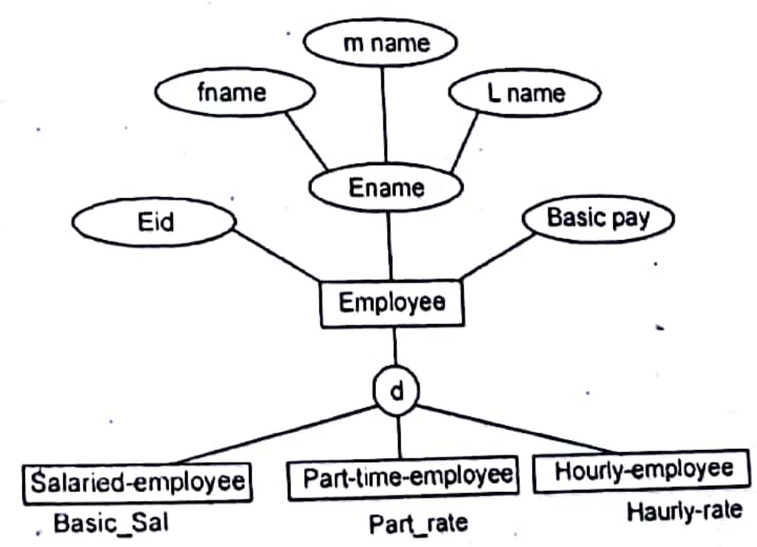
Example:



Specialization: It is defined as the process of identifying subsets of an entity (super-class or super-type that share some distinguishing characterstic. It is the process of taking subsets of a higher Level entity set to form lower entity sets and is actually specialization.

It is a top-down process of defininfg super-classes and their related sub-classes. Also note that we firstly define a super-class, then sub-classes and then their attributes and relationships. The advantage of doing generalizations or specialzations is that now each of these entities or sub-entities can have the attributes of their own.

Example:



Scanned by CamScanner

Q.4. (a) Compute the closure of the following set & of functional depencies for relation schema R = (A, B, C, D, E) **(6)**

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

List the candidate keys for R.

Ans. Given: $A \rightarrow BC$, So $A \rightarrow B$ and $A \rightarrow C$

Since $A \rightarrow B$ and $B \rightarrow D$, So $A \rightarrow D$ (Transitivity rule)

Now, $A \rightarrow CD$ and $CD \rightarrow E$, $A \rightarrow E$ (Union, decomposition, transitivity rule)

Since $A \rightarrow A$, we have (reflexity Rule)

$A \rightarrow ABCDE$ (UNION Rule)

Since, $E \rightarrow A$, $E \rightarrow ABCDE$ (Transitivty Rule)

Since $CD \rightarrow E$, So, $CD \rightarrow ABCDE$ (Transitivity Rule)

Since, $B \rightarrow D$ and $BC \rightarrow CD$

So, $BC \rightarrow ABCDE$ (Aujumentation Rule & Transitivity)

Also, $C \rightarrow C$, $D \rightarrow D \rightarrow D$, $BD \rightarrow D$ and so on Therefore, we find that any FD with A E, BC or CD on the L.H.S. of arrow is in F^+, in the FD. Say*, represents any set of attributes in R, then F^+ is

$$BD \rightarrow B$$
$$BD \rightarrow D$$
$$C \rightarrow C$$
$$D \rightarrow D$$
$$BD \rightarrow BD$$
$$B \rightarrow D$$
$$B \rightarrow B$$
$$B \rightarrow BD$$

and all FDs of the form

$$A^* \rightarrow \propto$$
$$BC^{**} \rightarrow \propto$$
$$CD^* \rightarrow \propto$$
$$E^* \rightarrow \propto$$

Where $\propto$ is any subset of (A, B, C, D, E).

The candidates keys are A, B, C, CD and E.

Q.4. (b) Using the FDs of above Fig 1, Find the canonical cover, Fc? **(6.5)**

Ans. Given: Set of FDs, F is

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

Here, L.H.S. (determinant) of all FDs is unique. Also none of the attributes in the left or right side of any FDS is extraneous. Therefore, the canonical cover Fc is equal to F.

Q.5. Draw an E.R diagram for a garment manufacuring company. The entities include warehouse, production units, marketing wing vendor and product types. Define the relationships between each of these entities. Choose the attribute so that they define a particular entity properly. **(12.5)**

Ans. Refer to Q.No. (2) (a) from First Term Examination, 2014.

Q.6. (a) What is the difference between serial schedule and Serializable schedule? **(5)**

Ans. Refer to Q.No. (1) (d) from second Term Examination, 2013.

Q.6. (b) Formulate a concurrency control protocol based on read and write locks. **(7.5)**

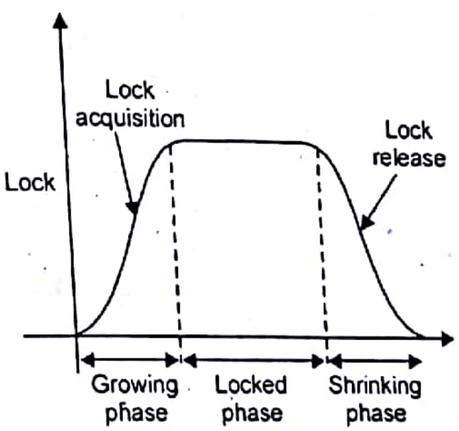
Ans. Mainly there are two protocols used i.e.

(i) Two-phase locking (2PL) protocol

(ii) Timestamp ordering protocol.

(i) Two-phase Locking Protocol: If all locking operations precede the first unlock operation in the transaction then a transaction follows the two phase locking protocol. 2PL has two phases in a given schedule.

Phase-1: Growing Phase: A phase during which all locks are requested.



2PL scheme

(ii) Timestamp ordering Protocol: This protocol works on a principle that - "We must have a prior knoweledge about the order in which the transactions will be accessed". In graph based protocol, we require the knowledge of ordering of data items in the transactions.

Timestamp have two properties:

(a) Uniqueness: This property assures that no equal timetamp values can exist.

(b) Monotonicity: It means that timestamp values always increase.

Q.7. (a) What is timestamp? Discuss the working of wait-die and wound-wait transaction. **(8)**

Ans. Refer to Q.NO. (4) (a), Second Term Examination, 2013 for Defination of timestamp and Q.NO. (1) (c) Second Term Examination, 2013 for working of wait-die and wound-wait transaction.

Q.7. (b) Compare Natural Join with Outer Join with suitable example? **(4.5)**

Ans. Natural Join: Natural join between two relations is written as ($R \bowtie S$). The result of the natural join is the set of all combinations of tuples in R and S that are equal on their common attribute names.

Scanned by CamScanner

Outer Join: An outer join operations returns all the rows returned by an inner join plus all the rows from one table that do not match any row form the other table. An outer join can be left, right or both tables are retained.

For Example: Consider the tables Employee and department and their Natural, Outer Join.

Department

Dept Id	Dept Name
d-01	Finance
d-02	Sales
d-03	Marketing

Employee

Name	Emp Id	Dept Id
Khanak	e-01	d-01
Kavita	e-02	d-03
Alpesh	e-03	d-01
Aruna	e-04	d-02

Result of Natural Join: Employee ⋈ Department

Name	Emp Id	Dept Id	Dept Name
Khank	e-01	d-01	Finance
Kavita	e-02	d-02	Sales
Alpesh	e-03	d-01	Finance
Aruna	e-04	d-02	Sales

Result of Outer Join: Employee ⋈ Department

Name	Emp Id	Dept Id	Dept Name
Khanak	e-01	d-01	Finance
Kavita	e-02	d-02	Sales
Alpesh	e-03	d-01	Finance
Aruna	e-04	d-02	Sales
NULL	NULL	d-03	Marketing

Q.8. Explain the follwoing terms: **(2.5×5=12.5)**

Q.8. (*a*) Fourth Normal Form.

Ans. Fourth Normal Form: A relation R is in Fourth Normal Form (4NF) if and only if the following conditions are satsified simultaneously:

(*i*) R is already in 3NF or BCNF

(*ii*) If it contains no multi-valued dependencies.

Example: Consider the following employee table. Suppose that employees can be assigned to miltiple projects. Also assume that employees can have multiple job skills.

Eno	Pno	Skill
A1	100	coding
	109	Testing
		maintence

Try to normalize this table

Sol: Firstly, we flatten the above table-emmp, in order to normalize it.

Eno	Pno	Skill
A1	100	Coding
A1	100	Testing
A1	100	Maintenance
A1	109	Coding
A1	109	Testing
A1	109	Maintenance

This table has lot of redundacies. It contains MVDs also, we need to remove them. And this can be done by decomposing it. Now, the MVDs are

(*i*) Eno →→ Pno

(*ii*) Eno →→ skill

so, we decompose now

ERPROJECT (Eno, Pno)

Eskill (Eno, skill)

Q.8. (*b*) DML

Ans. DML: It is used to manipulate data in the database. Some commands are: INSERT, DELETE, UPDATE and SELECT

Consider the database to add data to the fields interactively we write an SQL as follows:

INSERT INTO CUSTOMER VALUES
(& CUST_NO, '& LNAME', '&FNAME');

The names need not match the column names but the type and order is important.

Q.8. (*c*) Explain derived attributes?

Ans. It is defined as an attribute that represent a value that is derivable from the value of related attribute or set of attributes, not essentially in the same entity. So, the values held by some attributes are deriveds from two or more attribute values. For example, the age attribute can be derived from the date-of birth attribute and so they are related.

Q.8. (*d*) Triggers.

Ans: Triggers: A trigger consists of PL/SQL code which defines some action that the database should take when some database related event occurs. Triggers are executed when a data manipulation command (DML) is performed on a table for which the trigger has been written.

Triggers are executed automatically and are transparent to the user. Also note that in order to create a trigger, the user must have the CREATE TRIGGER privilege, This privilege is the part of the RESOURCE role.

Scanned by CamScanner

Triggers are required:

(*a*) To enforce rules that cannot be coded through referential integretiy.

(*b*) To enforce certain bussiness rules (validation) in an application.

(*c*) To maintain an audit of changes made of data.

(*d*) To derive column values.

Q.8. (*e*) Indexing

Ans. Indexes in SQL: An index is defined as a table or data structure that is maintained to determine the location of rows or records in a file that satisfy some condition. Such a table is known as an index table which consist of a key field and a pointer to the location where the record is stored. Each index entry corresponds to a data record on the secondary storage disk device. Firstly, this index table is searched for the address of the record in it and then reading of the record form this address.

For Example: The index of this book workds in the same way as that of an index for a file in a database system. Another example could be a library catalogue.

Indexing is of two types:

(*a*) Ordered indexing

(*b*) Hashed indexing.

(*a*) Ordered indexing: It is based on a sorted orderings of the values of records.

(*b*) Hashed indexing: It is based on the values of record being uniformly distributed using a fast mathematical function.

FIRST TERM EXAMINATION [FEB.–2016]
FOURTH SEMESTER [B.TECH]
DATABASE MANAGEMENT SYSTEM [ETCS–208]

Time : 1.30 Hrs. **M.M. : 30**

Note: Q. No. 1 is compulsory and attempt any two more questions from the remaining.

Q.1. (*a*) Differentiate between Procedural and Non-procedural Query Language. **(2)**

Ans.

Procedural query language:	Non procedural query language
What to retrieve from database and how to retrieve. Example : Relational Algebra. $\pi, \times, \sigma$, etc.	What to retrieve from database. For example : Tuple Relational Calculus. Use First Order Logic $\in, \exists, \wedge, \vee, \neg, \rightarrow$ etc.

Q.1. (*b*) Define Lattice in ER-Diagram. **(2)**

Ans. • In a lattice or hierarchy, a subclass inherits attributes not only of its direct superclass, but also of all its predecessor superclasses.

• A subclass with more than one superclass is called a shared subclass.

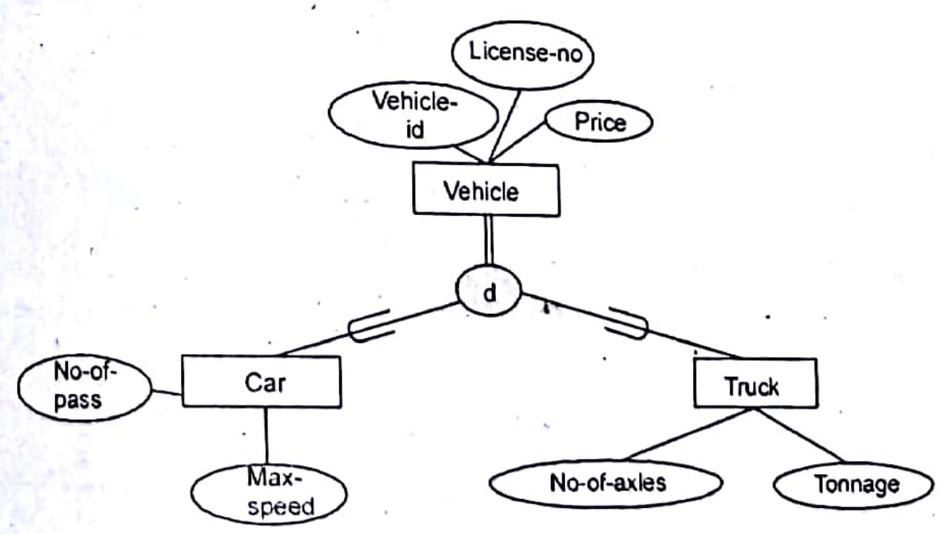
• Can have specialization hierarchies or lattices, or generalization, hierarchies or lattices.

• In specialization, start with an entity type and then define subclass of the entity type by successive specialization (top down coneptual refinement process).

• In generalization, start with many entity types and generalize those that have common properties (bottom up conceptual synthesis process).

• In practice, the combination of two processes is employed.

Example of lattice:



Q.1.(*c*) Define the concept of Aggregation with an example. **(2)**

Ans. Aggregation: Aggregation is the process of compiling information on an object, thereby abstracting a higher-level object sometimes we want to model a 'has-a' or 'is-part-of_relationship in which one entity represents a larger entity consisting of smaller enties. This special kind of relationship is called as an aggregation..

Scanned by CamScanner

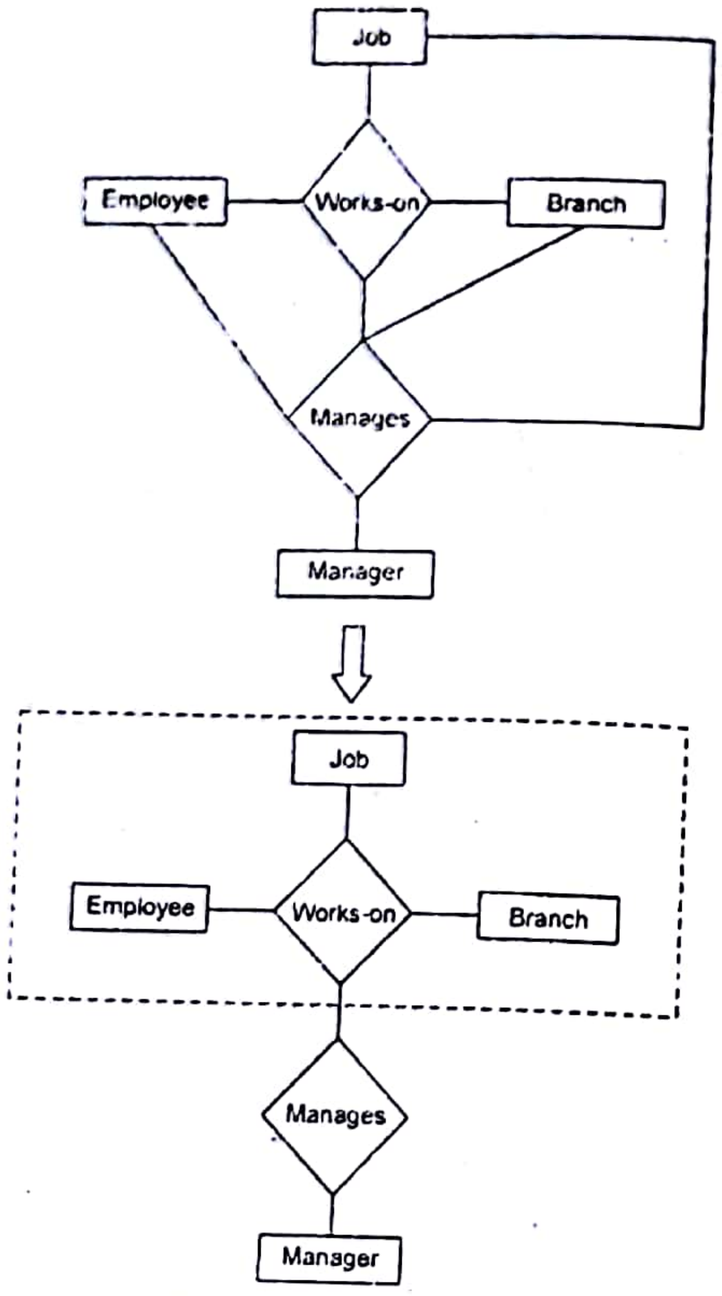


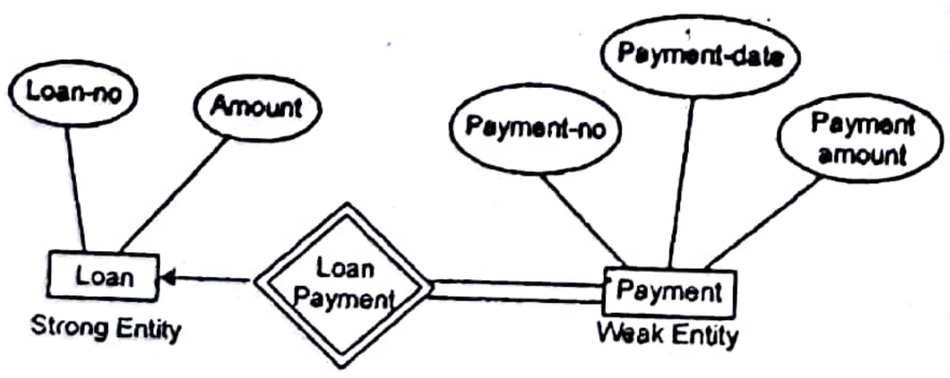
Fig. Aggregation concept.

Q.1.(*d*). How to represent weak entity set. (2

Ans. Weak Entity Sets:

(*i*) The entity set which does not have sufficient attribute to form a primary key.

(*ii*) Each weak set must be a part of 1 : M relationship set.



(*iii*) A member of weak entity set is called as a subordinate entity.

(*iv*) A weak entity set is represented by double outlined box and the corresponding identifying relation by a doubly outlined diamond.

Q.1.(*e*). Define view with the help of example. (2)

Ans. A view is nothing more than a SQL statement that is stored in the database with an associated name. A view is actually a composition of a table in the form of a predefined SQL query.

A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depends on the written SQL query to create a view.

Views, which are kind of virtual tables, allow users to do the following:

- Structure data in a way that users or classes of users find natural or intuitive.
- Restrict access to the data such that a user can see and (sometimes) modify exactly what they need and no more.
- Summarize data from various tables which can be used to generate reports.

Example:

Consider the CUSTOMERS table having the following records:

```
+-----+----------+-------+-----------+----------+
| ID | NAME     | AGE | ADDRESS | SALARY |
+-----+----------+-------+-----------+----------+
| 1  | Ramesh   | 32 | Ahmedabad | 2000.00 |
| 2  | Khilan   | 25 | Delhi     | 1500.00 |
| 3  | Kaushik  | 23 | Kota      | 2000.00 |
```

Now, following is the example to create a view from CUSTOMERS table:

```
SQL > CREATE VIEW CUSTOMERS_VIEW AS
SELECT name, age
FROM CUSTOMERS;
```

Now, you can query CUSTOMERS_VIEW in similar way as you query an actual table. Following is the example:

```
SQL > SELECT * FROM CUSTOMERS_VIEW;
```

This would produce the following result:

```
| name     | age |
+----------+-------+
| Ramesh   | 32 |
| Khilan   | 25 |
| Kaushik  | 23 |
```

Q. 2. (a) Given the relations

Loan			Borrower	
Loan_No	Branch_Name	Amount	Cust_Name	Loan_No.
L_170	ICICI	30,000	John	L–170
L-230	SBI	40,000	Smith	L–130
L_260	PNB	25,000	Adam	L–155

Scanned by CamScanner

(a) Give the relation obtained by Natural Join, Left outer join and r outer join, equi-join and theta join.

Ans. Natural join:

Loan No	Branch_Name	Amount	Cust_Name
L–170	ICICI	300,00	John

Left outer join:

Loan_No	Branch_Name	Amount	Cust_Name	Loan_No
L_170	ICICI	30,000	John	L_170
L–230	SBI	40,000	Null	Null
L–260	PNB	25,000	Null	Null

Right outer join:

Loan_No	Branch_Name	Amount	Cust_Name	Loan_No
L_170	ICICI	30,000	John	L_170
Null	Null	Null	Smith	L–130
Null	Null	Null	Adam	L–153

Full outer join (or outer join):

Loan_No	Branch_Name	Amount	Cust_Name	Loan_No
L_170	ICICI	30,000	John	L_170
L–230	SBI	40,000	Null	Null
L–260	PNB	25,000	Null	Null
Null	Null	Null	Smith	L–130
Null	Null	Null	Adam	L–155

Equi-join:

When theta john uses only equality comparison comperator. It is said to be eq join.

Loan_No	Branch_Name	Amount	Cust_Name	Loan_No
L–170	ICICI	30,000	John	L–170

Theta join

$(\text{LOAN} \bowtie_{\text{Loan loanNo=Borrower.Loan No}} \text{Borrower})$

Loan_No	Branch_Name	Amount	Cust_Name	Loan_No
L–170	ICICI	30,000	John	L_170

Q.2.(b). Given the relation Algebra for the following query

(i) Find the customer name having loan in ICICI branch.

(ii) Find the Branch Name and Loan No. with the Loan amount greater tha 10,000.

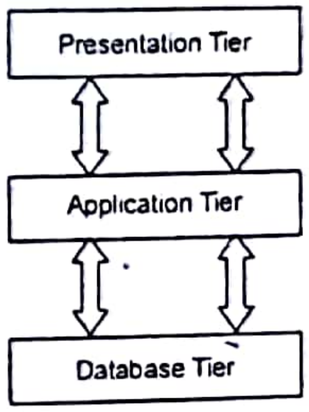
Ans. (i) $\pi_{Cust_Name}(\sigma_{Branch_Name=ICICI}(\text{Loan} \bowtie \text{Borrower}))$

(ii) $\pi_{Branch.Name,\ Loan_No}(\sigma_{Amount>10,000}(\text{Loan}))$

Q.3. (a) Explain the Three Level Architecture of DBMS. **(5)**

Ans. 3-tier architecture: Most widely used architecture is 3-tier architecture. 3-tier architecture separates it tier from each other on basis of users. It is described as follows:

- **Database (Data) Tier:** At this tier, only database resides. Database along with its query processing languages sits in layer-3 of 3-tier architecture. It also contains all relations and their constraints.
- **Application (Middle) Tier:** At this tier the application server and program, which access database, resides. For a user this application tier works as abstracted view of database. Users are unaware of any existence of database beyond application. For database-tier, application tier is the user of it. Database tier is not aware of any other user beyond application tier. This tier works as mediator between the two.
- **User (Presentation) Tier:** An end user sits on this tier. From a users aspect this tier is everything. He/she doesn't know about any existence or form of database beyond this layer. At this layer multiple views of database can be provided by the application. All views are generated by applications, which resides in application tier.



Q.3. (b) What is a key? Explain Super Key, Primary key and Foreign key. **(5)**

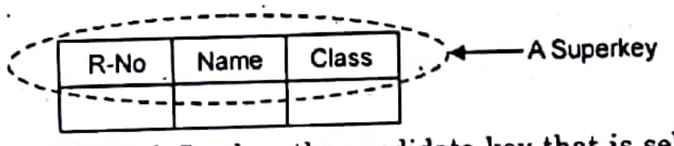
Ans. Key : are very important part of Relational database. They are used to establish and identify relation between tables. They also ensure that each record within a table can be uniquely identified by combination of one or more fields within a table.

Super Key: (*i*) A super key is defined as an attribute or a set attributes that uniquely identifies a tuple within a relation.

(*ii*) A super key has the uniqueness property bu not necessarily the inrreducibility property.

(*iii*) A candidate key is a special case of a super key,

For example:



Primary Key: (*i*) It is defined as the candidate key that is selected to identify tuples uniquely within the relation.

(*ii*) The primary key is an attribute or set of attrbutes that uniquely identify a specific instance of an entity.

(*iii*) The primary key cannot contain any NULL value because we cannot uniquely identify multiple NULL values.

Foreign Key: It says that the column in the child table that references a primary key of the parent table is called as a Foreign key. They are used to link together two or more different tablewitch have some form of relationship with each other.

For Example: Consider the following parent master table STD-ADD with the following attributes.

STD-ADD (Roll-No, Name, Place, Pin)

Now consider another transaction/child table: STD-Marks with the following attributes:

Scanned by CamScanner

STD-Marks (Roll-no, Subject, Marks, Date)

Then, Roll-no column of STU-ADD table is the primary key where as the roll-No column are not currently selected as the primary key. So can say that.

Q.4. Explain the following.

Q.4. (a) Triggers

Ans. A trigger is a block that executes implicitly whenever a particular event takes place. A trigger can be either a database trigger or an application triggers. Database triggers executes implicitly when a INSERT, UPDATE or DELETE statement is issued against the associated table, no matter which user is connected or which application is used. **(3)**

Creating Statment Triggers: The trigger action defines what need to be done when the triggering event is issued.

Syntax of Creating Staatement Triggers:

CREATE (OR REPLACE) TRIGGER trigger_name

Timing

Event 1 [OR Event 2 or Event 3]

On table-name

Trigger Body

Example: Create a trigger to restrict inserts in the EMP table to certi business hours, Monday through friday,

```
CREATE OR REPLACE TRIGGER secure-emp
BEFORE INSERT on emp
BEGIN
IF (TO-CHAR (sysdate, 'Dy') IN ('SAT', 'SNUO OR
(TO–CHAR (sysdate, 'HH24') NOT BETWEEN ('08'AND '18')
THEN RAISE – APPLICATION.ERROR (–20500, 'You may only insert into EMP
during business hours');
END IF;
END;
```

Q.4. (b) Data Definition Language

Ans. Data Definition Language (DDL) statements are used to define the database structure or schema. DDLs are used to define the metadata of the database. i.e.; using this, we create schema, tables, constraints, indexes in the database. DDLs are also used to modify Schema, tables index etc. Basically, using DDL statements, we create skeleton of the database. It helps to store the metadata information like number of schemas and tables, their names, columns in each table, indexes, constraints etc in the database. **(3)**

Commonly used DDL in SQL querying are:

- CREATE: This command builds a new table and has a predefined syntax. The CREATE statement syntax is CREATE TABLE [table name] ([column definitions]) [table parameters]. CREATE TABLE Employee (Employee Id INTEGER PRIMARY KEY, First name CHAR (50) NULL, Last name CHAR (75) NOT NULL).
- ALTER: An alter command modifies an existing database table. This command can add up additional column, drop existing columns and even change the data type of columns involved in a database table. An alter command syntax is ALTER object type object name parameters. ALTER TABLE Employee ADD DOB Date.

- DROP: A drop command deletes a table, index or view. Drop statement syntax is DROP object type object name. DROP TABLE Employee.
- TRUNCATE – remove all records from a table, including all spaces allocated for the records are removed.
- COMMENT – add comments to the data dictionary.
- RENAME – rename an object.

Q.4.(c) (i) Set of Relations are

USER_SCHEME (Card No, B_Name, B_addr)

SUPPLIER_SCHEME (Acc_No., S_Name, price, DOS)

BORROWER (Acc_No., Card_No. DOI)

Write the SQL Query for

(i) Find out the name and address of borrowers who have issued a look on 2/Feb/2016. **(2)**

Ans. SELECT B_Name, B_adder FROM, BORROWER

Where BORROWER. Card_No = USER_SCHEME. Card_No And

DOI = 2/Feb/2016'.

Q.4.(c) (ii) Find the name of the suppliers with maximum price. **(2)**

Ans. Select S_Name from SUPPLIER_SCHEME

WHERE PRICE IN (SELECT MAX (price) FROM SUPPLIER SCHEME).

Scanned by CamScanner

SECOND TERM EXAMINATION [APRIL–2016]
FOURTH SEMESTER [B.TECH]
DATABASE MANAGEMENT SYSTEM
[ETCS–208]

Time : 1.30 Hrs.

M.M. : 30

Note: *Q. No. 1 is compulsory and Attempt any two more questions from the remaining*

Q.1. (a) Define Normalization. (2)

Ans. Normalization of Database: Database Normalisation is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like Insertion, Update and Deletion Anamolies. It is a multi-step process that puts data into tabular form by removing duplicated data from the relation tables.

Normalization is used for mainly two purpose,

- Eliminating reduntant (useless) data.
- Ensuring data dependencies make sense i.e data is logically stored.

Problem without Normalization: Insertion. Updation and deletion Anamolies are very frequent if Database is not Normalized. To understand these anomailes let us take an example of Student table.

S_id	S_Name	S_address	Subject_opted
401	Adam	Noida	Bio
402	Alex	Panipat	Maths
403	Stuart	Jammu	Maths
404	Adam	Noida	Physics

- **Updation Anamoly:** To update address of a student who occurs twice or more than twice in a table, we will have to update **S_Address** column in all the rows, else data will become inconsistent.
- **Insertion Anamoly:** Suppose for a new admission, we have a Student id(S_id), name and address of a student but if student has not opted for any subjects yet then we have to insert **NULL** there, leading to Insertion Anamoly.
- **Deletion Anamoly:** If (S_id) 401 has only one subject and temporarily he drops it, when we delete that row, entire student record will be deleted along with it.

Normalization Rule: Normalization rule are divided into following normal form.

1. First Normal Form
2. Second Normal Form
3. Third Normal Form
4. BCNF

Q.1. (b) Differentiate between BCNF and 3 NF.

Ans.

3NF	BCNF
1. A relation R is said to be in 3NF if the following conditions are simulationeously satisfied: (*i*) R is already in 2NF. (*ii*) No non-prime attribute is transiviely dependent on the key.	1. A relation R is said to be in BCNF in (every determinant is a candidate key. Here, determinant is a simple attribute or composite attribute on which some other attribute is fully functionally dependent
2. Suppose a FD, A → B, 3NF allows this in a relation if B is a PK and A is not a candidate key.	2. BCNF insists that A → B dependency to remain in a relation, A must be a candidate key.

Q.1. (c) Define the term dirty reads.

Ans. A dirty read problem occurs when one transaction updates a database item and then the transaction fails for some reason. This problem is also known as "Un-committed Data" The dirty read is consequence of reading updates made by a transaction before it has successfully finished.

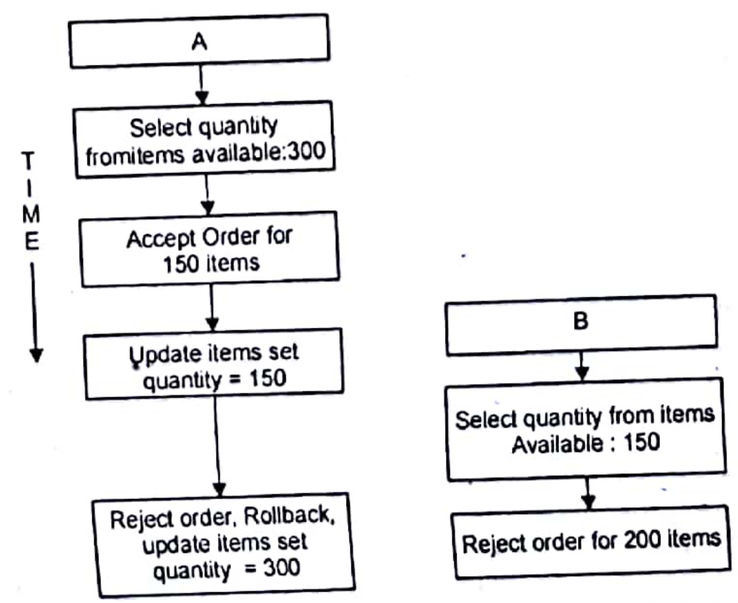
Example:

Connection A	Connection B
———	.....
———	Update Row 1
———	.....
Select Row 1	.....
———	.....
———	Roll Back

Situation:

Connection A reads an object that has been modified by connection B but connection B has not committed yet. Here uncommitted data is already read by connection A. This is known as "Dirty Read".

In a dirty read problem, another example as : A is processing an order for a client for 150 items. He checks from the ITEMS table that there are 300 available items. So he starts placing the order. A confirms the order for 150 items and updates the ITEMS table and sets the quantity to 150. Now, B receives an order for 200 items. He checks the ITEMS table to find that there is not enough inventories (150 available) and rejects the order. By using business rules and transactions a note is sent informing that more items are required. Now, due to some reason client asks A to cancel the order, so A cancels the order, rolls back and updates the ITEMS table back to 300 items. This problem is known as dirty read because B saw the uncommitted update of A.



Q.1. (d) What benefit does strict two-pase locking provide? What disadvantages result?

Ans. Two-phase (2 PL) Locking protocol: If all locking operations precede the first unclock operation in the transaction then a transaction follows the two phase locking protocol. 2PL has two phases in a given schedule:

(*a*) **Growing Phase:** A phase during which all locks are requested.

(*b*) **Shrinking Phase:** A phase during which all locks are released.

Scanned by CamScanner

Rules for 2PL:

1. Two transactions cannot have conflicting locks.
2. No unlock operation can precede a lock operation in the same transaction.
3. No data are affected untill all locks are obtained,

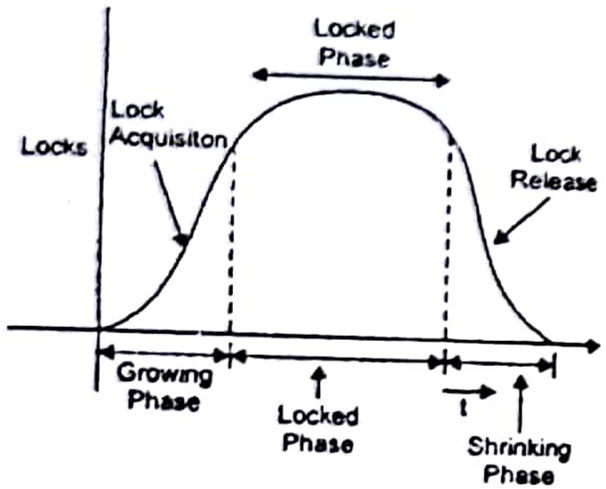


Fig. PL Scheme.

Note: As per 2PL protocol, the transaction cannot acquire a new lock after it has unlocked any of its existing locked items.

Limitations of 2PL protocol: There are two problems associated with tendency) 2PL protocols:

1. Deadlocks **2. Cascading Rollback**

Deadlocks: 2PL protocls do not prevent deadlocks. The transaction, T_1 is in two phase but still has a problem of deadlock. This is because if we do not unlock a data item, before requesting a lock on another data item, then deadlocks may occur.

Cascading Rollback; Cascading rollback is a phenomenon in which a single transaction failure leads to a series of transaction rollabck:

T_1	T_2	T_3
Read (X); Read (Y); Z = X + Y; Write (Z);		
	Read (Z); Write (Z);	
		Read (Z)

Here, the transaction, T_1 writes a value of Z that is read by transaction, T_3, T_2 then writes a value of Z which is read by T_3. Suppose now that at this point, T_1 fails. So, T_1 must be rolled back. Since T_2 depends on T_1 S_0 T_2 must be rolled back. Since T_3 is dependent on T_2, so T_3 also must be cascading Rollback Effect.)

Q.1. (e) Explain the term shadow-paging. **(2)**

Ans. Shadow paging recovery scheme does not require the use of a log in a single-user environment. In a multiuser environment, a log may be needed for the concurrency control method. Shadow paging considers the database to be made up of a number of fixed-size disk pages (or disk blocks)—say, *n*—for recovery purposes. A directory with *n* entries is constructed, where the ith entry points to the ith database page on disk. The directory is kept in main memory if it is not too large, and all references—reads or writes—to database pages on disk go through it. When a transaction begins executing, the current directory—whose entries point to the most recent or current database pages on disk—is copied into a shadow directory. The shadow directory is then saved on disk while the current directory is used by the transaction.

During transaction execution, the shadow directory is *never* modified. When a write_item operation is performed, a new copy of the modified database page is created, but the old copy of that page is *not overwritten*. Instead, the new page is written elsewhere—on some previously unused disk block. The current directory entry is modified to point to the new disk block, whereas the shadow directory is not modified and continues to point to the old unmodified disk block. For pages updated by the transaction, two versions are kept. The old version is referenced by the shadow directory, and the new version by the current directory.

To recover from a failure during transaction execution, it is sufficient to free the modified database pages and to discard the current directory. The state of the database before transaction execution is available through the shadow directory, and that state is recovered by reinstating the shadow directory. The database thus is returned to its state prior, to the transaction that was executing when the crash occurred, and any modified pages are discarded. Committing a transaction corresponds to discarding the previous shadow directory. Since recovery involves neither undoing nor redoing data items, this technique can be categorized as a NO-UNDO/NO-REDO technique for recovery.

Q.2. A relation NADDR is defined as follows:

NADDR = (name, street, city, state, postal_code) where name is unique, and for any given postal code, there is just one city and state.

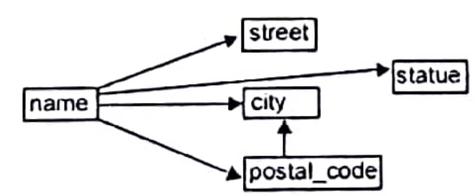
(a) Given a set of FDs for this relation. **(2.5)**

Q.2. (b) What are the candidate keys?

Q.2. (c) Is NADDR in 3NF? 2NF? Explain why?

Q.2. (d) If NADDR is not in 3NF, normalize it into 3NF relations. **(2.5)**

Ans. (a)



(b) The candidate key is Name.

(c) It is in 2NF as there is dependency between nonkey attributes as postal-code determines city and state.

(d) NADDR = (name, street, city, state, postal_code) will be decomposed into NADDRI = (name, street, postal_code)

Postal = (postal_code, city, state)

Now after decomposition NADDR1 and postal are in 3nf.

Q.3. (a) Explain Timestamp based and validation based protocols. **(5)**

Ans. Timpestamp: This protocol works on a principle that -"we must have a prior knowledge about the order in which the transactions will be accessed". In graph based protocol, we require the knowledge of ordering of data items in the transactions.

Scanned by CamScanner

How to do it?

To order the transactions, we assosiate a unique fixed timestamp, denoted by TS(Ti) for each transaction, Ti. This time stamp is an integer (unique) value. It is assigned by DBMS before the transaction Ti, starts execution.

Timestampes nave 2 properties:

1. Uniqueness: This property assures that no equal timestamp values can exist.

2. Monotonicity: It means that timestamp values always increase.

Timestamp-Ordering Protocol: It ensures that any conflicting read and write operations are executed in timestamp order. This protocol operates as follows.

1. Suppose that transaction Ti issue read (*Q*):

(a) If TS (Ti) < w-timestamp (Q), then Ti needs to read a value of *Q* that was already overwritten. Hence the read operation is rejected, and Ti is rolled back.

(b) If TS (Ti) w-timestamp (Q), then the read operation is executed, and R-timestamp (Q) is set to the maximum of R-timestamp (Q) i.e. TS (Ti).

2. Suppose that transaction Ti issues write (Q):

(a) If Ts (Ti) < R-timestamp (Q), then the value of Q that Ti is producing was needed previously and the system assumed that value would never be produced. Hence, the write operation is rejected, and Ti is rolled back.

(6) If TS (Ti) < w-timestamp (Q), then Ti is attempting to write an absolute value of Q, Hence the write operation is rejected and Ti is rolled back.

Validation Based Protocols:

We assume that each transaction *Ti* executes in two or three different phases in its lifetime, depending on whether it is a read-only or an update transaction. The phases are, in order.

1. Read phase. During this phase, the system executes transaction *Ti*. It reads the values of the various data items and stores them in variables local to *Ti*. It performs all write operations on temporary local variables, without updates of the actual database.

2. Validation phase. Transaction *Ti* performs a validation test to determine whether it can copy to the database the temporary local variables that hold the results of write operations without causing a violation of serializability.

3. Write phase. If transaction *Ti* succeeds in validation (step 2), then the system applies the actual updates to the database. Otherwise, the system rolls back *Ti*.

Each transaction must go through the three phases in the order shown. However, all three phases of concurrently executing transactions can be interleaved.

To perform the validation test, we need to know when the various phases of transactions *Ti* took place. We shall, therefore, associate three different timestamps with transaction *Ti*:

1. Start(*Ti*), the time when *Ti* started its execution.

2. Validation(*Ti*), the time when *Ti* Finished its read phase and started its validation phase.

3. Finish(*Ti*), the time when *Ti* finished its write phase.

We determine the serializability order by the timestamp-ordering technique, using the value of the timestamp Validation(*Ti*). Thus, the value TS(*Ti*) = Validation(*Ti*) and, if TS(*Tj*) < TS(*Tk*), then any produced schedule must be equivalent to a serial schedule in which transaction *Tj* appears before transaction *Tk* . The reason we have chosen Validation(*Ti*), rather than Start(*Ti*), as the timestamp of transaction *Ti* is that we can expect faster response time provided that conseict rates among transactions are indeed low.

The validation test for transaction *Tj* requires that, for all transactions *Ti* with TS(*Ti*) < TS(*Tj*), one of the following two conditions must hold:

1. Finish(*Ti*) < Start(*Tj*). Since *Ti* completes its execution before *Tj* started, the serializability order is indeed maintained.

2. The set of data items written by *Ti* does not intersect with the set of data items read by *Tj* , and *Ti* completes its write phase before *Tj* starts its validation phase (Start(*Tj*) < Finish(*Ti*) < Validation(*Tj*)). This condition ensures that the writes of *Ti* and *Tj* do not overlap. Since the writes of *Ti* do not affect the read of *Tj* , and since *Tj* cannot affect the read of *Ti*, the serializability order is indeed maintained.

T_{14}	T_{15}
read(B)	
	read(B)
	B:=–50
	reads (A)
read(A)	A:=A+|30
<validate>	
display(A+B)	
	<validate>
	wnte (B)
	write(A)

Figure. Schedule 5,a schedule produced by using validation.

Consider again transactions *T*14 and *T*15. Suppose that TS(*T*14) < TS(*T*15). Then, the validation phase succeeds in the schedule 5 in Figure. Note that the writes to the actual variables are performed only after the validation phase of *T*15. Thus, *T*14 reads the old values of *B* and *A*, and this schedule is serializable.

The validation scheme automatically guards against cascading rollbacks, since the actual writes take place only after the transaction issuing the write has committed.

Q.3.(*b*)What is the need for log file? When is the log file used and how? (5)

Ans. Log-Based Recovery: It is the most common structure used for recording any modifications made to the database. In this recovery procedure, a log-file, is maintained. A log file is a sequence of log-records. A log records maintain a record of all operations of the database. There are four types of log record,

(*i*) <start> log record (*ii*) <update> log record

(*iii*) <commit> log record (*iv*) <Abort> log record.

(*ii*) <Start> log record: It contains information about the start of each transaction. It has a transaction identifier, which is always unique. It's syntax is

<Ti, Start>

$<T_i, X_j, V_1, V_2>$

(*ii*) <Update> log Record: It describes a single database write. Its syntax is

$<T_i, X_j, V_1, V_2>$

Where T_i – Transaction identifier

X_j – Data item

V_1 – Old data-item value

V_2 – New or modified value

(*iii*) <commit> Log Record: When a transaction T_i, is successfully committed or completed, a $<T_i$, commit> log record is stored in the log file.

(*iv*) <Abort> log Record: When a transaction T_i is aborted due to any reason, a $<T_i$, Abort> log record is stored in the log file.

Whenever a transaction performs a write operation, it is necessary that the log record for that write be created before the database is modified. A log record plays a vital role during recovery process of the database.

Scanned by CamScanner

Q.4. (a) Consider two sets of functional dependencies

F1 = {A->C, AC->D,E->AD, E->H}

F2 = {A->CD, E->AH} **(5)**

Check whether they are equivalent.

Ans. To check that F1 and F2 are equivalent.

Lets assume F1 as F and F2 as G.then

- Prove that F is covered by G.

$\{A\}^+ = \{A, C, D\}$ (wrt to G). Since $\{C\} \subseteq A^+$, $A \rightarrow C$ can be inferred from G.

$\{AC\}^+ = \{A, C, D\}$ (wrt to G). Since $\{D\} \subseteq \{AC\}^+$, $AC \rightarrow D$ is covered by G.

$\{E\}^+ = \{E,A,H,C,D\}$ (wrt G). Since $\{AD\} \subseteq \{E\}^+$, $E \rightarrow AD$ is covered by G.

Since $\{H\} \subseteq \{E\}^+$, $E \rightarrow H$ is covered by G.

$$F = \{A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H\}$$

$$G = \{A \rightarrow CD, E \rightarrow AH\}$$

Prove that G is covered by F:

- $\{A\}^+ = \{A,C,D\}$ (wrt F)

Since $\{CD\} \subseteq \{A\}^+$, $A \rightarrow CD$ is covered by F.

- $\{E\}^+ = \{E, A, D, H, C\}$ (with respect to F)

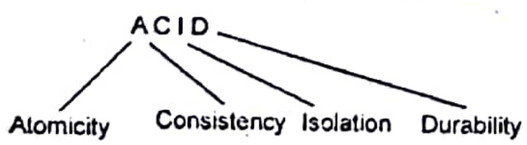
$\{A,H\} \subseteq \{E\}^+$, $E \rightarrow AH$ is covered by F.

Since F covers G and G covers F, F and G are equivalent.

So F1 and F2 are equivalent.

Q.4. (b) What are the ACID properties of transactions. Why they are essential. **(5)**

Ans. Atomictiy: A transaction is said to be atomic if a transaction always executes all of its actions in one step or not execute any actions at all.



Consistency: A transaction must preserve the consistency of a database after its execution. It means that the execution of a transaction must leave a database in either its prior stable state or new stable state.

Isolation: It means that the data used during the execution of a transaction cannot be used by a second transaction untill the first one is completed. The transactions do not interface with each other.

Durability: The effect of a completed or committed transaction should persist even after a system's failure (crash). It means that once a transaction committs, the system must guarantee that result of its operations will never be lost.

S.No.	Property	Meaning
1.	Atomicity	All or nothing
2.	Consistency	No violation of integrity constraints
3.	Isolation	Concurrent changes invisible
4.	Durability	Committed update persist.

END TERM EXAMINATION [MAY-JUNE, 2016]
FOURTH SEMESTER [B.TECH]
DATABASE MANAGEMENT SYSTEM [ETCS–208]

Time : 3 Hrs. **M.M. : 75**

Note: *Attempt any five questions including Q no. 1 which is compulsory.*

Q.1. Define the following (Give example where necessary): **(10×2.5=25)**

Q.1. (a) Multivalued dependency

Ans. MVD Formal definition

Let R be a relation scheme and X and Y are subsets of R,

X ->> Y

holds if for all pairs of tuples s and t in r, such that

s[X] = t[X] there also exist tuples u and v where

s[X] = t[X] = u[X] = v[X]

s[Y] = u[Y]

t[R-XY] = u[R-XY]

s[R-XY] = v[R-XY]

t[Y] = v[Y]

Multi-valued dependencies occur in pairs.

Example: JointAppoint(facId, dept, committee) assuming a faculty member can belong to more than one department and belong to more than one committee.

Table must list all combinations of values of department and committee for each facId.

MVD Example

Course ->> Instructor

Course ->> Text

Course(Y)	Instructor(X)	Text(R-XY)
Intro	Kruse	Intro to CS
Intro	Wright	Intro to CS
CS1	Thomas	Intro to Java
CS1	Thomas	CS Theory Survey
CS2	Rhodes	Java Data Structures
CS2	Rhodes	Unix
CS2	Kruse	Java Data Structures
CS2	Kruse	Unix

Q.1. (b) Dependency preservation **(2.5)**

Ans. A decomposition of a relation R into $R_1, R_2, R_3, \ldots R_n$ is dependency preserving decomposition with respect to the set of Functional Dependencies F that hold on R only if the following is hold:

$$(F_1 \cup F_2 \cup F_3 \cup \ldots \cup F_n)+ = F^+$$

where,

$F_1, F_2, F_3 \ldots, F_n$_ sets of functional dependencies of relations $R_1, R_2, R_3, \ldots R_n$.

$(F_1, F_2, F_3, \ldots, \cup F_n)+$_closure of union of all sets of functional dependencies.

F+_closure of set of R.

If the closure of set of functional dependencies of individual relations $R_1, R_2, R_3, \ldots, R_n$ are equal to the set of functional dependencies of the main relation R (before

Scanned by CamScanner

decomposition), then we would say the decomposition D is lossless dependecy preserving decomposition.

Discussion with Example of Non-dependency preserving decomposition:

Dependency preservation is a concept that is very much related to Normalization process. Remember that the solution for converting a relation into a higher normal form is to decompose the relation into two or more relations. This is done using the set of functional dependencies identified in the lower normal form state.

For example, let us assume a relation R(A, B, C, D) with set of functional dependencies $F = \{A \to B, B \to C, C \to D\}$. There is no partial dependency in the given set F. Hence, this relation is in 2NF.

Is R(A, B, C, D) in 3NF? No. The reason is Transitive Functional Dependency. How do we convert R into 3NF? The solution is decomposition.

Assume that we decompose R(A, B, C, D) into R_1 (A,C,D) and R_2 (B,C).

In R_1 (A,C,D), the F D $C \to D$, the FD $C \to D$ holds. In R_2(B, C), the FD $B \to C$ holds. But, there is no trace of the FD $A \to B$ Hence this decomposition does not preserve dependency.

Q.1. (c) Functional dependency.

Ans. Functional dependency is a relationship that exists when one attribute uniquely determines another attribute. If R is a relation with attributes X and Y, a functional dependency between the attributes is represented as X–>Y, which specifies Y is functionally dependent on X. Here X is a determinant set and Y is a dependent attribute. Each value of X is associated precisely with one Y value.

Augmentation rule in Functional dependencey states that

when X –> Y, then XZ –>YZ.

Q.1. (d) Referential integrity.

Ans. Referential Integrity is set of constraints applied to foreign key which prevents entering a row in child table (where you have foreign key) for which you don't have any corresponding row in parent table i.e. entering NULL or invalid foreign keys. *Referential Integrity* prevents your table from having incorrect or incomplete relationship e.g. If you have two tables Order and Customer where Customer is parent table with primary key customer_id and Order is child table with foreign key customer_id. Since as per business rules you can not have an Order without a Customer and this business rule can be implemented using *referential integrity in SQL* on relational database. **Referential Integrity** will cause failure on any INSERT or UPDATE SQL statement changing value of customer_id in child table. If value of customer_id is not present in Customer table.

Advantage of Referential Integrity in RDBMS

(1) Referential Integrity prevents inserting records with incorrect details in table. Any insert or update operation will fail if it doesn't satisfy referential integrity rule.

(2) If a records from parent table is deleted, referential integrity allows to delete all related records from child table using cascade-delete functionality.

(3) Similar to second advantage if a record is updated on parent table, Referential Integrity helps to update in child table same attribute using cascade-update.

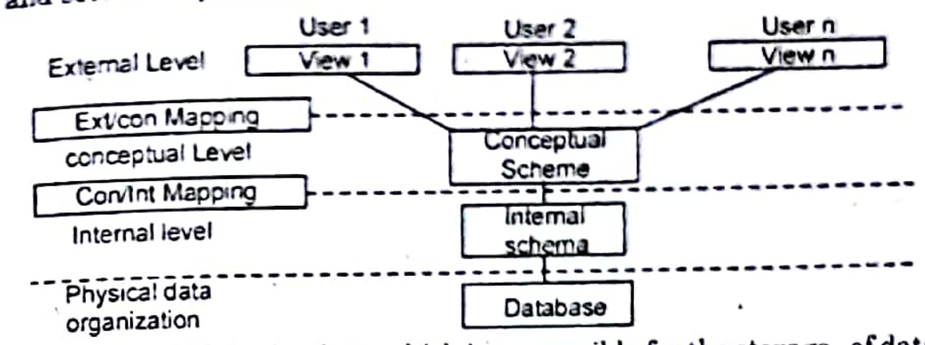
Q.1. (e) Internal schema.

Ans. The Internal Schema: A database will have only one internal schema, which contains definitions of the way in which data is physically stored. The interface between the internal schema and the conceptual schema identifies how an element in the conceptual schema is stored, and how it may be accessed.

If the internal schema is changed, this will need to be addressed in the interface between the internal and the conceptual schemas, but the conceptual and external schemas will not need to change. This means that changes in physical storage devices

such as disks, and changes in the way files are organised on storage devices, are transparent to users and application programs.

In distinguishing between "logical" and "physical" views of a system, it should be noted that the difference could depend on the nature of the user. While "logical" describes the user angle, and "physical" relates to the computer view, database designers may regard relations (for staff records) as logical and the database itself as physical. This may contrast with the perspective of a systems programmer, who may consider data files as logical in concept, but their implementation on magnetic disks in cylinders, tracks and sectors as physical.



This is the level of the database which is responsible for the storage of data on the storage media and places the data in such a format that it is only readable by the DBMS. Although the internal view and the physical view are so close that they are generally referred to a single layer of the DBMS but there lays thin line which actually separated the internal view from the physical view DBMS to some extent decides the way data is to be stored on the disk. This decision of the DBMS is based on the requirements specified by the DBA when implementing the database. Moreover the DBMS itself adds information to the data which is to be stored. Another important consideration for the storage of data at the internal level is that the data should be stored in such a way that it is secure and does not involve any security risks. **(2.5)**

Q.1. (f) Candidate key.

Ans. Candidate Key: (*i*) When more than one or graup of attributes serve as a uunique indntifier, they are each called as candidate key.

(*ii*) Candidate key are those attributes of a relation which posses the properties of uniqueness an irreducibility.

Uniquness Property: It means that at any given time, no. two tuples in the value R, at that time are duplicates of one another.

Irreducibility Property: "No proper subset of K has th uniquness property."

For example: If (Name, class) is unique then it can be identified as the candidate key if and only if Name and class individually are not unique.

Q.1. (g) Triggers.

Ans. Refer Q.4.(a). First Term Examination 2016.

Q.1. (h) DBA.

Ans. Refer Q.1.(c). End Term Examination 2015. **(2.5)**

Q.1. (i) Weak entity.

Ans. Refer Q.1.(d). First Term Examination 2016.

Q.1. (j) Foreign key.

Ans. Refer Q.3. (b) First Term Examination 2016.

Q.2. (a) Differentiate between the term Generalization and Specialization with example. **(6.5)**

Ans. Refer Q.3. (b) End Term Examination 2015.

Q.2. (b) Differentiate between Fragmentation, Replication and Transparency. **(6)**

Ans. Replication: Data replication is the process of storing separate copies of the database at two or more sites. It is a popular fault tolerance technique of distributed databases.

Scanned by CamScanner

Advantages of Data Replication:

• **Reliability** " In case of failure of any site, the database system continues to work since a copy is available at another site(s).

• **Quicker Response** "Availability of local copies of data ensures quick query processing and consequently quick response time.

Disadvantages of Data Replication:

• **Increased Storage Requirements** "Maintaining multiple copies of data is associated with increased storage costs. The storage space required is in multiples of the storage required for a centralized system.

• **Increased Cost and Complexity of Data Updating** " Each time a data item is updated, the update needs to be reflected in all the copies of the data at the different sites. This requires complex synchronization techniques and protocols.

• **Undesirable Application – Database coupling** " If complex update mechanisms are not used, removing data inconsistency requires complex co-ordination at application level. This results in undesirable application – database coupling.

Some commonly used replication techniques are:

• Near-real-time replication

• Pull replication

Fragmentation:

Fragmentation is the task of dividing a table into a set of smaller tables. The subsets of the table are called **fragments**. Fragmentation can be of three types: horizontal, vertical, and hybrid (combination of horizontal and vertical). Horizontal fragmentation can further be classified into two techniques: primary horizontal fragmentation and derived horizontal fragmentation.

Fragmentation should be done in a way so that the original table can be reconstructed from the fragments.

Advantages of Fragmentation:

• Since data is stored close to the site of usage, efficiency of the database system is increased.

• Local query optimization techniques are sufficient for most queries since data is locally available.

• Since irrelevant data is not available at the sites, security and privacy of the database system can be maintained.

Disadvantages of Fragmentation:

• When data from different fragments are required, the access speeds may be very high.

• Lack of back-up copies of data in different sites may render the database ineffective in case of failure of a site.

Vertical Fragmentation:

In vertical fragmentation, the fields or columns of a table are grouped into fragments. In order to maintain reconstructiveness, each fragment should contain the primary key field(s) of the table. Vertical fragmentation can be used to enforce privacy of data.

For example,

STUDENT

Regd_No	Name	Course	Address	Semester	Fees	Marks

Now, the fees details are maintained in the accounts section. In this case, the designer will fragment the database as follows.

```
CREATE TABLE STD_FEES AS
SELECT Regd_No, Fees
FROM STUDENT;
```

Horizontal Fragmentation

Horizontal fragmentation groups the tuples of a table in accordance to values of one or more fields. Horizontal fragmentation should also confirm to the rule of reconstructiveness. Each horizontal fragment must have all columns of the original base table.

For example, in the student schema, if the details of all students of Computer Science Course needs to be maintained at the School of Computer Science, then the designer will horizontally fragment the database as follows "

```
CREATE COMP_STD AS
SELECT * FROM STUDENT
WHERE COURSE = "Computer Science";
```

Hybrid Fragmentation:

In hybrid fragmentation, a combination of horizontal and vertical fragmentation techniques are used. This is the most flexible fragmentation technique since it generates fragments with minimal extraneous information. However, reconstruction of the original table is often an expensive task.

Hybrid fragmentation can be done in two alternative ways:

• At first, generate a set of horizontal fragments; then generate vertical fragments from one or more of the horizontal fragments.

• At first, generate a set of vertical fragments; then generate horizontal fragments from one or more of the vertical fragments.

Transparency: Transparency is the property of distributed databases by the virtue of which the internal details of the distribution are hidden from the users. The DDBMS designer may choose to fragment tables, replicate the fragments and store them at different sites. However, since users are oblivious of these details, they find the distributed database easy to use like any centralized database.

The dimensions of distribution transparency are:

• Location transparency • Fragmentation transparency

Location Transparency: Location transparency ensures that the user can query on any table(s) or fragment(s) of a table as if they were stored locally in the user's site. The fact that the table or its fragments are stored at remote site in the distributed database system, should be completely oblivious to the end user. The address of the remote site(s) and the access mechanisms are completely hidden.

Fragmentation Transparency: Fragmentation transparency enables users to query upon any table as if it were unfragmented. Thus, it hides the fact that the table the user is querying on is actually a fragment or union of some fragments. It also conceals the fact that the fragments are located at diverse sites.

Q.3. (*a*) Define closure of FD set. For the following relation

R(A, B, C, D)

With FDs as follows:

(*i*) $AB \rightarrow C$

$C \rightarrow A$

$BC \rightarrow D$

$ACD \rightarrow D$

$D \rightarrow EG$

$BE \rightarrow C$

$CG \rightarrow BD$

$CE \rightarrow AC$

Find the closure of (B, D) and (C, A). **(6.5)**

Ans. Closure of Functional Dependency set:

1. We need to consider *all* functional dependencies that hold. Given a set *F* of functional dependencies, we can prove that certain other ones also hold. We say these ones are **logically implied** by *F*.

Scanned by CamScanner

2. The **closure** of a set F of functional dependencies is the set of all functional dependencies logically implied by F.

3. We denote the closure of F by F^+.

4. To compute F^+, we can use some rules of inference called **Armstrong's Axioms**:

1. Reflexivity rule: If α is a set of attributes and $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$ holds.

2. Augmentation rule: If $\alpha \rightarrow \beta$ holds, and γ is a set of attributes, $\gamma\alpha \rightarrow \gamma\beta$ then holds.

3. Transitivity rule: If $\alpha \rightarrow \beta$ holds, and $\beta \rightarrow \gamma$ holds, then $\alpha \rightarrow \gamma$ holds.

4. These rules are **sound** because they do not generate any incorrect functional dependencies. They are also **complete** as they generate all of F^+.

Closure of (B, D) and (C, A):

{B,D}= {B,D,C} by DE → C.

{B,D,C}={B, D, A, C} by C → A.

So closure of (B,D) is {A, B, C, D}.

Closure of (C,A):

{C,A}= {B, A, C} by CG → BD.

{B,A,C}={B, D, A, C} by BC → D.

So closure of (C, A) is {A, B, C, D}.

Q.3. (b) What is canonical cover? How it is computted? **(6)**

Ans. (*i*) Set F: = E.

(*ii*) Replace each functional dependency $x \rightarrow \{A_1, A_2 \ldots A_n\}$ in F *by* the n functional dependencies : $x \rightarrow A_1$, $x \rightarrow A_2, \ldots x \rightarrow A_n$.

(*iii*) For each functional dependency $x \rightarrow$ A in F for each attribute B that is an element of X if {(F – {X→ A}} U {(X – {B}) → A}} is equivalent to F,

then replace X → A with (X – {B3) → A in F.

(*iv*) For each remaining functional dependency X → A in F.

if {F–{X → A}} is equivalent to F, then remove X → A from F.

For example: Let the given set of FDs be E: {B → A, D → A, AB → D}. We have to 1 find the minimum cover of E.

All above dependencies are in canonical form: so we have completed step of [Algorithm and can proceed to step (*ii*). In step (*ii*) we need to determine if AB → D any redundant attribute on the left hand side that is can it be replaced by ; B → Ddr A → D?

- Since B → A, by augmenting with B on both sides we have BB → AB, or B → AB i). However, AB → D as given (b).
- Hence by the transitive Rule, we get from (a) and (b), B → *D*. Hence AB → *D may* be replaced by B → D.
- We now have a self equivalent to original E, say E': {B → A, D → A, B → D}. No further reduction is possible in step (*ii*) since all FDs have a single attribute on the left hand side.
- In step (*iii*) we look for a redundant FD in E_r By using the transitive rule on B → D and D → A, we derive B → A. Hence, B → A is redundant in E' and can be eliminated.

⇒ Hence the minimum cover of E is {B → D, D → A}

Q.4. (a) Explain the "ACID" properties in brief. **(6)**

Ans. Refer Q.4. (b) Second Term Examination 2016.

Q.4. (b) What do you mean by serializability. Discuss the conflict serializability and view serializability with examples. **(6.5)**

Ans. Serializability: Serializability of a schedule means equivalence (in the outcome, the database state, data values) to a *serial schedule* (i.e., sequential with no

transaction overlap in time) with the same transactions. It is the major criterion for the correctness of concurrent transactions' schedule, and thus supported in all general purpose database systems.

Conflict Serializability

Two schedules are **conflict-equivalent** if one can be reached from the other through a series of swaps of adjacent operations, where no swap falls into one of the following patterns:

- The operations are by the same transaction
- The operations use the same database element, and at least one is a write

For example, Schedule U is conflict-equivalent to Serial Schedule 2, as shown by the following series of swaps.

Schedule U: $r_2(A), w_2(A), r_1(A), w_1(A), r_2(B), w_2(B)$

swap $w_1(A)$ and $r_2(B)$: $r_2(A), w_2(A), r_1(A), r_2(B), w_1(A), w_2(B)$

swap $w_1(A)$ and $w_2(B)$: $r_2(A), w_2(A), r_1(A), r_2(B), w_2(B), w_1(A)$

swap $r_1(A)$ and $r_2(B)$: $r_2(A), w_2(A), r_2(B), r_1(A), w_2(B), w_1(A)$

swap $r_1(A)$ and $w_2(B)$: $r_2(A), w_2(A), r_2(B), w_2(B), r_1(A), w_1(A)$

A schedule is **conflict-serializable** if it is conflict-equivalent to some serial schedule. We've just shown that Schedule U is conflict-serializable.

Testing for conflict-serializability

Using the definition of conflict-serializability to show that a schedule is conflict-serializable is quite cumbersome. There's a much more efficient algorithm:

1. Build a directed graph, with a vertex for each transaction.

2. Go through each operation of the schedule.

- If the operation is of the form $w_i(X)$, find each subsequent operation in the schedule also operating on the same data element X by a different transaction: that is, anything of the form $r_j(X)$ or $w_j(X)$. For each such subsequent operation, add a directed edge in the graph from T_i to T_j.
- If the operation is of the form $r_i(X)$, find each subsequent *write* to the same data element X by a different transaction: that is, anything of the form $w_j(X)$. For each such subsequent write, add a directed edge in the graph from T_i to T_j.

3. The schedule is conflict-serializable if and only if the resulting directed graph is acyclic. Moreover, we can perform a topological sort on the graph to discover the serial schedule to which the schedule is conflict-equivalent.

As an example, consider the following schedule:

$w_1(A), r_2(A), w_1(B), w_3(C), r_2(C), r_4(B), w_2(D), w_4(E), r_5(D), w_5(E)$

1. We start with an empty graph with five vertices labeled T_1, T_2, T_3, T_4, T_5.

2. We go through each operation in the schedule:

$w_1(A)$: A is subsequently read by T_2, so add edge $T_1 \rightarrow T_2$

$r_2(A)$: no subsequent writes to A, so no new edges

$w_1(B)$: B is subsequently read by T_4, so add edge $T_1 \rightarrow T_4$

$w_3(C)$: C is subsequently read by T_2, so add edge $T_3 \rightarrow T_2$

$r_2(C)$: no subsequent writes to C, so no new edges

$r_4(B)$: no subsequent writes to B, so no new edges

$w_2(D)$: C is subsequently read by T_2, so add edge $T_3 \rightarrow T_2$

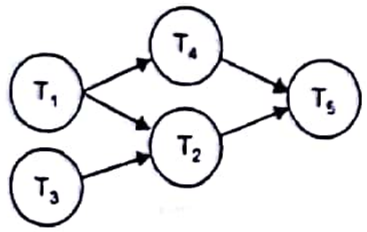
$w_4(E)$: E is subsequently written by T_5, so add edge $T_4 \rightarrow T_5$

$r_5(D)$: no subsequent writes to D, so no new edges

$w_5(E)$: no subsequent operations on E, so no new edges

3. We end up with the following directed graph.

Scanned by CamScanner

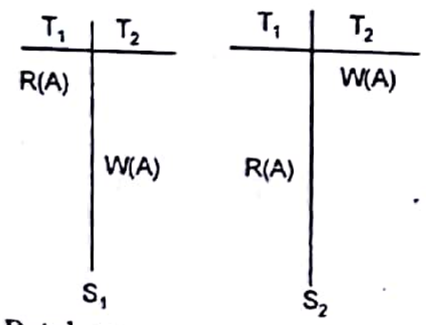


4. This graph has no cycles, so the original schedule must be serializable.

View Seriaizability: View equivalent schedule and View Serializable Schedule

View Equivalent Schedule : Consider two schedules S_1 and S_2, they are said to be view equivalent if following conditions are true :

- **Initial read must be same.**

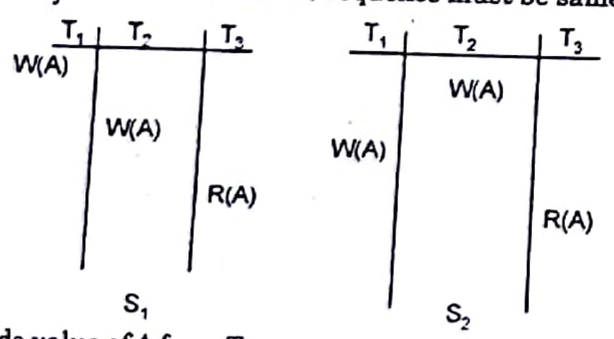


- S_1 : T_1 reads A from Database.

S_2 : T_1 reads A from T_2.

$\therefore S_1 \neq S_2$.

- There are two transactions say Ti and Tj, The schedule S_1 and S_2 are view equivalent if in schedule S_1, Ti reads A that has been updated by T_j, and in schedule S_2, Ti must read A from T_j. i.e. write-read(WR) sequence must be same between S_1



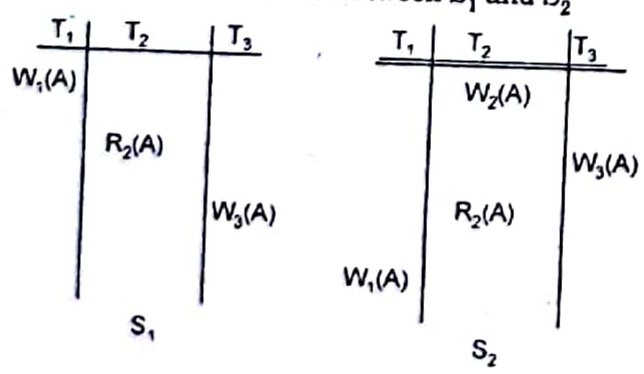
- S_1 : T_3 reads value of A from T_2.

S_2 : T_3 reads value of A from T_1.

$\therefore S_1 \neq S_2$.

i.e. write-read sequence is not same between S_1 and S_2.

Final write operations should be same between S_1 and S_2



S_1 : A is finally updated by T_3.

S_2 : A is finally updated by T_1.

$\therefore S_1 \neq S_2$.

- **View serializable schedule**

A Schedule is view serializable if it is view equivalent to any serial schedule. The following two examples will illustrate how to find view equivalent schedule.

Q.5. (a) Draw and explain a neat diagram of three level architecture of database system. **(6)**

Ans. Database client server architecture are partitioned into 3 types:

(1) Single Tier: In single tier system all software are present on a single machine (server system), which are accesed by terminals.

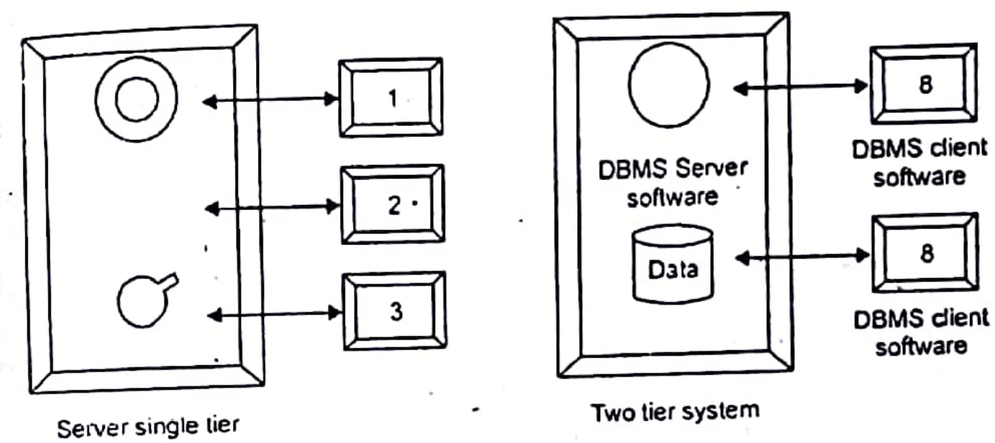


Fig. (A) **Fig. (B)**

(2) Two Tier: The application is divided into a component that besides at the client machine, which invokes database system functionally at the server machine through query language statement (Fig. B) API (Application program Interface) standards like ODBC and JDBC are required.

(3) Three Tier: This system adds middle ware which provide a way for the clients of one DBMS to access data from another DBMS on other machine (Fig. C). This system is more suitable for very large applications.

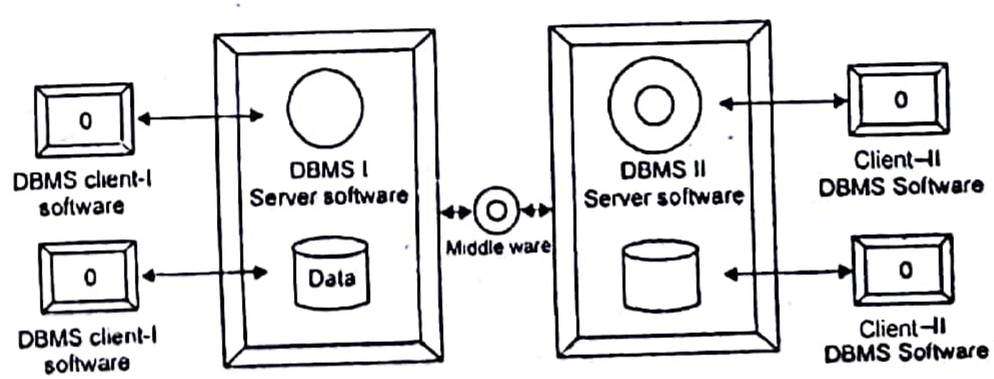
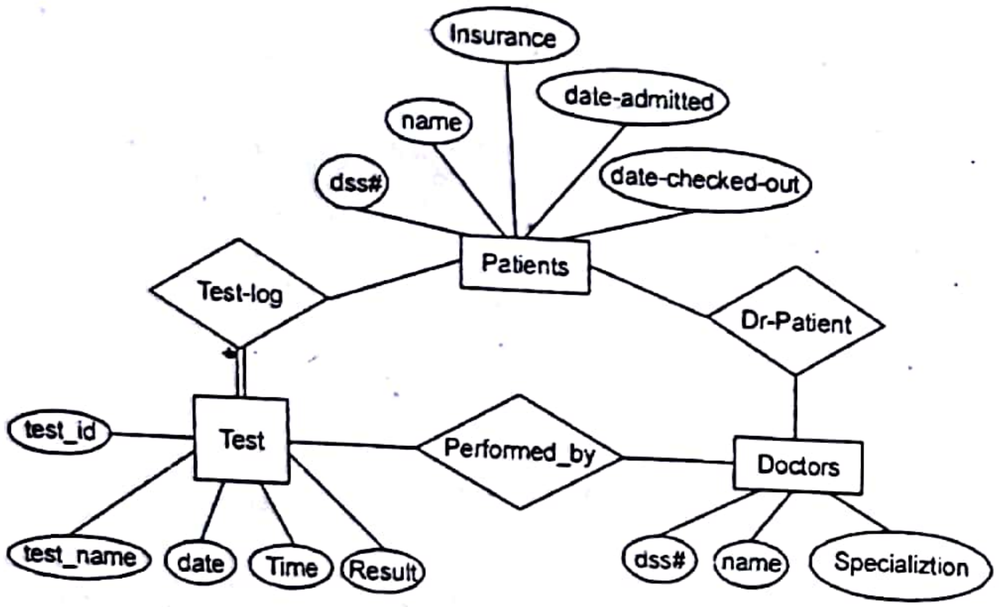


Fig. (c) Three Tier System

Q.5. (b) Construct an ER diagram for 'Hospital management system' There would be a set of patients and set of doctors. For each patient, there would be a log of various tests and examinations conducted. Make assumptions if necessary and clearly state them. **(6.5)**

Scanned by CamScanner

Ans.



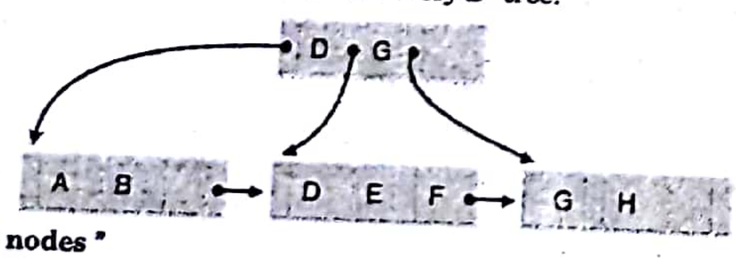
E-R diagram for a hospital.

Q.6. (*a*) How the B+ tree index files are maintained? Explain. **(6)**

Ans. B⁺ Tree:

A B⁺ tree is a balanced binary search tree that follows a multi-level index format. The leaf nodes of a B⁺ tree denote actual data pointers. B⁺ tree ensures that all leaf nodes remain at the same height, thus balanced. Additionally, the leaf nodes are linked using a link list; therefore, a B⁺ tree can support random access as well as sequential access.

Structure of B⁺ Tree: Every leaf node is at equal distance from the root node. A B⁺ tree is of the order **n** where **n** is fixed for every B⁺ tree.



Internal nodes "

- Internal (non-leaf) nodes contain at least $\lceil n/2 \rceil$ pointers, except the root node.
- At most, an internal node can contain **n** pointers.

Leaf nodes:"

- Leaf nodes contain at least $\lceil n/2 \rceil$ record pointers and $\lceil n/2 \rceil$ key values.
- At most, a leaf node can contain **n** record pointers and **n** key values.
- Every leaf node contains one block pointer **P** to point to next leaf node and forms a linked list.

B⁺ Tree Insertion

- B⁺ trees are filled from bottom and each entry is done at the leaf node.
- If a leaf node overflows–
  - Split node into two parts.
  - Partition at $i = \lceil (m+1)/2 \rceil$.
  - First **i** entries are stored in one node.
  - Rest of the entries (i+1 onwards) are moved to a new node.
  - i^{th} key is duplicated at the parent of the leaf.

- If a non-leaf node overflows"
  - Split node into two parts.
  - Partition the node at $i = \lceil (m+1)/2 \rceil$.
  - Entries up to **i** are kept in one node.
  - Rest of the entries are moved to a new node.

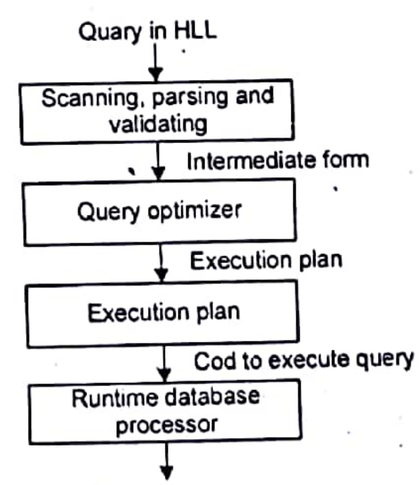
B⁺ Tree Deletion

- B⁺ tree entries are deleted at the leaf nodes.
- The target entry is searched and deleted.
  - If it is an internal node, delete and replace with the entry from the left position.
- After deletion, underflow is tested,
  - If underflow occurs, distribute the entries from the nodes left to it.
- If distribution is not possible from left, then
  - Distribute from the nodes right to it.
- If distribution is not possible from left or from right, then
  - Merge the node with left and right to it.

Q.6. (*b*) Explain the method of "query processing" in brief. **(6.5)**

Ans. The procedure of selecting the best plan or strategy to be used in responding to a database request and executing the plain to generate the response is known as query processing. It is a stepwise process. The SQL query is first of all scanned, parsed and validated. Let us draw a flow chart now.

The scanner identifies the language takens such as SQL key words, attribute names and relation names in the text of query. The parser checks the query syntax to determine whether it is formulated according to the syntax rules of the query language. The query must also be validated by checking that all attributes and relation names are valid. An internal representation of the query is then created usually as a tree data structure called as a query-tree.



Flowcharts for query processing

Q.7. (*a*) Discuss clearly inner join, outer join and theta join with example. **(6)**

Ans. Theta (0) Join

Theta join combines tuples from different relations provided they satisfy the theta condition. The join condition is denoted by the symbol 0.

Notation

$R1 \bowtie_0 R2$

Scanned by CamScanner

R1 and R2 are relations having attributes (A1, A2, .., An) and (B1, B2,.. ,Bn) such that the attributes don't have anything in common, that is R1 ∩ R2 = ϕ.

Theta join can use all kinds of comparison operators.

Student

SID	Name	Std
101	Alex	10
102	Maria	11

Subjects

Class	Subject
10	Math
10	English
11	Music
11	Sports

Student_Detail

STUDENT $\bowtie_{\text{Student.Std = Subject.Class}}$ SUBJECT

Student_detail

SID	Name	Std	Class	Subject
101	Alex	10	10	Math
101	Alex	10	10	English
102	Maria	11	11	Music
102	Maria	11	11	Sports

INNER Join or EQUI Join

This is a simple JOIN in which the result is based on matched data as per the equality condition specified in the query.

SELECT column-name-list
from *table-name1*
INNER JOIN
table-name2
WHERE table-name1.column-name = table-name2.column-name;

Example of Inner JOIN

The **class** table,

ID	NAME.
1	abhi
2	adam
3	alex
4	anu

The **class_info** table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI

Inner JOIN query will be,

SELECT * from class, class_info where class.id = class_info.id;

The result table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI

Outer Joins

Theta Join, Equijoin, and Natural Join are called inner joins. An inner join includes only those tuples with matching attributes and the rest are discarded in the resulting relation. Therefore, we need to use outer joins to include all the tuples from the participating relations in the resulting relation. There are three kinds of outer joins– left outer join, right outer join, and full outer join.

Left Outer Join(R ⟕ S)

All the tuples from the Left relation, R, are included in the resulting relation. If there are tuples in R without any matching tuple in the Right relation S, then the S-attributes of the resulting relation are made NULL.

Left

A	B
100	Database
101	Mechanics
102	

Right

A	B
100	Alex
102	Maya
104	Mira

Courses ⟕ HoD

A	B	C	D
100	Database	100	Alex
101	Mechanics	—	—
102	Electronics	102	Maya

Right Outer Join: (R ⟖ S)

All the tuples from the Right relation, S, are included in the resulting relation. If there are tuples in S without any matching tuple in R, then the R-attributes of resulting relation are made NULL.

Courses HoD

A	B	C	D
100	Database	100	Alex
102	Electronics	102	Maya
—	—	104	Mira

Full Outer Join: (R ⟗ S)

All the tuples from both participating relations are included in the resulting relation. If there are no matching tuples for both relations, their respective unmatched attributes are made NULL.

Scanned by CamScanner

Courses HoD

A	B	C	D
100	Database	100	Alex
101	Mechanics	—	—
102	Electronics	102	Maya
—	—	104	Mira

Q.7. (*b*) For the relation R = (A, B, C) and S = (D,E,F) and relation r(R) and s(s), give SQL statement for following expressions and explain. **(6.5)**

(*i*) $\pi_A(r)$ (*ii*) $\sigma_n = 17(r)$

(*iii*) $r\ X\ s$ (*iv*) $\pi_{A,F}(\sigma_{C=D}(rXs))$

Ans. (*i*). SELECT A FROM r.

(*ii*) SELECT * FROM r WHERE r.B=17.

(*iii*) SELECT * FROM r,s.

(*iv*) SELECT A , F FROM r, s WHERE r.C=s.D

Q.8. Attempt any two: **(2×6.2 = 12.5)**

Q. 8. (a) How the deadlock is detected in transaction? Explain its recovery process also. **(6.25)**

Ans. Deadlock: Deadlock are preferable to there inconsistent states because they can be handled by rolling back of transactions. The inconsistent states may lead to real world problems that cannot be handled by DBMS.

The rule is "When a transaction requests a Lock on a data item in a particular mode and no other transaction has a lock on the same data item in a conflicting mode, then the lock can be granted."

Deadlock Detection: It is defined as a periodic check mechanism by DBMS to determine whether a deadlock has occured or not. Actually, the frequency of deadlocks depends upon the query load and the physical organization of database.

In 1981, Gray proposed that the deadlock frequency (deadlock per second) is:

(*a*) Square of the degree of multi-programming (DOM).

(*b*) Fourth power of the transaction size. For this a system must maintain information about the current allocation of data items to transactions provide on appropriate algorithm to find if the system has enered a deadlock state and to recover from deadlock if it exists.

Wait-For Graph (WFG) for Deadlock Detection: A very simple way to detect a state of deadlock is to draw wait-for graph. This graph is constructed and maintained by the system.

Notations Used:

1. One node is created in the wait-for graph for each transaction that is currently executing.

2. Whenever a transaction, Ti, is waiting to Lock an item—A that is currently locked by a transaction, Tj a directed edge (Ti -> Tj) is created in the wait-for graph. When Tj releases the lock on the items that Ti is waiting for, the directed edge is dropped from the wait for graph.

Q.8. (*b*) Normalize the following relation to as much as possible citing the reasons and anamoloies.

R(emp-no, name, street, city, companyname, company-city, manager-name, age, salary, marital-status, spouse-name)

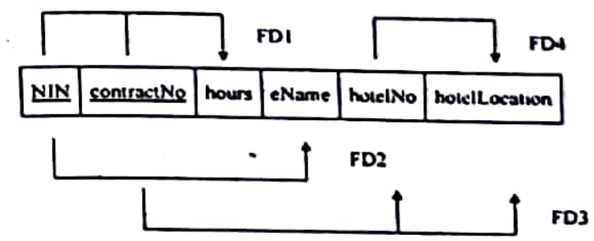
Ans. An agency called InstantCover supplies part-time/temporary staff to hotels throughout Scotland. The table shown in Figure lists the time spent by agency staff working at two hotels. The National Insurance Number (NIN) is unique for employee.

NIN	contract No	hours per week	eName	hotelNo	hotel Location
113567WD	C1024	16	John Smith	H25	Edinburgh
234111XA	C1024	24	DianeHocine	H25	Edinburgh
712670YD	C1025	28	Sarah White	H4	Glasgow
113567WD	C1025	16	John Smith	H4	Glasgow

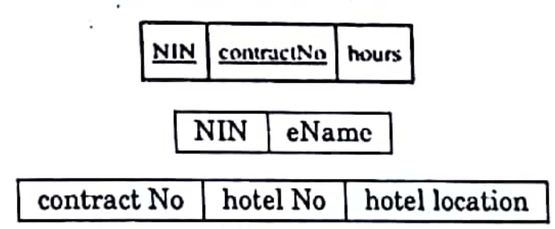
Describe and illustrate the process of normalizing the table . State any assumptions you make about the data shown in this table.

The table shown is susceptible to update anomalies. Provide examples of insertion, deletion, and modification anomalies.

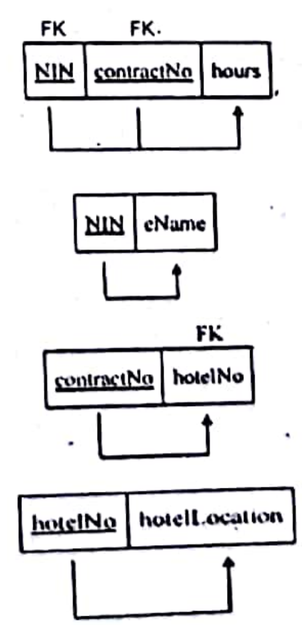
Answer: In the 1NF we remove all the repeating groups if any and assigned primary keys (candidate keys). Then we figure out the functional dependencies (FDs). By using dependency diagram we represent the table as shown below. (NF –stand for Normal Form)



Obviously FD2 and FD3 violate teh 2NF (partial dependency) and FD4 violate 3NF (transitive dependency). In the 2NF we remove the partial dependency of the FD2 and FD3 by splitting it into new tables as shown below.



In 3NF we remove the transitive dependency of teh FD4 by splitting it into new table as shown below.



Scanned by CamScanner

Finally, let re-arrange the table and assign names:

NIN	eName

Staff(NIN, eName)

hotel No	hotel Location

Hotel(hotelNo. hotalLocation)

	FK
contractNo	hotelNo

Contract(contract No, hotel No.)

FK	FK	
NIN	contractNo	hours

WorkHours(NIN, contract No, hours)

It is obvious that the NIN and contractNo can be candidate keys. There are data redundancies in term of repeating groups.

Insertion anomalies: To insert the details of new hotel we must know the NIN of the staff because this is primary key and we cannot assign NULL value for primary key. For example if we want to insert the details of new hotel, we must also insert the NIN and contractNo.

To recruit a new staff we need to know the details of hotel. For example if we want to recruit and assign new staff at H4 we need to enter the correct details of H4 so that the hotel details are consistent with values for hotel H4.

Deletion anomalies: If we delete a record from the table that represent the staff name for example, John Smith, the details about the contract also will be lost from the database and will affect other record as seen in the second row of the contractNo (C1024).

Oppositely, when we delete a contractNo, other staff details will also be lost. For example if we delete C1025, both Sarah White and John Smith details will be lost.

Modification anomalies: If we change the value of the contractNo, several records need to changed, for example row 1 and 2, also row 3 and 4 if we edit either contractNo.

If we edit the staff details, for example updating John Smith, several records need to be updated. This also applies to the hotelNo column.

Q.8. (c) State the multi-version time stamp based protocol. Suggests a scheme to avoid the phantom phenomenon.

Ans. In this scheme, there are three locking modes for an item-read, write and certify. Hence the state of lock (X) for an item X can be one of road-locked, write locked, certify locked or unlocked. Certify lock is also exclusive lock. We can describe the relationship between read, write and certify locks by means of the lock compatibility table shown in fig. Here yes means 'T' can obtain the lock and 'No' means T' must wait until T release the lock'.

The idea behind multiversion 2PL is to allow other transaction T' to read an item X while a single transaction T holds a write locks on X. This is accomplished by allowing two versions for each item X; one version must always have been written by some committed transaction. The second version of X while T holds the write lock. Transaction T can write the value of X as needed, without affecting the value of the committed version X.

(a) read/write locking scheme

		Read	Write
T hold the lock	Read	yes	no
	write	no	no

(b) read/write/certify locking scheme.

		Read	Write	certify
T hold the lock	Read	yes	yes	no
	write	yes	no	no
	certify	no	no	no

However, once T is ready to commit, it must obtain a certify lock on all items that it currently holds write locks on before it can commit. The certify lock is not compatible with read locks, so the transaction may have to wait until all its write-locked items are released by any reading transactions in order to obtain the certify locks. Once the certify are locked, the committed X of the data item is set to the value of version X', version X' is discarded, and the certify locks are then released.

IMPORTANT QUESTIONS

Q.1. Which of the following schedules is (conflict) serializable? For each serializable schedule, determine the equivalent serial schedules. (5)

(i) $r_1(X); r_3(X); w_1(X); r_2(X); w_3(X);$ **(ii)** $r_1(X); r_3(X); w_3(X); w_1(X); r_2(X);$

(iii) $r_3(X); r_2(X); w_3(X); r_1(X); w_1(X);$ **(iv)** $r_3(X); r_2(X); r_1(X); w_3(X); w_1(X);$

Ans. (a) Let there be three transactions T_1, T_2 and T_3. They are executed concurrently and produce a schedule (S). S is serializable if it can be reproduced as at least one serial schedule ($T_1 \to T_2 \to T_3$ or $T_1 \to T_3 \to T_2$ or $T_2 \to T_1 \to T_3$ or $T_3 \to T_1 \to T_2$).

This schedule is not serializable because T_1 reads x ($r_1(X)$) before T_3 but T_3 reads $X(r_3(X))$ before T_1 writes $X(w_1(X))$, where X is a common data item. The operation $r_2(X)$ of T_2 does not affect the schedule at all so its positioning the schedule is irrelevant. In a serial schedule T_1, T_2 and T_3 the operation $w_1(X)$ comes before $r_3(X)$, which does not happen in the equation.

(b) The schedule is not serializable because T_1 reads $X(r_1(X))$ before T_3 but T_3 writes $X(w_3(X))$ before T_1 writes $X(w_1(X))$. The operation $r_2(X)$ of T_2 does not affect the schedule at all so its position in the schedule is irrelevant. In a serial schedule, T_1 T_3 and T_2, $r_3(X)$ and $w_3(X)$ must come after $w_1(X)$, which does not happen in the equation.

(c) This schedule is seriabilizable because all conflicting operations of T_3 happens before all conflicting operations of T_3 happens before all conflicting operation of T_1, T_2 has only one operation, which is a read on $X(r_2(X)$, which does not conflict with any other operation. Thus this serializable schedulers equivalent to $r_2(X); r_3(X); w_3(X); r_1(X); w_1(X);$ (e.g.: $T_2 \to T_3 \to T_1$) serial schedule.

(d) This is not a serialziable schedule because T_3 reads $X(r_3(X))$ before T_1 reads $X(r_1(X))$ but $r_1(X)$ happens before T_3 writes $X(w_3(X))$. In a serial schedule T_1, T_2 and T_1, $r_1(X)$ will happen after $w_5(X)$, which, does not happen in the question.

Q.2. Relation $R(A, B, C, D, E, F)$ satisfies the following functional depenednecies: $AB \to C$, $C \to A$, $BC \to D$, $ACD \to B$, $BE \to C$, $CE \to FA$, $CF \to BD$, $D \to EF$. Find out the closure of FD's. (5)

Ans. (i) Start with

$[A, B] = \{A, B, C\}$ By $AB \to C$

$\{A, B, C\} = \{A, B, C, D\}$ By $BC \to D$

$\{A,B,C,D\} = \{A, B, C, D, E, F\}$ By $D \to EF$

So $\{A, B\}^+ = \{A, B, C, D, E, F\}$

(ii) $C \to A$

So, $\{C\} = \{C, A\}$ By $C \to A$

$\{A, C\} = \{A, C\}$ No more attribute

So, $\{C\}^+ = \{A, C\}$

(iii) $BC \to D$

So $\{B, C\} = \{B, C, D\}$ By $BC \to D$

$\{B,C,D\} = \{A, B, C, D\}$ By $C \to A$

$\{A, B, C, D\} = \{A, B, C, D, E, F\}$ By $D \to EF$

So $\{B, C\}^+ = \{A, B, C, D, E, F\}$

(iv) $ACD \to B$

So $\{A, C, D\} = \{A, C, D, B\}$ By $ACD \to B$

$\{A, B, C, D\} = \{A, B, C, D, E, F\}$ By $D \to EF$

So, $\{A, C, D\}^+ = \{A, B, C, D, E, F\}$

Scanned by CamScanner

(v) $BE \rightarrow C$, So, $\{B, E\}$ = $\{B, C, D\}$ By $BE \rightarrow C$
$\{B, C, E\}$ = $\{B, C, D, E\}$ By $BC \rightarrow D$
$\{B, C, D, E\}$ = $\{B, C, D, E, A, F\}$ By $C \rightarrow A$
$\{B, C, D, E, A\}$ = $\{B, C, D, E, A, F\}$ By $D \rightarrow EF$
So $\{B, E\}^+$ = $\{A, B, C, E, F\}$

(vi) $CE \rightarrow FA$, So $\{C, E\}$ = $\{C, E, A, F\}$ By $CE \rightarrow FA$
$\{A, C, E, F\}$ = $\{A, B, C, D, E, F\}$ By $CF \rightarrow BD$
So, $\{C, E\}^+$ = $\{A, B, C, D, E, F\}$

(vii) $CF \rightarrow BD$, So, $\{C, F\}$ = $\{C, F, B, D\}$ By $CF \rightarrow BD$
$\{B, C, D, F\}$ = $\{B, C, D, E, F\}$ By $D \rightarrow EF$
$\{B, C, D, E, F\}$ = $\{A, B, C, D, E, F\}$ By $C \rightarrow A$
So, $\{C, F\}^+$ = $\{A, B, C, D, E, F\}$

(viii) $D \rightarrow EF$, So, $\{D\}$ = $\{D, E, F\}$ By $D \rightarrow EF$
$\{D, E, F\}$ = $\{D, E, F\}$
$\{D\}^+$ = $\{D, E, F\}$

Q.3. What is Data Abstraction? Explain with suitable example physical level, logical level and view level. (4)

Ans. Data Abstraction: Data Abstraction is one of the fundamental characterstic of any database management system, which helps in making data more accurate and easy to use. Abstraction refers to the act of representing essential features, without including background details or explanations. So, data abstraction refers to the act of representing data without giving details that how data are stored or maintained.

Different level of abstraction are:

1. Physical level or Internal level: It is the lowest level of abstraction which specifies stroage details that how data are actually stored on disks or on tapes. It specifies in the manner in which records are stored either as the collection of pages or as the collection of records.

2. Logical Level: The next higher level of abstraction describes what data are stored in the database, and what relationships exist among those data. There is only one conceptual schema per database. This schema also contains the method of deriving the objects in the conceptual view from the internal view.

3. View level: It is the highest level of abstraction which describes different views of the entire database. These views are designed according to the requirements of users who wants to access only a part of the database. A database may have several views, according to the demand of individual user or the group of users. The data in these views are not exactly stored in DBMS but they are computed using specification of view described by user.

Example of three levels

Customer–Loan			
Cust_ID	:	101	View
Loan_No	:	1011	
Amount in Dollars	:	8755.00	

CREATE TABLE Customer_Loan ()

Cust_ID	Number (4)	Logical
Loan_No	Number (4)	
Amount-in-Dollars	Number (7, 2)	

Cust-ID	TYPE = BYTE(4), OFFSET = 0	Physical
Loan-No.	TYPE = BYTE(4), OFFSET = 4	
Amount_in_Dollars	TYPE = BYTE (7), OFFSET = 8	

FIRST TERM EXAMINATION [FEB. 2017]
FOURTH SEMESTER [B.TECH]
DATABASE MANAGEMENT SYSTEMS [ETCS–208]

Time : 1.5 Hrs. **M.M. : 30**

Note: Q.No 1 is compulsory and attempt any two more questions from the remaining-

Q.1. (a) Differentiate between primary key, candidate key and super key. (2)

Ans. A **candidate** key is the most minimal subset of fields that uniquely identifies a tuple. For example if you have a candidate key on the column "user_id" and "pet_id" you'll never have more than 1 tuple with the same user_id and pet_id and neither user_id nor pet_id individually will work as a unique identifier for the tuple.

A **super key** is a set of fields that contains a key. Using the above example where the combination of "user_id" and "pet_id" uniquely identifies a tuple if we added "pet_name" (which is not key because we can have multiple pets named "fluffy") it would be a super key. Basically it's like a candidate key without the "minimal subset of fields" constraint.

A **primary key** is a candidate key that you tell the DB to optimize on. There might be multiple ways of referring to a unique tuple (ie. multiple candidate keys) but you can specify one when you're creating the table that you will use the most frequently.

Q.1.(b) List two reasons why null values might be introduced into the database. (2)

Ans. Nulls may be introduced into the database because the actual value is either unknown or does not exist. For example,

(1) An employee whose address has changed and whose new address is not yet known should be retained with a null address.

(2) If employee tuples have a composite attribute dependents and a particular employee has no dependents, then that tuple's dependents attribute should be given *a* null value.

Q.1.(c) Define the concept of aggregation in an E-R diagram. Give one example where this concept is useful. (2)

Ans. Consider a data box with information about employes who work on a particular project and use number of machines doing that work. We get the E.R. diagram shown in fig.

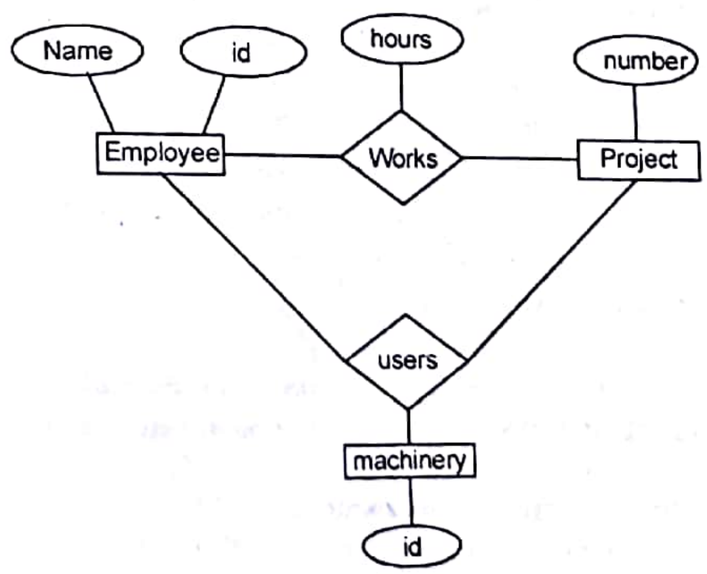


Fig. E.R. diagram with redundant relationship

Scanned by CamScanner

The solution is to use aggregation : An abstraction through which relationship are treated as higher level entities.

For an example we treat the relationship set works and the entity sets employee and project as a higher level entity set called works.

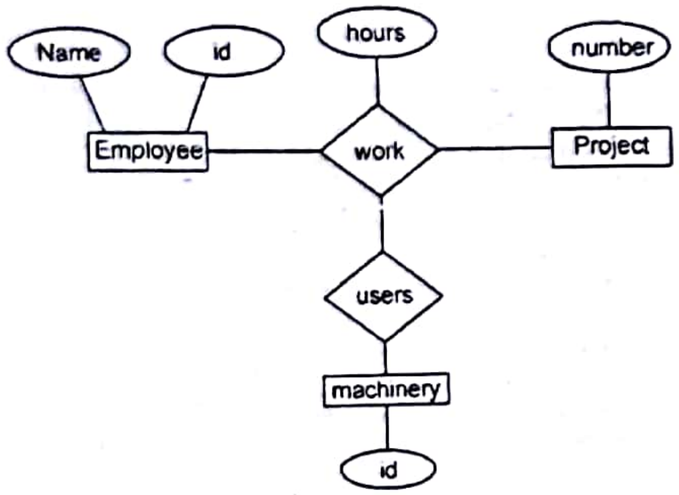


Fig. E.R. diagram with aggregation

Q.1.(d) What is generalized projection? Explain with the help of an example. **(2)**

Ans. GENERALIZED PROJECTION: - It extends the projection operation by allowing arithmetic functions to be used in projection list.

$\Pi_{F1,F2...Fn}(E)$

Where E: relational algebra expression; Fi: arithmetic expression

Example:

Table "Credit-info" :-

Code:

Customer-name	Limit	Credit_Balance
abc	2000	500
xyz	500	250
pqr	700	100
mno	1500	1000

Find how much money a person can spend.

Code:

[B]Π [SIZE=1]Customer-name,(Limit-Credit_balance)[/SIZE](Credit-info) [/B]

AGGREGATE FUNCTION:-It takes a collection of values and returns a single value as a result.

Q.1.(e) Explain how triggers are executed. **(2)**

Ans. Refer Q.4. (a) First Term Exam 2016.

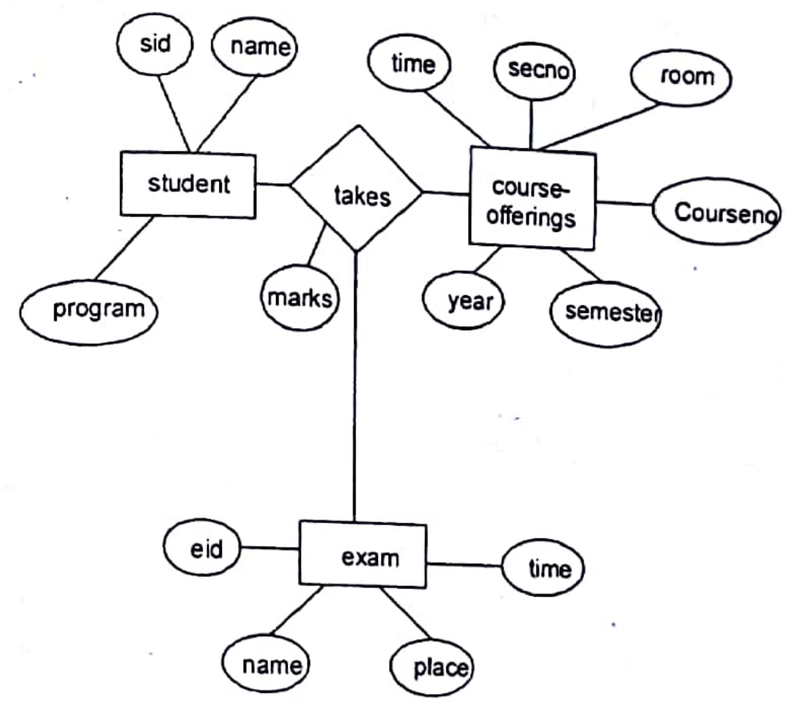
Q.2.(a) What is the difference between the terms 'relation' and 'relation schema'? Explain with an example. **(3)**

Ans.

Relation	Relation Scheme
(i) A relation is defined as a table.	A Relation schema also known as database schema is defined as a set of attributes.
(ii) It is a collection of rows and columns.	It is collection of metadata.
(iii) Attributes, degree, domain, tuples, cardinality etc belong to a relation.	It describes the structure and constraints of data representing a domain

Q.2.(b) Consider a database used to record the marks that students get in different exams of different courses. Construct an E-R diagram that models exam as an entity and uses a ternary relationship for the above database. Clearly state the assumptions made if any. **(5)**

Ans.



E-R diagram for marks database.

Q.2.(c) Does the relational model, as seen by an SQL: query writer, provide physical and logical data independence? Explain. **(2)**

Ans.(i) Data independence is the type of data transparency that matters for a centralized DBMS. It refers to the immunity of user applications to changes made in the definition and organization of data.

Physical data independence deals with hiding the details of the storage structure from user applications. The application should not be involved with these issues, since there is no difference in the operation carried out against the data.

The ability to modify schema definition in one level without affecting schema definition in the next higher level is called data independence. There are two levels of data independence, they are Physical data independence and Logical data independence.

Scanned by CamScanner

1. Physical data independence is the ability to modify the physical schema without causing application programs to be rewritten. Modifications at the physical level are occasionally necessary to improve performance. It means we change the physical storage/level without affecting the conceptual or external view of the data. The new changes are absorbed by mapping techniques.

2. Logical data independence is the ability to modify the logical schema without causing application program to be rewritten. Modifications at the logical level are necessary whenever the logical structure of the database is altered (for example, when money-market accounts are added to banking system). Logical Data independence means if we add some new columns or remove some columns from table then the user view and programs should not change. It is called the logical independence. For example: consider two users A & B. Both are selecting the empno and ename. If user B add a new column salary in his view/table then it will not effect the external view user; user A, but internal view of database has been changed for both users A & B. Now user A can also print the salary. It means if we change in view then program which use this view need not to be changed.

Q.3.(a) Explain Division operation in relational algebra. Given the following relations. **(4)**

A

sno	pno
S1	P1
S1	P2
S1	P3
S1	P4
S2	P1
S2	P2

B_1

pno
P2

B_2

pno
P2
P4

Calculate (1) $A \div B_1$ (2) $A \div B_2$

Ans. 1.

sno
S1
S2

2.

sno
s1

Q.3.(b) Consider the follwoing schema for an institude's library:

Student (Roll No, Name, Father – name, Branch)

Book (ISBN, Title, Author, Publisher)

Issue (Roll No, ISBN, Date-of-issue)

Write the following queries either in relational algebra or SQL

(1) List roll no. and name of all students of the branch CSE. **(2)**

(2) Find the name of students who have issued a book published by 'ABC' publisher. **(2)**

(3) List title of all books issued on or before Jan 1,2016. **(2)**

Ans. (*i*) $\pi_{S.\text{roll no.}, S.\text{name}}$ (student) $p_{s.\text{branch}}$ = "CSE"

(*ii*) $\pi_{S.\text{name}}$ ($\sigma_{B.\text{Publisher} = \text{"ABC"}}$ (Students $\bowtie$ Books B)).

(*iii*) $\pi_{s.\text{name}}$ ($\sigma_{\text{date-of-issue} = \text{"jan-1-2016"}}$ (Books B $\bowtie$ Issue I)

$\pi_{I.\text{Date_of_Issue}}$ = 'Jan-1-2016'

Q.4.(a). Let R=(A, B, C) and let r_1 and r_2 both be relations on schema R. Give an expression in SQL that is equivalent to each of the following queries: (2+1)

1. $\Pi_{AB}(r_1) \bowtie \Pi_{BC}(r_2)$

2. $r_1 \cup r_2$

Ans. 1. select r_1.A, r_1.B, r_2.C from r_1, r_2 where r_1.B = r_2.B

2. (select * fromr_1) union (select * fromr_2)

Q.4.(b). What is a stored procedure? Why it is useful? Give an example. (3)

Ans. A stored procedure is a set of Structured Query Language (SQL) statements with an assigned name, which are stored in a relational database management system as a group, so it can be reused and shared by multiple programs.

Most major DBMs support stored procedures; however, not all do. You will need to verify with your particular DBMS help documentation for specifics.

To create a stored procedure the syntax is fairly simple:

```
CREATEPROCEDURE<owner>.<procedure name>
<Param><datatype>
AS
<Body>
```

So for example:

```
CREATEPROCEDUREUsers_GetUserInfo
  @login nvarchar(30)=null
AS
SELECT * from [Users]
WHEREISNULL(@login,login)=login
```

Need for Stored Procedure:

A benefit of stored procedures is that you can centralize data access logic into a single place that is then easy for DBA's to optimize. Stored procedures also have a security benefit in that you can grant execute rights to a stored procedure but the user will not need to have read/write permissions on the underlying tables.

Q.4.(c) Differentiate between theta-join and equi-join. **(2)**

Ans. Theta (θ) Join

Theta join combines tuples from different relations provided they satisfy the theta condition. The join condition is denoted by the symbol θ.

Notation

$R1 \bowtie_\theta R2$

R1 and R2 are relations having attributes (A1, A2, .., An) and (B1, B2,.. ,Bn) such that the attributes don't have anything in common, that is $R1 \cap R2 = \Phi$.

Theta join can use all kinds of comparison operators.

Student

SID	Name	Std
101	Alex	10
102	Maria	11

Scanned by CamScanner

Subjects

Class	Subject
10	Math
10	English
11	Music
11	Sports

Student_Detail

STUDENT $\bowtie_{\text{Student.Std = Subject.Class}}$ SUBJECT

Student_detail

SID	Name	Std	Class	Subject
101	Alex	10	10	Math
101	Alex	10	10	English
102	Maria	11	11	Music
102	Maria	11	11	Sports

Equijoin

When Theta join uses only **equality** comparison operator, it is said to be equijoin. The above example corresponds to equijoin.

Q.4.(d) Explain the importance of defining view. **(2)**

Ans. A view is nothing more than a SQL statement that is stored in the database with an associated name. A view is actually a composition of a table in the form of a predefined SQL query.

A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depends on the written SQL query to create a view.

Views, which are kind of virtual tables, allow users to do the following:

- Structure data in a way that users or classes of users find natural or intuitive.
- Restrict access to the data such that a user can see and (sometimes) modify exactly what they need and no more.
- Summarize data from various tables which can be used to generate reports.

Example:

Consider the CUSTOMERS table having the following records:

```
+----+----------+-------+----------+----------+
| ID | NAME     | AGE   | ADDRESS  | SALARY   |
+----+----------+-------+----------+----------+
| 1  | Ramesh   | 32    | Ahmedabad | 2000.00 |
| 2  | Khilan   | 25    | Delhi     | 1500.00 |
| 3  | Kaushik  | 23    | Kota      | 2000.00 |
```

Now, following is the example to create a view from CUSTOMERS table:

```
SQL > CREATE VIEW CUSTOMERS_VIEW AS
SELECT name, age
FROM CUSTOMERS;
```

Now, you can query CUSTOMERS_VIEW in similar way as you query an actual table. Following is the example:


```
SQL > SELECT * FROM CUSTOMERS_VIEW;
```

This would produce the following result:

```
| name     | age |
+----------+-----+
| Ramesh   | 32  |
| Khilan   | 25  |
| Kaushik  | 23  |
```

Q.5. What is the difference between serial schedule and Serializable schedule?

Ans. Serial Schedule: Each serial schedule consists of a sequence of instruction from various transactions, where the instructions belonging to one single transaction appear together in that schedule.

T_1	T_2
A = A × 10 B = B – 10;	
	A = A/100; B = B × 4

Schedule 1: Serial Schedule

Serializable Schedule: A non-serial schedule that is equivalent to some serial execution of transactions is known as serializable schedle.

Scanned by CamScanner

END TERM EXAMINATION [MAY-JUNE 2017]
FOURTH SEMESTER [B.TECH]
DATABASE MANAGEMENT SYSTEMS [ETCS–208]

Time : 3 Hrs. **M.M. : 75**

Note: Attempt five questions including Q. No. 1 which is compulsory.

Q.1.Answer the following equations in brief:

Q.1. (a) List the role of Databse Administrator? **(2.5)**

Ans. DBA has central control over the system. The database administrator is responsibles for following functions:

(i) Schema Design and maintainance,

(ii) Physical schema and organisation Modification ,

(iii) Authorization and security,

(iv) Integrity constraint specification,

(v) Recovery from failure and give backup,

(vi) Database upgradation

(vii) Granting access to users

(viii) Physical database design

DBA is the central controller of the database system who manages all resources like database. DBMS and related software. A DBA provides the necessary technical support for implementing policy decision of databases.

Q.1.(b) What are the types of mapping constraints? **(2.5)**

Ans. An E-R schema may define certain constraints to which the content of a database must confirm. Two of the most important types of constraints are mapping cardinalities and existence dependencies.

Mapping cardinalities: mapping cardinalities, or cardinality ratios, express the number of entities of which another entity can be associated via a relationship set.

Mapping cardinalities are most useful in describing of relationship sets that involves more than two entity sets.

For binary relationship set R between sets A sets B, the mapping cardinality must be one of the following.

One to one: An entity in A is associated with any number of entities in B and an entity in B is associated with at most one entity in A.

One to many: A entity in A is associated with any number of entities in B. An entity in B. An entity in B, however, can be associated with at most one entity in A.

Many to one: An entity in A is associated with at most one entity in B, and an entity in B, however can be associated with any number of entities in A.

Many to many: An entity In A associated with any number of entities in B, and an entity in B is associated with any number of entities in A.

Q.1.(c) What are the types of Data Independence? **(2.5)**

Ans. Refer Q.2.(c) First Term Exam 2017.

Q.1.(d) Explain difference between a weak and a strong entity set? **(2.5)**

Ans.

BASIS FOR COMPARISON	STRONG ENTITY	WEAK ENTITY
Basic	The Strong entity has a primary key.	The weak entity has a partial discriminator key.
Depends	The Strong entity is independent of any other entity in a schema.	Weak entity depends on the strong entity for its existence.
Denoted	Strong entity is denoted by a single rectangle.	Weak entity is denoted with the double rectangle.
Relation	The relation between two strong entities is denoted by a single diamond simply called relationship.	The relationship between a weak and a strong entity is denoted by Identifying Relationship denoted with double diamond.
Participation	Strong entity may or may not have total participation in the relationship.	Weak entity always has total participation in the identifying relationship shown by double line.

Q.1.(e) What is the need of Normalization? **(2.5)**

Ans. Need of Normalization:

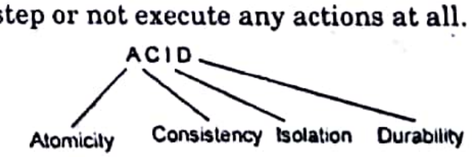
(i) They should have a minimal number of attributes necessary to suport the data requirements of the enterprise.

(ii) The attributes with a close logical relationship are found in the same relation.

(iii) It has a minimal redundancy, which requires that the values of no attribute of a database instance is replicated except where tuples are linked by foreign keys.

Q.1.(f) What are ACID properties? Explain in brief. **(2.5)**

Ans. Atomictiy: A transaction is said to be atomic if a transaction always executes all of its actions in one step or not execute any actions at all.



Consistency: A transaction must preserve the consistency of a database after its execution. It means that the execution of a transaction must leave a database in either its prior stable state or new stable state.

Isolation: It means that the data used during the execution of a transaction cannot be used by a second transaction until the first one is completed. The transactions do not interface with each other.

Durability: The effect of a completed or committed transaction should persist even after a system's failure (crash). It means that once a transaction committs, the system must guarantee that result of its operations will never be lost.

S.No.	Property	Meaning
1.	Atomicity	All or nothing
2.	Consistency	No violation of integrity constraints
3.	Isolation	Concurrent changes invisible
4.	Durability	Committed update persist.

Scanned by CamScanner

Q.1.(g) Explain the Thoma's write Rule? **(2.5)**

Ans. Thoma's write rule which states that "in order to increase the concurrency and to reduce the rollback of time stamp based protocols, obsolete write operations can be ignored under certain circumstances."

So, the protocol rules for read operations remain unchanged whereas for write operations, they are slightly different from our timestamp ordering protocol.

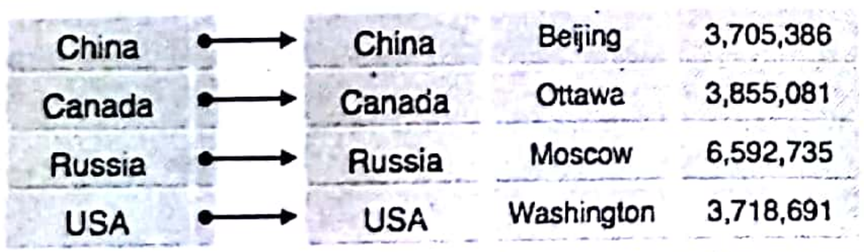
To understand the concept of Thoma's write rule let as consider the following schedule:

T_1	T_2
read(A,a)	
	write(A,a)
write(A,a)	

Apply the timestamp ordering protocol on above schedule, since T_1 starts before T_2, so TS (T_1) < TS (T_2). The read (A, a) operation of T_1 succeeds, as does the write (A, a) operation of T_2. When T_1 attempts its write (A, a) operations, we find that TS (T_1) < w - timestamp (A), since w - timestamp (A) = TS (T_2). Thus write (A, a) by T_1 is rejected and T_1 is required by the timestamp ordering protocol it is unnecessary. Since T_2 has already written A, the value of T_1 is attempting to write is one that will never need to be read. Any transaction Ti with TS (Ti) < TS (T_1) that attempts a read (A) will be rolled back, since TS(Ti) < w - timestamp (A) and any transaction Tj with TS (Tj) > TS (T_2) must read value of A written by T_2, rather than the value written by T_1.

Q.1.(h) What are Dense and Sparse Indices? **(2.5)**

Ans. Dense Index: In dense index, there is an index record for every search key value in the database. This makes searching faster but requires more space to store index records itself. Index records contain search key value and a pointer to the actual record on the disk.



Sparse Index: In sparse index, index records are not created for every search key. An index record here contains a search key and an actual pointer to the data on the disk. To search a record, we first proceed by index record and reach at the actual location of the data. If the data we are looking for is not where we directly reach by following the index, then the system starts sequential search until the desired data is found.

Q.1.(i) What are Multi valued Dependency? Explain. **(2.5)**

Ans. MVD Formal definition

Let R be a relation scheme and X and Y are subsets of R,

X ->> Y

holds if for all pairs of tuples s and t in r, such that

s[X] = t[X] there also exist tuples u and v where

s[X] = t[X] = u[X] = v[X]

s[Y] = u[Y]

t[R-XY] = u[R-XY]

s[R-XY] = v[R-XY]

t[Y] = v[Y]

Multi-valued dependencies occur in pairs.

Example: JointAppoint(facId, dept, committee) assuming a faculty member can belong to more than one department and belong to more than one committee.

Table must list all combinations of values of department and committee for each facId.

MVD Example

Course ->> Instructor

Course ->> Text

Course(Y)	Instructor(X)	Text(R-XY)
Intro	Kruse	Intro to CS
Intro	Wright	Intro to CS
CS1	Thomas	Intro to Java
CS1	Thomas	CS Theory Survey
CS2	Rhodes	Java Data Structures
CS2	Rhodes	Unix
CS2	Kruse	Java Data Structures
CS2	Kruse	Unix

Q.1.(j) What are DDL, DML and DCL commands? Explain. **(2.5)**

Ans. DDL: "Data Defination Language". DDL is used to create and remote database objects.

For Example:

- Create
- ALTER
- DROP

DML: "Data Manipulation Language". DML is used to manipulate data in the database. Some commands are:

- INSERT
- DELETE
- UPDATE
- SELECT

DCL commands: COMMIT, ROLLBACK, GRANT and REVOKE.

Commit command: It is the transactional command used to save changes invoked by a transaction to the database. It saves all transactions to the database since the last COMMIT or ROLLBACK command Syntax is

COMMIT [WORK];

Where COMMIT is the keyword and work is optional keyword and it is used to make the command more user friendly.

Example DELETE FROM EMP

WHERE BASIC > 7000;

COMMIT;

After deletion, a COMMAND is issued to save the changes to the database, completing the transaction.

Scanned by CamScanner

Q.2.(a) Give the relational schemas: **(2×5=10)**

ENROLL (S#,C#SECTION)
TEACH(PROF, C#, SECTION)
ADVISE (PROF. S#)
PRE-REQ (C# , PRE-C#)
GRADES (S# C#, GRADE, YEAR)
STUDENT (S#, SNAME)
(Where, S#=student number, C#=course)
Give Queries expressed in SQL:
(i) List the students taking course with Smith or Jones.
(ii) List all students taking atleast one course that their advisor teaches.
(iii) List those professors who teach more than one section of the same course.
(iv) List all student's number and course number.
(v) List the student number and course number who got grade A.

Ans. (i) SELECT SIS NAME FROM EN ROLL As E1, STUDENT As S1,
where E1. S# = S1. S# AND.
C# IN (Select C# from ENROLL as E, STUDENT as S where S.Sname = 'SMITH' OR S.Sname = "Jones";
(ii) Select S.NAME FROM ENROLL As E, TEACH As T, ADVISE As A, STUDENT As S WHERE E.C# = T.C#,
T.PROF = A.PROF, A.S# = S.S#;
(iii) SELECT T1 PROF FROM TEACH as T1 TEACH or T2,
where T1. C# = T2.C# AND T1. SECTION <> T2 SECTION;
(iv) SELECT S# FROM STUDENT
UNION
SELECT C# FROM ENROLL;
(v) SELECT S#, C# FROM GRADES where GRADE = 'A';

Q.2.(b) What are Canonical Covers? Explain. **(2.5)**

Ans. (*i*) Set F: = E.

(*ii*) Replace each functional dependency $x \to \{A_1, A_2 \ldots A_n\}$ in *F by* the *n* functional dependencies : $x \to A_1$, $x \to A_2 \ldots x \to A_n$.

(*iii*) For each functional dependency $x \to A$ in F for each attribute B that is an element of X if $\{\{F - \{X \to A\}\} \cup \{(X - \{B\}) \to A\}\}$ is equivalent to F,

then replace $X \to A$ with $(X - \{B3\}) \to A$ in F.

(*iv*) For each remaining functional dependency $X \to A$ in F.

if $\{F - \{X \to A\}\}$ is equivalent to F, then remove $X \to A$ from F.

For example: Let the given set of FDs be E: $\{B \to A, D \to A, AB \to D\}$. We have to 1 find the minimum cover of E.

All above dependencies are in canonical form: so we have completed step of [Algorithm and can proceed to step (*ii*). In step (*ii*) we need to determine if $AB \to D$

any redundant attribute on the left hand side that is can it be replaced by ; $B \to D$dr $A \to D$?

• Since $B \to A$, by augmenting with B on both sides we have $BB \to AB$, or $B \to AB$ i). However, $AB \to D$ as given (b).

• Hence by the transitive Rule, we get from (a) and (b), $B \to D$. Hence $AB \to D$ *may* be replaced by $B \to D$.

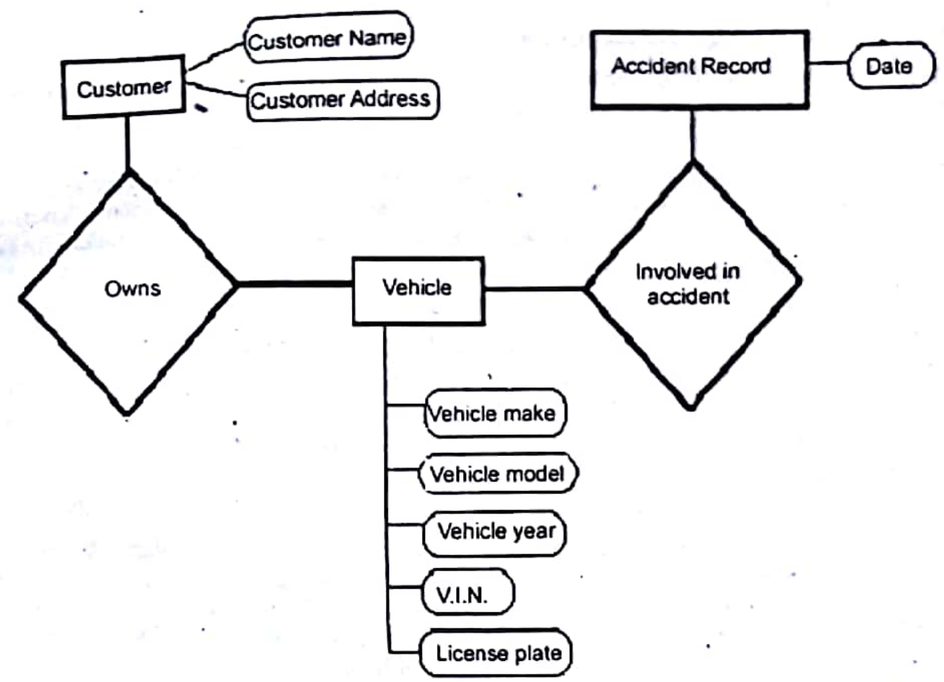
• We now have a self equivalent to original E, say E': $\{B \to A, D \to A, B \to D\}$. No further reduction is possible in step (*ii*) since all FDs have a single attribute on the left hand side.

• In step (*iii*) we look for a redundant FD in E, By using the transitive rule on $B \to D$ and $D \to A$, we derive $B \to A$. Hence, $B \to A$ is redundant in E' and can be eliminated.

$\Rightarrow$ Hence the minimum cover of E is $\{B \to D, D \to A\}$

Q.3.(a) Construct an E-R diagram for a car-insurance company that has a set of customers, each of whom owns one or more cars. Each car has an associated number indicating the number of recorded accidents. **(10)**

Ans.



Q.3.(b). Explain the difference between primary key, super key and candidate key? **(2.5)**

Ans. Refer Q.1.(a) First Term Exam 2017.

Q.4.(a) Given two sets F1 and F2 of FD's for a relation. **(5)**

F1:A → B, AB → C,D → AC, D → E
F2:A → BC,D → AE
Are the two sets equivalent?

Ans. If F1 covers F2 then every FD in F2 logically implies in F1.

FDs in F2 $A \to BC$, $D \to AE$

Check for $A \to BC$

$(A)^+ \to ABC$ $\{A \to B, AB \to C$ in F1$\}$

Check for $D \to AE$

$(D)^+ \to DACE$ $\{D \to AC, D \to E$ in F1$\}$

Hence F1 covers F2 (i.e., $F1 \supseteq F2$)

If F2 covers F1 then every FD in F1 logically implies in F2.

F1 has 4 FDs $\{A \to B, AB \to C, D \to AC, D \to E\}$

Check for $A \to B$

$(A)^+ \to AB$ $\{A \to BC$ in F2$\}$

Check for $AB \to C$

$(AB)^+ \to ABC$ $\{A \to BC$ in F2$\}$

Scanned by CamScanner

Check for D → AC

$(D)^+ \rightarrow DAEBC$ {D → AE, A → BC in F2}

Check for D → E

$(D)^+ \rightarrow DAEBC$ {D → AE in F2}

Hence F2 covers F1 i.e., F2 ⊇ F1

So F1 = F2

Q.4.(b) Consider the universal relation R={A, B, C, D, E, F, G, H, I, J} and set of FDs F = AB → C,A → DE,B → F,F→ GH,D → IJ

What is the key for R? Decompose R into 3NF. **(7.5)**

Ans. A minimal set of attributes whose closure includes all the attributes in R is a key. Since the closure of $\{A, B\}$, $\{A, B\}^+ = R$, one key of R is $\{A, B\}$ (in this case, it is the only key).

To normalize R intuitively into 2NF then 3NF, we take the following steps: First, identify partial dependencies that violate 2NF. These are attributes that are functionally dependent on either parts of the key. {A} or {B} alone. We can calculate the closures $\{A\}^+$ and $\{B\}^+$ to determine partially dependent attributes:

$\{A\}^+ = \{A, D, E, I, J\}$. Hence $\{A\} \rightarrow \{D, E, I, J\}$ ($\{A\} \rightarrow \{A\}$ is a trival dependency)

$\{B\}^+ = \{B, F, G, H\}$, hence $\{A\} \rightarrow \{F, G, H\}$ ($\{B\} \rightarrow \{B\}$ is trivial dependency)

To normalize into 2NF, we remove the attributes that are functionally dependent on part of the key (A or B) from R and place them in seperate relations R_1 and R_2, along with the part of the key they depend on (A or B), which are copied into each of these relations but also remains in the original relation, which we call R_3 below:

$R_1 = \{A, D, E, I, J\}, R_2 = \{B, F, G, H\}, R_3 = \{A, B, C\}$

The new keys for R_1, R_2, R_3 are underlined. Next, we look for transitive dependencies in R_1, R_2, R_3. The relation R_1 has the transitive dependency $\{A\} \rightarrow \{D\} \rightarrow \{I, J\}$, so we remove the transitively dependent attributes $\{I, J\}$ from R_1 into a relation R_{11} and copy the attribute D they are dependent on R_{11}. The remaining attributes are kept in a relation R_{12}. Hence, R_1 is decomposed into R_{11} and R_{12} as follows:

$R_{11} = \{D, I, J\}, R_{12} = \{A, D, E\}$

The relation R_2 is similarly decomposed into R_{21} and R_{22} based on the transitive dependency.

$\{B\} \rightarrow \{F\} \rightarrow \{G, H\}$:

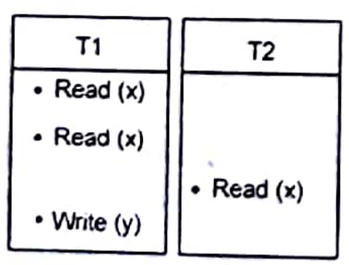
$R_{21} = \{F, G, H\}, R_{22} = \{B, F\}$

The final set of relations in $3NF$ are $\{R_{11}, R_{12}, R_{21}, R_{22}, R_3\}$

Q.5.(a) What are recoverable and cascadeless schedules? Explain **(5)**

Ans. RECOVERABLE SCHEDULE

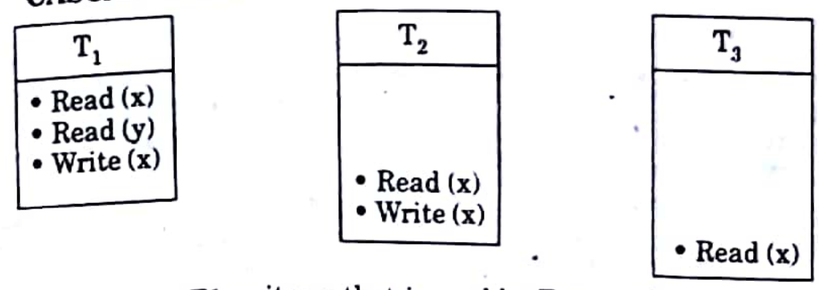
A recoverable schedule is one where, for each pair of Transaction Ti and Tj such that if Tj reads data item previously written by Ti , then the commit operation of Ti appears before the commit operation Tj.



Suppose after T2 Read(x) operation, it commits. And then somehow T1 fails. So the transaction T2 must be aborted so as to ensure atomicity. However, since T2 is commited , and can't be aborted . Hence a situation arrives where it is impossible to recover.

And hence its necessary that Ti commits before Tj.

CASCADELESS SCHEDULE:



Transaction T1 writes x that is read by Transaction T2. Transaction T2 writes x that is read by Transaction T3. Suppose at this point T1 fails. T1 must be rolled back, since T2 is dependent on T1, T2 must be rolled back,and since T3 is dependent on T2, T3 must be rolled back.

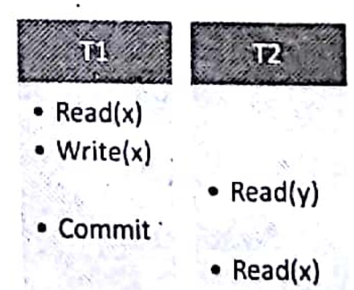
This phenomenon, in which a single transaction failure leads to a series of transaction rollbacks is called Cascading rollback.

Cascading rollback is undesirable, since it leads to the undoing of a significant amount of work.

It is desirable to restrict the schedules to those where cascading rollbacks cannot occur, Such schedules are called Cascadeless Schedules.

Formally, a cascadeless schedule is one where for each pair of transaction Ti and Tj such that Tj reads data item, previously written by Ti the commit operation of Ti appears before the read operation of Tj .

Cascadeless Schedule Example:



Q.5.(b) What do you understand by lock compatibility? Discuss in detail with example? **(7.5)**

Ans. Compatibility of Locks:

Suppose that there are A and B two different locking modes. If a transaction T1 requests a lock of mode on item Q on which transaction T2 currently hold a lock of mode B. If transaction can be granted lock, in spite of the presence of the mode B lock, then we say mode A is compatible with mode B. Such a function is shown in one matrix as shown below:

	S	X
S	true	false
X	false	false

Compatibility Graph

The graphs shows that if two transactions only read the same data object they do not conflict, but if one transaction writes a data object and another either read or write the same data object, then they conflict with each other. A transaction requests a shared

Scanned by CamScanner

lock on data item Q by executing the lock-S(Q) instruction. Similarly, an exclusive lock is requested through the lock- X(Q) instruction. A data item Q can be unlocked via the unlock(Q) instruction

To access a data item, transaction T1 must first lock that item. If the data item is already locked by another transaction in an incompatible mode, the concurrency control manager will not grant the lock until all incompatible locks held by other transactions have been released Thus, T1 is made to wait until all incompatible locks held by other transactions have been released

Example: As an illustration consider the simplified banking system. Let A and B be two accounts that are accessed by transactions TI and T2. Transaction TI transfers Rs.50 from account B to account a and is defined as:

T1
lock-X(B);
read(B,b);
b := b-50;
Write (B,b);
unlock (B) ;
lock-x(A)
read(A,a);
a:=a+50;
Write(A,a);
unlock (A);

Transaction T2 displays the total amount of money in accounts A and B that is, the sum A+B and is defined as

T2
lock-S(A);
read(A,a);
unlock(A);
lock-S(B);
read(B,b);
unlock(B);
display(a+b);

Suppose that the values of accounts A and B are Rs.100 and Rs.200, respectively. If these two transactions are executed serially, either in the order T 1 and T2 or the order T2, T 1 then transaction T2 will display the value Rs.300. If however, these transactions are executed concurrently, as shown in Schedule 1 In this case, transaction T2 displays Rs.250, which is incorrect. The reason for this mistake is that the transaction TI unlocked data item B too early, as a result of which T2 shows an inconsistent state.

Solution of Inconsistency Problem:

Suppose now that unlocking is delayed to the end of the transaction. The transaction T3 corresponds to T1 with unlocking delayed and is defined as

T3
lock-X(B);
read(B,b);
b:=b-50 ;
Write(B,b) ;
lock -X(A) ;
. read(A,a) ;
a:=a+50 ;
write (A,a) ;
unlock(B);
unlock(A) ;

Transaction T4 corresponds to T2 with unlocking delayed, and is defined as

T4
lock-S(A);
read(A,a);
lock-S(B);
read (B,b);
display (a+b);
unlock(A);
unlock [B]

Scanned by CamScanner

You should verify that the sequence of reads and writes in schedule I which leads to an incorrect total of Rs 250 being displayed, is no longer possible with T3 and T4 as shown in Schedule

T3	T4
look - X(B)	
read(B)	
B := B-50	
Write(B)	
	lock -S(A)
	read(A)
	lock - S(B)
	wait..
	wait...
lock-X(A)	
wait...	
wait...	

Q.6.(a) Discuss the basic time stamp ordering protocol? Also explain the values asssociated with timestamps? (7.5)

Ans. Timestamp: This protocol works on a principle that -"we must have a prior knowledge about the order in which the transactions will be accessed". In graph based protocol, we require the knowledge of ordering of data items in the transactions.

How to do it?

To order the transactions, we assosiate a unique fixed timestamp, denoted by TS(Ti) for each transaction, Ti. This time stamp is an integer (unique) value. It is assigned by DBMS before the transaction Ti, starts execution.

Timestampes have 2 properties:

1. Uniqueness: This property assures that no equal timestamp values can exist.

2. Monotonicity: It means that timestamp values always increase.

Timestamp-Ordering Protocol: It ensures that any conflicting read and write operations are executed in timestamp order. This protocol operates as follows.

1. Suppose that transaction Ti issue read (Q):

(a) If TS (Ti) < w-timestamp (Q), then Ti needs to read a value of Q that was already overwritten. Hence the read operation is rejected, and Ti is rolled back.

(b) If TS (Ti) >= w-timestamp .(Q), then the read operation is executed, and R-timestamp (Q) is set to the maximum of R-timestamp (Q) i.e. TS (Ti).

2. Suppose that transaction Ti issues write (Q):

(a) If Ts (Ti) < R-timestamp (Q), then the value of Q that Ti is producing was needed previously and the system assumed that value would never be produced. Hence, the write operation is rejected, and Ti is rolled back.

(b) If TS (Ti) < w-timestamp (Q), then Ti is attempting to write an absolute value of Q. Hence the write operation is rejected and Ti is rolled back.

Q.6.(b) Discuss the 'Wait-die' and 'wound-wait' techniques of deadlock prevention. (5)

Ans. Wait Die Scheme: If Ti has a higher priority, it is allowed to wait else it is aborted. This means that when Ti request for a data-item that is currently being held by T_j, then T_i is allowed to wait only if it has a timestamp smaller than that of T_j is older than T_j otherwise, T_i is rolled back (dies).

For Example: Say, T_1 requests for a data item held by T_2 and T_3 also requests for data item held by T_2 i.e.

T_1 requests → [T_2] ← T_3 requests

Wound-Wait Method: If T_i transaction has a higher priority then we abort T_j else T_i wait. It means that when T_i requests a data-item currently held by T_j, then T_i is allowed to wait only if it has a Ts (T_i) > Ts (T_j). Else T_j is rolled back.

For Example: If T_1 requests a data-item held by T_2 then the data-item will be preempted from T_2 and T_2 will be rolled back i.e.,

$T_1 \rightarrow T_2$ (Rollbacks)

Now, if T_3 requests a data-item held by T_2, then T_3 waits

$T_2 \leftarrow T_3$ (waits)

Wait-die Scheme	Wound-Wait Scheme
(i) This scheme is Non-Preemptive	(i) This scheme is Preemptive.
(ii) Lower priority transactions can never wait for higher transactions.	(ii) The higher priority transaction never wait for lower priority transaction.

Q.7.(a) Explain check points. How does it helps in reducing the amount of time required during Recovery? Also discuss the concept of fuzzy check pointing? (6.5)

Ans. When the system restarts after a failure, the entire log needs to be scanned to determine the transactions that need to be redone and undone. The main disadvantage of this approach is that scanning of entire log is time consumming. Moreover, the committed transactions that have already written their updates on the database also need to be redone as there is no way to find out the extent to which the changes of committed transactions have been propagated to the database. This causes recovery to take longer time.

To reduce the time for recovery by avoiding redo of the unnecessary work, checkpoints are used. The checkpoint approach is an additional component of the logging scheme. A **checkpoint** is a record periodically written into the log. It is typically written at a point when the system writes out all the modified buffers to the database on the disk. As a result, the transactions that have committed prior to the checkpoint will have their [T_1, commit] record before the [checkpoint] record. The write operations of these transactions need not to be redone in case of a failure as all updates are already recorded in the database on the disk. When a checkpoint is taken, following actions are performed.

1. The execution of the currently running transaction is suspended.
2. All the log records currently residing in the main memory are written to the stable storage.
3. All the modified buffer blocks are force-written to the disk.
4. A [checkpoint] record is written to the log on the stable storage.
5. The execution of suspended transaction is resumed.

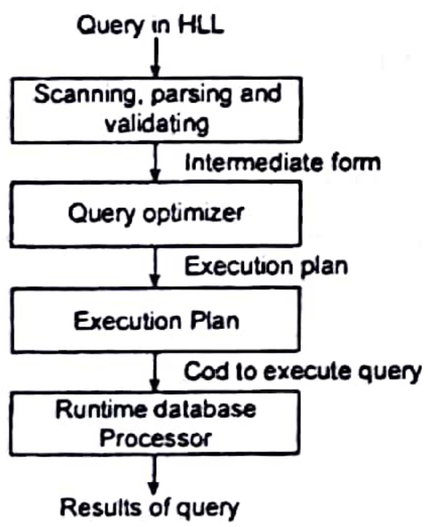
Scanned by CamScanner

Since the transactions committed before the checkpoint time need not be considered during recovery process, it reduces the amount of work required to be done during recovery.

Fuzzy checkpointing: If the number of blocks in the buffer is large force-writing of all these pages may take long time to finish. The time taken to force-write all modified buffers can result in undsirable delay in the processing of current transactions. For reducing this delay, a technique called **fuzzy checkpointing** is used, In fuzzy checkpointing, a [checkpoint] record (known as **fuzzy checkpoint**) is written in the log before writing the modified buffer blocks to the disk. The system is then allowed to resume the execution of other transactions without having to wait for the completion of step 3. However, a system failure may occur while modified buffer blocks are being written to the disk. In that case, the fuzzy checkpoint would no longer be valid. Thus, the system is required to keep track of the previous checkpoint for which all the actions (1 to 5) have been successfully performed. For this, the system maintains a pointer to that checkpoint and it is updated to point to new checkpoint only when all the modified buffer blocks have been written to the disk.

Q.7.(b) What do you mean by Query processing? What are the various steps involved in Query Processing? Explain with the help of a block diagram. **(6)**

Ans. The procedure of selecting the best plan or strategy to be used in responding to a database request and executing the plain to generate the response is known as query processing. It is a stepwise process. The SQL query is first of all scanned, parsed and validated. Let us draw a flow chart now.



Flowcharts for query processing

The scanner identifies the language takens such as SQL key words, attribute names and relation names in the text of query. The parser checks the query syntax to determine whether it is formulated according to the syntax rules of the query language. The query must also be validated by checking that all attributes and relation names are valid. An internal representation of the query is then created usually as a tree data structure called as a query-tree.

Q.8.(a) Write short notes on: **(2.5×5 = 12.5)**

(a) Database Tuning:

Ans. The DBMS is the system layer that an application must access directly. Uniface interacts with a DBMS via a DBMS-specific connector. The DBMS contributes to the system load, so it is an area of interest for performance optimization.

You should tune the DBMS after the application is optimized, because a DBMS often serves other applications too, and might already be optimized for those situation. DBMS tuning is a vast area on its own and requires DBMS-specific expertise. Areas that can be addressed include:

Partitioning the DBMS in the infrastructure

Scaling and balancing the various DBMS processes

Size and structure of shared (real) memory

Size and structure of the DBMS files

Splitting volumes, such as data and indexes for the same table on separate disks.

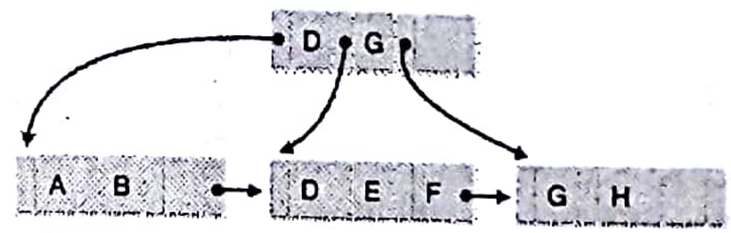
You should also pay particular attention to tuning the Uniface driver for the DBMS. Issues that play a role include a number of parameters for managing real memory required for communicating with the DBMS.

Q.8.(b). B⁺ Tree Index File

Ans. B⁺ Tree:

A B⁺ tree is a balanced binary search tree that follows a multi-level index format. The leaf nodes of a B⁺ tree denote actual data pointers. B⁺ tree ensures that all leaf nodes remain at the same height, thus balanced. Additionally, the leaf nodes are linked using a link list; therefore, a B⁺ tree can support random access as well as sequential access.

Structure of B⁺ Tree: Every leaf node is at equal distance from the root node. A B⁺ tree is of the order **n** where **n** is fixed for every B⁺ tree.



Internal nodes:

- Internal (non-leaf) nodes contain at least [$n/2$] pointers, except the root node.
- At most, an internal node can contain **n** pointers.

Leaf nodes:

- Leaf nodes contain at least [$n/2$] record pointers and [$n/2$] key values.
- At most, a leaf node can contain **n** record pointers and **n** key values.
- Every leaf node contains one block pointer **P** to point to next leaf node and forms a linked list.

B⁺ Tree Insertion

- B⁺ trees are filled from bottom and each entry is done at the leaf node.
- If a leaf node overflows–
  - Split node into two parts.
  - Partition at $i = [(m+1)_{/2}]$.
  - First **i** entries are stored in one node.
  - Rest of the entries (i+1 onwards) are moved to a new node.
  - i^{th} key is duplicated at the parent of the leaf.
- If a non-leaf node overflows"
  - Split node into two parts.
  - Partition the node at $i = [\ (m+1)_{/2}\ ]$.
  - Entries up to **i** are kept in one node.
  - Rest of the entries are moved to a new node.

B⁺ Tree Deletion

- B⁺ tree entries are deleted at the leaf nodes.
- The target entry is searched and deleted.
  - If it is an internal node, delete and replace with the entry from the left position.

Scanned by CamScanner

- After deletion, underflow is tested.
  - If underflow occurs, distribute the entries from the nodes left to it.
- If distribution is not possible from left, then
  - Distribute from the nodes right to it.
- If distribution is not possible from left or from right, then
  - Merge the node with left and right to it.

Q.8.(c) Join dependency

Ans. Join Dependency: A Join Dependency (JD) denoted by JD $(R_1, R_2, ..., R_n)$, specified on relation scheme R, specifies a constraint on the states r of R. The constraint states that every legal state r of R should have a non additive join decomposition into $R_1, R_2, ... R_n$ that is, for every such row have

$$(\pi R_1(r), \pi R_2(r), ..., \pi R_n(r)) = r$$

Notice that an MVD is a special case of a JD where $n = 2$. That is, a JD denoted as JD (R_1, R_2) implies an MVD $(R_1 \cap R_2) \rightarrow (R_1 - R_2)$. A join Dependency JD $(R_1, R_2, ... R_n)$, specified on relation scheme R, is a trivial JD if one of the relation schemes R_1 in JD $(R_1, R_2 ... R_n)$ is equal to R. Such a dependency is called trivial because it has the non-additive join property for any relation state r of R and hence does not specify any constraint on R.

Fifth Normal Form: A relation scheme R is in Fifth Normal Form (SNF) [or Project Join Normal Form (PJNF)) with respect to a set F of functional, multivalued and join dependencies, if for every non-trivial join dependency JD $(R_1, R_2, ... R_n)$ in F^+ (that is, implied by F), every R_i is a super key of R.

For example: Consider the SUPPLY all-key relation of Figure 1. Suppose that the following additional constraint always holds: whenever a suppliers supplies part p, and a project j uses part p, and the suppliers supplies at least one part to project j, then suppliers s will also be supplying part p to project j. This constraint can be restarted in other ways and specifies a join dependency JD (R_1, R_2, R_3) among the three projections R_1 (Sname, Part-name), R_2 (Sname, Proj-name) and R_3 (Part-name, Proj-name) of SUPPLY. If this constraint holds, the tuple below the dotted line in Figure 1 must exist in any legal state of the SUPPLY relation that also contains the tuples above the dotted line. Figure 2 shows how the supply relation with the join dependency is decomposed into three relations R_1, R_2 and R_3 that are each in SNF.

Supply

Sname	Part-name	Proj-name
Smith	Bolt	Proj X
Smith	Nut	Proj Y
Adanisky	Bolt	Proj Y
Walton	Nut	Proj Z
Adamsky	Nail	Proj X
Adamsky	Bolt	Proj X
Smith	Bolt	Proj Y

Fig. 1

R1

Sname	Proj-name
Smith	Bolt
Smith	Nut
Adamsky	Bolt
Walton	Nut
Adamsky	Nail

R2

Sname	Proj-name
Smith	Proj X
Smith	Proj Y
Adamsky	Proj Y
Walton	Proj Z
Adamsky	Proj X

R3

Part-name	Proj-name
Bolt	Proj X
Nut	Proj Y
Bol:	Proj Y
Nut	Proj Z
Nail	Proj X

Fig. 2

Q.8.(d) PL/SQL concept

Ans. PL SQL basically **stands** for "Procedural Language extensions to SQL". This is the extension of Structured Query Language (SQL) that is used in Oracle.

It combines the data manipulation power of SQL with the processing power of procedural language to create a super powerful SQL queries.

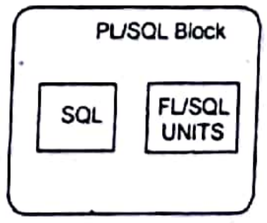
Architecture of PL/SQL

The PL/SQL architecture mainly consists of following 3 components:

1. PL/SQL block 2. PL/SQL Engine 3. Database Server

PL/SQL block:

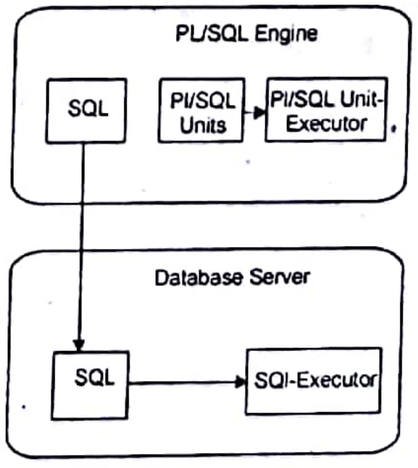
- This is the component which has the actual PL/SQL code.
- This consists of different sections to divide the code logically (declarative section for declaring purpose, execution section for processing statements, exception handling section for handling errors)
- It also contains the SQL instruction that used to interact with the database server.
- All the PL/SQL units are treated as PL/SQL blocks, and this is the starting stage of the architecture which serves as the primary input.
- Following are the different type of PL/SQL units.



o Anonymous Block o Function
o Library o Procedure
o Package Body o Package Specification
o Trigger o Type
o Type Body

PL/SQL Engine

- PL/SQL engine is the component where the actual processing of the codes takes place.
- PL/SQL engine separates PL/SQL units and SQL part in the input (as shown in the image below).
- The separated PL/SQL units will be handled with the PL/SQL engine itself.
- The SQL part will be sent to database server where the actual interaction with database takes place.
- It can be installed in both database server and in the application server.



Scanned by CamScanner

Database Server:

- This is the most important component of Pl/SQL unit which stores the data.
- The PL/SQL engine uses the SQL from PL/SQL units to interact with the database server.
- It consists of SQL executor which actually parses the input SQL statements and execute the same.

Q.8.(e) Relational Algebra

Ans. Relational Algebra

- Relational Algebra operations manipulate some relations and provide some expression in the form of queries.
- Relational Algebra have operator like join, Union, Intersection, Division, Difference, Projection, Selection etc.
- Relational Algebra is procedural language.
- Relational Algebra queries are more powerful.

Q.9. Given the following relations:

Vehicle (Reg No., Make, Colour)

Person (ENO, Name, Address)

Owner (ENo, Reg No.)

Write expressions in relational algebra or SQL to answer the following queries.

(i) List the names of persons who do not own any car.

(ii) List the names of persons who own only Maruti Cars.

Ans. (i) SELECT PERSON. name FROM
VEHICLE, PERSON, OWNER
WHERE VEHICLES. reg-no= OWNER. reg. no
AND PERSON. eno = OWNER. eno
AND VEHICLE. reg. no. <> 'NOTHING'

(ii) SELECT PERSON. name FROM
VEHICLE, PERSON, OWNER
WHERE VEHICLE. reg no = OWNER. reg. no
AND PERSON. eno = OWNER, eno
AND VEHICLE, reg-no = 'MARUTI';

Q.9. (b) Discuss division operation in relational algebra giving an exampl

Ans. Division: The division operator divides a dividend relation, A, of degree (m+ by a divisior relation, B, of degree, n, and produces a resultant relation of degree, m

For Example: Consider the following relation.

Relation–A	Relation B	Result
		A Divide by B
Sno / *Pno* *S1* / *P1* *S1* / *P2* *S2* / *P1* *S2* / *P2*	*Pno* P_1	*Sno* *S1* *S2*