SYLLABUS
DATABASE MANAGEMENT SYSTEM (CIC-210)
Applicable from the Academic Session 2021-22

Marking Scheme:

1. Teachers Continuous Evaluation: 25 marks
2. Term end Theory Examinations: 75 marks

Instructions for paper setter:

1. There should be 9 questions in the term end examinations question paper.
2. The first (1st) question should be compulsory and cover the entire syllabus. This question should be objective, single line answers or short answer type question of total 15 marks.
3. Apart from question 1 which is compulsory, rest of the paper shall consist of 4 units as per the syllabus. Every unit shall have two questions covering the corresponding unit of the syllabus. However, the student shall be asked to attempt only one of the two questions in the unit. Individual questions may contain upto 5 sub parts / sub-questions. Each Unit shall have a marks weightage of 15.

UNIT I

Basic concepts: database & database users, characteristics of the database systems, concepts and architecture, date models, schemas & instances, DBMS architecture & data independence, database languages & interfaces, data modelling using the entity-relationship approach. Enhanced ER concepts - Specialization/Generalization, Aggregation, Mapping of ER model to Relational Model.

SQL - DDL, DCL & DML views and indexes in SQL. Basics of SQL, DDL, DML,DCL, structure - creation, alteration, defining constraints - Primary key, foreign key, unique, not null, check, IN operator.

UNIT II

Relational model concepts, relational model constraints, relational algebra, relational calculus.

SQL- Functions - aggregate functions, Built-in functions - numeric, date, string functions, set operations, sub-queries, correlated sub-queries, Use of group by, having, order by, join and its types, Exist, Any, All, view and its types. Transaction control commands-Commit, Rollback, Save point.

UNIT III

Relational data base design: functional dependencies & normalization for relational databases, normal forms based on functional dependencies, (INF, 2NF, 3NF & BCNF), lossless join and dependency preserving decomposition, normal forms based on multivalued & join dependencies (4NF & 5NF) & domain key normal form.

Properties of Transaction, Transaction states, Transaction Schedule, Serializability, Concurrency control techniques, locking techniques, time stamp ordering, Recoverable schedules, granularity of data items, Deadlock detection and Recovery, recovery techniques: recovery concepts, database backup and recovery from catastrophic failures.

Database Programming – control structures, exception handling, stored procedures, Triggers.

UNIT IV

File Structures and Indexing: Secondary Storage Devices, Operations on Files, Heap Files, Sorted Files, Hashing, Single level indexes, Multi-level indexes, B and B+ tree indexes.

Concepts of Object Oriented Database Management systems & Distributed Database Management Systems

syllabusx.live

SYLLABUS

[Academic Session 2015-16]

DATABASE MANAGEMENT SYSTEM
(ETCS-208)

Instruction to Paper Setters: **Maximum Marks : 75**

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.

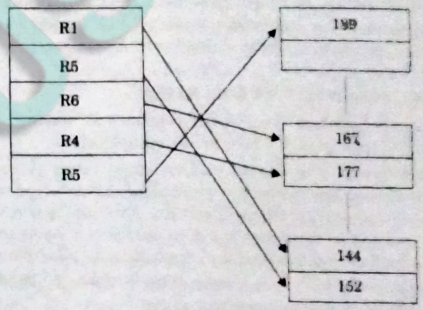
Implementation Techniques: Overview of Physical Storage Media, File Organization, Indexing and Hashing, B+ tree Index Files, Query Processing Overview, Catalog Information for Cost Estimation, Selection Operation, Sorting, Join Operation, Materialized views, Database Tuning.

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

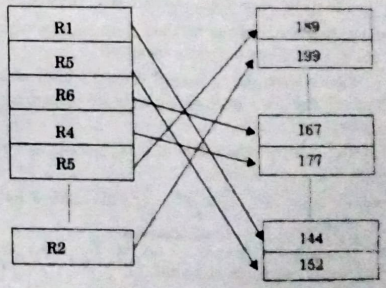
NEW TOPICS ADDED FROM ACADEMIC SESSION 2021-22 ONWARDS
FOURTH SEMESTER
DATABASE MANAGEMENT SYSTEM [CIC-210]

Q.1. What do you mean by the Heap file Organization?

Ans. Heap File Organization – Heap File Organization works with data blocks. In this method records are inserted at the end of the file, into the data blocks. No Sorting or Ordering is required in this method. If a data block is full, the new record is stored in some other block, Here the other data block need not be the very next data block, but it can be any block in the memory. It is the responsibility of DBMS to store and manage the new records.



Insertion of new record – Suppose we have four records in the heap R1, R5, R6, R4 and R3 and suppose a new record R2 has to be inserted in the heap then, since the last data block i.e data block 3 is full it will be inserted in any of the data blocks selected by the DBMS, let's say data block 1.



If we want to search, delete or update data in heap file Organization the we will traverse the data from the beginning of the file till we get the requested record. Thus if the database is very huge, searching, deleting or updating the record will take a lot of time.

syllabusx.live

Q.2. What are the Pros and Cons of Heap File Organization?

Ans. Pros –

• Fetching and retrieving records is faster than sequential record but only in case of small databases.

• When there is a huge number of data needs to be loaded into the database at a time, then this method of file Organization is best suited.

Cons –

• Problem of unused memory blocks.

• Inefficient for larger databases.

Q.3. What is the Object oriented database management system and what are its components?

Ans. The **ODBMS** which is an abbreviation for **object-oriented database management system** is the data model in which data is stored in form of objects, which are instances of classes. These classes and objects together make an object-oriented data model.

Components of Object-Oriented **Data Model:**

The OODBMS is based on three major components, namely: Object structure, Object classes, and Object identity. These are explained below.

1. Object Structure: The structure of an object refers to the properties that an object is made up of. These properties of an object are referred to as an attribute. Thus, an object is a real-world entity with certain attributes that makes up the object structure. Also, an object encapsulates the data code into a single unit which in turn provides data abstraction by hiding the implementation details from the user.

The object structure is further composed of three types of components: Messages, Methods, and Variables. These are explained below.

(a) Messages – A message provides an interface or acts as a communication medium between an object and the outside world. A message can be of two types:

• **Read-only message:** If the invoked method does not change the value of a variable, then the invoking message is said to be a read-only message.

• **Update message:** If the invoked method changes the value of a variable, then the invoking message is said to be an update message.

(b) Methods – When a message is passed then the body of code that is executed is known as a method. Whenever a method is executed, it returns a value as output. A method can be of two types:

• **Read-only method:** When the value of a variable is not affected by a method, then it is known as the read-only method.

• **Update-method:** When the value of a variable change by a method, then it is known as an update method.

(c) Variables – It stores the data of an object. The data stored in the variables makes the object distinguishable from one another.

2. Object Classes: An object which is a real-world entity is an instance of a class. Hence first we need to define a class and then the objects are made which differ in the values they store but share the same class definition. The objects in turn correspond to various messages and variables stored in them.

Example –

```
class CLERK
 { //variables
   char name;
   string address;
   int id;
   int salary;
  //Messages
   char get_name();
   string get_address();
   int annual_salary();
 };
```

In the above example, we can see, CLERK is a class that holds the object variables and messages.

An OODBMS also supports inheritance in an extensive manner as in a database there may be many classes with similar methods, variables and messages. Thus, the concept of the class hierarchy is maintained to depict the similarities among various classes. The concept of encapsulation that is the data or information hiding is also supported by an object-oriented data model. And this data model also provides the facility of abstract data types apart from the built-in data types like char, int, float. ADT's are the user-defined data types that hold the values within them and can also have methods attached to them.

Thus, OODBMS provides numerous facilities to its users, both built-in and user-defined. It incorporates the properties of an object-oriented data model with a database management system, and supports the concept of programming paradigms like classes and objects along with the support for other concepts like encapsulation, inheritance, and the user-defined ADT's (abstract data types).

Q.4. What is the distributed database management system and explain its types?

Ans. A distributed database is basically a database that is not limited to one system, it is spread over different sites, i.e, on multiple computers or over a network of computers. A distributed database system is located on various sites that don't share physical components. This may be required when a particular database needs to be accessed by various users globally. It needs to be managed such that for the users it looks like one single database.

Types:

1. Homogeneous Database: In a homogeneous database, all different sites store database identically. The operating system, database management system, and the data structures used – all are the same at all sites. Hence, they're easy to manage.

2. Heterogeneous Database: In a heterogeneous distributed database, different sites can use different schema and software that can lead to problems in query processing and transactions. Also, a particular site might be completely unaware of the other sites. Different computers may use a different operating system, different database application. They may even use different data models for the database. Hence, translations are required for different sites to communicate.

Q.5. How can we store the data in distributed database management system?

Ans. Distributed Data Storage :

There are 2 ways in which data can be stored on different sites. These are:

syllabusx.live

(a) Replication – In this approach, the entire relationship is stored redundantly at 2 or more sites. If the entire database is available at all sites, it is a fully redundant database. Hence, in replication, systems maintain copies of data.

This is advantageous as it increases the availability of data at different sites. Also, now query requests can be processed in parallel.

However, it has certain disadvantages as well. Data needs to be constantly updated. Any change made at one site needs to be recorded at every site that relation is stored or else it may lead to inconsistency. This is a lot of overhead. Also, concurrency control becomes way more complex as concurrent access now needs to be checked over a number of sites.

(b) Fragmentation – In this approach, the relations are fragmented (i.e., they're divided into smaller parts) and each of the fragments is stored in different sites where they're required. It must be made sure that the fragments are such that they can be used to reconstruct the original relation (i.e, there isn't any loss of data).

Fragmentation is advantageous as it doesn't create copies of data, consistency is not a problem.

Fragmentation of relations can be done in two ways:

- **Horizontal fragmentation – Splitting by rows –** The relation is fragmented into groups of tuples so that each tuple is assigned to at least one fragment.
- **Vertical fragmentation – Splitting by columns –** The schema of the relation is divided into smaller schemas. Each fragment must contain a common candidate key so as to ensure a lossless join.

In certain cases, an approach that is hybrid of fragmentation and replication is used.

Q.6. What are the applications of Distributed database management system?

Ans. Applications of Distributed Database:

- It is used in Corporate Management Information System.
- It is used in multimedia applications.
- Used in Military's control system, Hotel chains etc.
- It is also used in manufacturing control system.

ACCORDING TO NEW SYLLABUS (CIC-210)

UNIT - I

Q.1. Refer to Q.2 (a) of First Term Examination 2015 (Pg. No. 1-2015)
Q.2. Refer to Q.1 (a), (c), (e) of End Term Examination 2015 (Pg. No. 8-2015)
Q.3. Refer to Q.1 (a), (b), (c), (d) of First Term Examination 2016 (Pg. No. 1,2-2016)
Q.4. Refer to Q.3 (a), (b) of First Term Examination 2016 (Pg. No. 5-2016)
Q.5. Refer to Q.4 (b),(c) of First Term Examination 2016 (Pg. No. 6,7-2016)
Q.6. Refer to Q.1 (d), (e), (f) of End Term Examination 2016 (Pg. No. 16,17-2016)
Q.7. Refer to Q.2 (b) of End Term Examination 2016 (Pg. No. 17-2016)
Q.8. Refer to Q.5 (a),(b) of End Term Examination 2016 (Pg. No. 23-2016)
Q.9. Refer to Q.6 (b) of End Term Examination 2016 (Pg. No. 25-2016)
Q.10. Refer to Q.3 of Important Questions 2016 (Pg. No. 32-2016)
Q.11. Refer to Q.1 (a),(b),(c) of First Term Examination 2017 (Pg. No. 1-2017)
Q.12. Refer to Q.2 (b) of First Term Examination 2017 (Pg. No. 3-2017)
Q.13. Refer to Q.1 (a),(b),(d),(j) of End Term Examination 2017 (Pg. No. 8,9,11-2017)

Q.14. Refer to Q.3 (a) of End Term Examination 2017 (Pg. No. 13-2017)
Q.15. Refer to Q.7 (b) of End Term Examination 2017 (Pg. No. 20-2017)
Q.16. Refer to Q.8 (a) of End Term Examination 2017 (Pg. No. 20-2017)
Q.17. Refer to Q.1 (a),(b),(e) of First Term Examination 2018 (Pg. No. 1,2-2018)
Q.18. Refer to Q.2 (a),(c) of First Term Examination 2018 (Pg. No. 3-2018)
Q.19. Refer to Q.1 (a) of End Term Examination 2018 (Pg. No. 9-2018)
Q.20. Refer to Q.4 (c) of End Term Examination 2018 (Pg. No. 18-2018)
Q.21. Refer to Q.1 (a),(b) of First Term Examination 2019 (Pg. No. 1-2019)
Q.22. Refer to Q.2 (b) of First Term Examination 2019 (Pg. No. 2-2019)
Q.23. Refer to Q.4 (b) of First Term Examination 2019 (Pg. No. 5-2019)
Q.24. Refer to Q.1 (a),(b),(c) of End Term Examination 2019 (Pg. No. 7-2019)
Q.25. Refer to Q.3 (a) of End Term Examination 2019 (Pg. No. 11-2019)
Q.26. Refer to Q.7 (b) of First Term Examination 2019 (Pg. No. 15-2019)

UNIT - II

Q.1. Refer to Q.3 (a) of First Term Examination 2015 (Pg. No. 2-2015)
Q.2. Refer to Q.4 First Term Examination 2015 (Pg. No. 3-2015)
Q.3. Refer to Q.1 (g), (i) of End Term Examination 2015 (Pg. No. 9-2015)
Q.4. Refer to Q.2 (a) of End Term Examination 2015 (Pg. No. 9-2015)
Q.5. Refer to Q.7 (b) of End Term Examination 2015 (Pg. No. 13-2015)
Q.6. Refer to Q.8 (b), (c) of End Term Examination 2015 (Pg. No. 15-2015)
Q.7. Refer to Q.1 (e) of End Term Examination 2016 (Pg. No. 3-2016)
Q.8. Refer to Q.2 (a),(b) of First Term Examination 2016 (Pg. No. 3,4-2016)
Q.9. Refer to Q.7 (a),(b) of End Term Examination 2016 (Pg. No. 25,28-2016)
Q.10. Refer to Q.1 (d) of First Term Examination 2017 (Pg. No. 2-2017)
Q.11. Refer to Q.2 (a),(c) of First Term Examination 2017 (Pg. No. 2,3-2017)
Q.12. Refer to Q.3 (a),(b) of First Term Examination 2017 (Pg. No. 4-2017)
Q.13. Refer to Q.4 (a),(c),(d) of First Term Examination 2017 (Pg. No. 5,6-2017)
Q.14. Refer to Q.2 (a) of End Term Examination 2017 (Pg. No. 12-2017)
Q.15. Refer to Q.8 (a), (e) of End Term Examination 2017 (Pg. No. 23,24-2017)
Q.16. Refer to Q.9 (a),(b) of End Term Examination 2017 (Pg. No. 24-2017)
Q.17. Refer to Q.1 (c) of First Term Examination 2018 (Pg. No. 2-2018)
Q.18. Refer to Q.2 (d) of First Term Examination 2018 (Pg. No. 4-2018)
Q.19. Refer to Q.3 (a),(b) of First Term Examination 2018 (Pg. No. 5,6-2018)
Q.20. Refer to Q.4 (a),(b),(c) of First Term Examination 2018 (Pg. No. 6,7,8-2018)
Q.21. Refer to Q.1 (c),(d) of End Term Examination 2018 (Pg. No. 9,10-2018)
Q.22. Refer to Q.3 (a),(b),(c),(d) of End Term Examination 2018 (Pg. No. 15,16-2018)
Q.23. Refer to Q.4 (a),(b) of End Term Examination 2018 (Pg. No. 18-2018)
Q.24. Refer to Q.1 (d) of First Term Examination 2019 (Pg. No. 2-2019)
Q.25. Refer to Q.2 (a) of First Term Examination 2019 (Pg. No. 2-2019)
Q.26. Refer to Q.3 (a),(b),(c),(d),(e) of First Term Exam 2019 (Pg. No. 3,4-2019)
Q.27. Refer to Q.4 (a) of First Term Examination 2019 (Pg. No. 4-2019)
Q.28. Refer to Q.2 (a) of End Term Examination 2019 (Pg. No. 10-2019)
Q.29. Refer to Q.4 of End Term Examination 2019 (Pg. No. 13-2019)
Q.30. Refer to Q.7 (a) of End Term Examination 2019 (Pg. No. 15-2019)

syllabusx.live

UNIT - III

Q.1. Refer to Q.2 (a),(b) of Second Term Examination 2015 (Pg. No. 4-2015)
Q.2. Refer to Q.3 (a),(b) Second Term Examination 2015 (Pg. No. 5,6-2015)
Q.3. Refer to Q.4 (a), (b) of Second Term Examination 2015 (Pg. No. 6,7-2015)
Q.4. Refer to Q.1 (d),(h),(j) of End Term Examination 2015 (Pg. No. 8,9-2015)
Q.5. Refer to Q.2 (b) of End Term Examination 2015 (Pg. No. 10-2015)
Q.6. Refer to Q.3 (a),(b) of End Term Examination 2015 (Pg. No. 10,11-2015)
Q.7. Refer to Q.4 (a),(b) of End Term Examination 2015 (Pg. No. 12-2015)
Q.8. Refer to Q.6 (b) of End Term Examination 2015 (Pg. No. 13-2015)
Q.9. Refer to Q.8 (a),(d) of End Term Examination 2015 (Pg. No. 14,15 -2015)
Q.10. Refer to Q.4 (a) of First Term Examination 2016 (Pg. No. 6-2016)
Q.11. Refer to Q.1 (a),(b),(c),(d),(e) of Second Term Exam 2016 (Pg. No. 8,9,10-2016)
Q.12. Refer to Q.2 of Second Term Examination 2016 (Pg. No. 11-2016)
Q.13. Refer to Q.3 (a),(b) of Second Term Examination 2016 (Pg. No. 11,13-2016)
Q.14. Refer to Q.4 (a),(b) of Second Term Examination 2016 (Pg. No. 14-2016)
Q.15. Refer to Q.1 (a),(b),(c) of End Term Examination 2016 (Pg. No. 15,16-2016)
Q.16. Refer to Q.3 (a),(b) of End Term Examination 2016 (Pg. No. 19,20-2016)
Q.17. Refer to Q.4 (b) of End Term Examination 2016 (Pg. No. 20-2016)
Q.18. Refer to Q.8 (a),(b),(c) of End Term Examination 2016 (Pg. No. 28,30-2016)
Q.19. Refer to Q.1 of Important Questions 2016 (Pg. No. 31-2016)
Q.20. Refer to Q.2 of Important Questions 2016 (Pg. No. 31-2016)
Q.21. Refer to Q.4 (b) of First Term Examination 2017 (Pg. No. 5-2017)
Q.22. Refer to Q.5 of First Term Examination 2017 (Pg. No. 7-2017)
Q.23. Refer to Q.1 (e),(f),(g),(i) of End Term Examination 2017 (Pg. No. 9,10-2017)
Q.24. Refer to Q.2 (b) of End Term Examination 2017 (Pg. No. 12-2017)
Q.25. Refer to Q.4 (a),(b) of End Term Examination 2017 (Pg. No. 13,14-2017)
Q.26. Refer to Q.5 (a),(b) of End Term Examination 2017 (Pg. No. 14,15-2017)
Q.27. Refer to Q.6 (a),(b) of End Term Examination 2017 (Pg. No. 18,19-2017)
Q.28. Refer to Q.7 (a) of End Term Examination 2017 (Pg. No. 19-2017)
Q.29. Refer to Q.8 (c) of End Term Examination 2017 (Pg. No. 22-2017)
Q.30. Refer to Q.1 (e) of End Term Examination 2018 (Pg. No. 12-2018)
Q.31. Refer to Q.2 (a),(b),(c) of End Term Examination 2018 (Pg. No. 12,13,14-2018)
Q.32. Refer to Q.6 (a) of End Term Examination 2018 (Pg. No. 21-2018)
Q.33. Refer to Q.7 (b),(c) of End Term Examination 2018 (Pg. No. 23-2018)
Q.34. Refer to Q.1 (d),(e),(f) of End Term Examination 2019 (Pg. No. 8,9-2019)
Q.35. Refer to Q.2 (c) of End Term Examination 2019 (Pg. No. 11-2019)
Q.36. Refer to Q.5 (a),(c) of End Term Examination 2019 (Pg. No. 13,14-2019)
Q.37. Refer to Q.6 (a),(b),(c) of End Term Examination 2019 (Pg. No. 14,15-2019)

UNIT - IV

Q.1. Refer to Q.6 (a) of End Term Examination 2016 (Pg. No. 24-2016)
Q.2. Refer to Q.1 (h) of End Term Examination 2017 (Pg. No. 10-2017)
Q.3. Refer to Q.8 (b) of End Term Examination 2017 (Pg. No. 21-2017)
Q.4. Refer to Q.5 (a),(b) of End Term Examination 2018 (Pg. No. 19,20-2018)
Q.5. Refer to Q.7 (c) of End Term Examination 2019 (Pg. No. 16-2019)

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 associated entities that each contains specialized attributes that apply to some but not all of the instances of the entity.

Q.2. (*a*) Descirbe 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 dataindependence? Explain logical and physical data independence.

Ans. Refer Q.No 2(c) of First Term Exam 2017

syllabusx.live

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*) $\sigma_p = a$ (*T*1)
(*ii*) $\pi_{P,R}$ (*T*1)
(*iii*) *T*1 ∪ T_2
(*iv*) T_1 ∩ 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*) **T1 – T2**

P	*Q*	*R*
a	1	–
c	2	10
t	2	–
a	–	–

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

Ans. Refer Q.No 1(c) of First Term Exam 2016

Q.4. Emplyee (employee-name, street, city) **(5x2)**
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';

syllabusx.live

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 executed 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)

syllabusx.live

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

T2 ← T3 (waits)

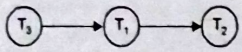
Wait-die Scheme	Wound-Wait Scheme
(i) This scheme is Non-Preemptive (ii) Lower priority transaction can never wait for higher transactions.	(i) This scheme is Preemptive. (ii) The higher priority transaction never wait for lower priority transaction.

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

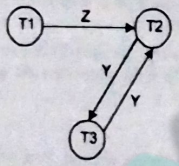


Schedule is not a serializable schedule because

- 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
A → ABCD
A → AB
A → CD
A → ABC
A → B
A → C
A → A
AB → D
AB → ABC
AB → AB
AB → A
AB → B
AB → C
DC → 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
BC → DEF
BC → EF
BC → D
B → CDEF
B → CD
B → EF
B → C
B → B
E → F
E → BCDEF
E → CD
E → EF

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.

syllabusx.live

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 Q.No.3(b) of First Term Exam 2016

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.

syllabusx.live

(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 β ≤ α then α → β holds.

(ii) Augmentation Rule: If α → β holds, and *w* is a set of attributes, then *v* α → *v*β holds.

(iii) Transitivity Rule: If α→ β holds and β→ *v* holds then α → *v* holds.

(iv) Complementation Rule: If α→β holds, then α → → R – β – α holds

(v) Multivalued transtivity rule: If α→ → β holds and β →→ *v* holds then α → → *v* holds then α → → *v*–β holds.

(vi) Replication Rule: If α → β holds then α → → β.

(vii) Coalescence Rule: If α→→β holds and *v* ≤ β and there is a *z* such that δ ≤ *R* and δ ∩ β = ϕ and δ → *v* then α → v holds.

(viii) Intersection Rule: If α → → β holds and α → → *v* holds then α → → β ∩ *v* holds.

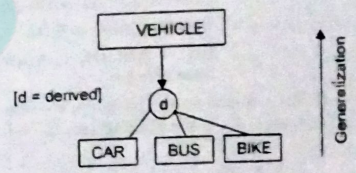
(ix) Difference Rule: If α→→β holds and α → → *v* holds then α → → β-*v* holds and α → → *v*-β holds.

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 characterics.

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.

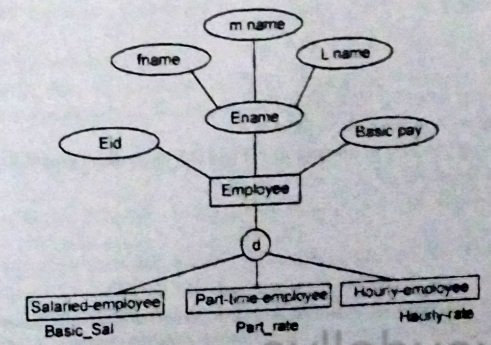
Example:



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:



syllabusx.live

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$^\alpha$, 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

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

Where α 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?

Ans.6.a. Refer Q.No.5 of First Term 2017

Q.6. (b) Formulate a concurrency control protocol based on read and write locks. **(7.5)**

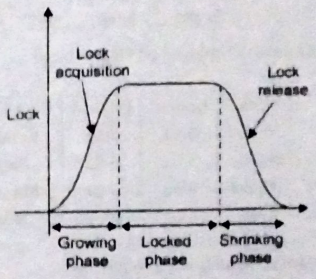
Ans. Mainly there are two protocols used i.e.

(*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.?

Ans.7.a. Refer Q.No.3(a) of second term 2016 and Q.No. 6(b) end term 2017

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.

syllabusx.live

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 derivedfrom 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.

syllabusx.live

Triggers are required:

(a) To enforce rules that cannot be coded through referential integrety.

(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.5. (a) Indexing

Ans. Indexes in SQL: An index is defined as a table or data structure that 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 pointer to the location where the record is stored. Each index entry corresponds to 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 worksis in the same way as that of an index in a file in a database system. Another example could be a library catalogue.

Indexing is of two types:

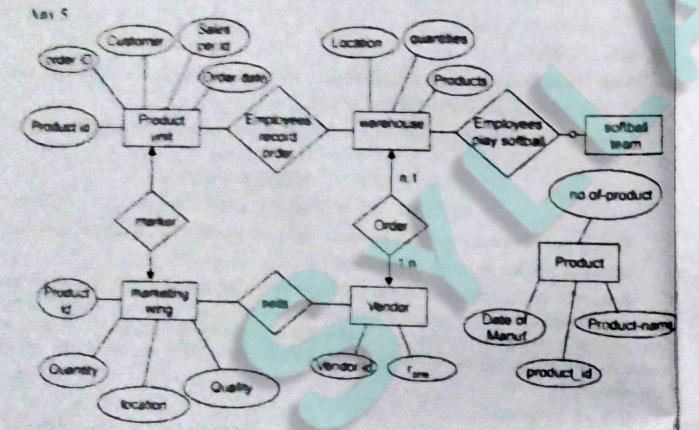
(a) Ordered indexing

(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.

Ans. 5



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. π, x , σ, 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.

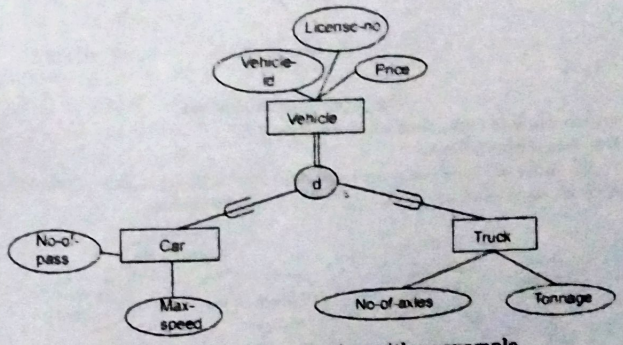
• Can have specialization hierarchies or lattices, or generalization, hierarchies or lattices.

• In specialization, start with an entity type and then define subclass of the entity type by successive specialization (top down conceptual 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.

syllabusx.live

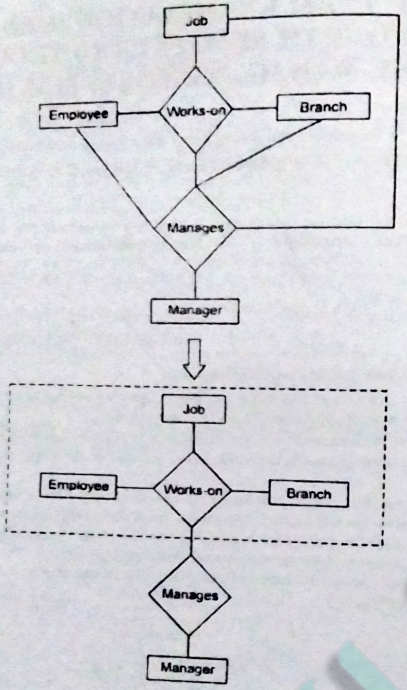


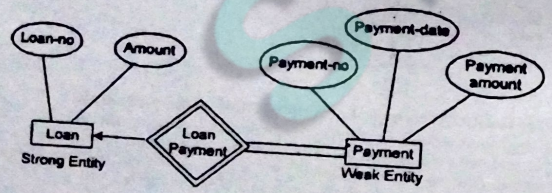
Fig. Aggregation concept.

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

SYLLABUS
syllabusdrive

(a) Give the relation obtained by Natural Join, Left outer join and right outer join, equi-join and theta join. **(8)**

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 equ join.

Loan_No	Branch_Name	Amount	Cust_Name	Loan_No
L–170	ICICI	30,000	John	L–170

Theta join

$(LOAN \bowtie_{Loan.loanNo=Borrower.Loan_No} Borrower)$

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 **(4)**

(i) Find the customer name having loan in ICICI branch.

(ii) Find the Branch Name and Loan No. with the Loan amount greater than 10,000.

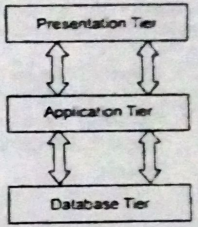
Ans. (i) $\pi_{Cust_Name}(\sigma_{Branch_Name=ICICI}(Loan \bowtie Borrower))$

(ii) $\pi_{Branch_Name,\ Loan_No}(\sigma_{Amount>10,000}(Loan))$

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)**

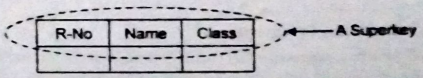
Ans. Key : are very important part of Relational database. They are used to establish and identify relation between tables. They also ensure that each record within a table can be uniquely identified by combination of one or more fields within a table.

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:

syllabusx.live

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 **(3)**

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.

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 **(3)**

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.

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).

syllabusx.live

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 five more questions from the remaining.*

Q.1. (a) Define Normalisation. (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 Normalised. 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 transitively 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.

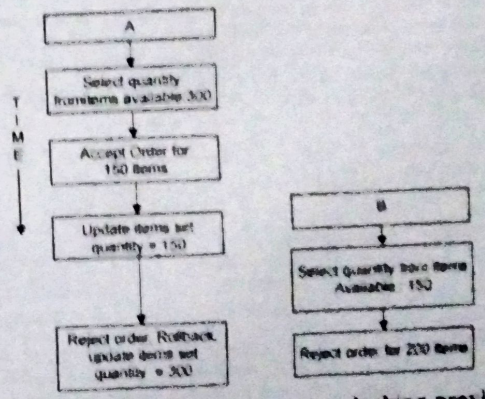
Example:

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 inventorics (150 available) and reports 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.

syllabusx.live

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,

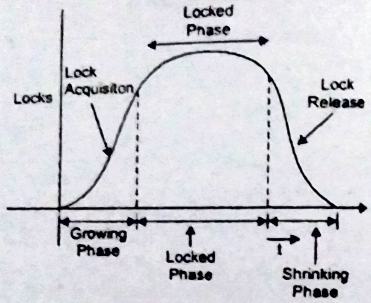


Fig. PL Scheme.

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.

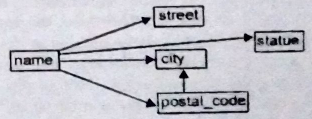
(a) Given a set of FDs for this relation. **(2.5)**

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

syllabusx.live

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) A =A+30
read(A) <validate> display(A+B)	
	<validate> write (B) write(A)

Figure. Schedule 5,a schedule produced by using validation

Consider again transactions $T14$ and $T15$. Suppose that TS($T14$) < TS($T15$). 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 $T15$. Thus, $T14$ 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.

syllabusx.live

Q.4. (a) Consider two sets of functional dependencies **(5)**

F1 = {A->C, AC-> D,E->AD, E->H}

F2 = {A->CD, E->AH}

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)

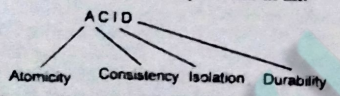
$\{A,H\} \subseteq \{E\}^+$, $E \rightarrow AH$ is covered by F.

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

syllabusx.live

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.

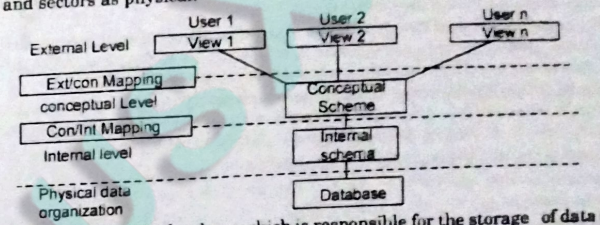
Q.1. (e) Internal schema.

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 cach 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.

syllabusx.live

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.

syllabusx.live

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 $\{(F - \{X \rightarrow A\}) \cup \{(X - \{B\}) \rightarrow A\}\}$ is equivalent to F,

then replace $X \rightarrow A$ with $(X - \{B\}) \rightarrow A$ 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 Algurithm 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 $B \rightarrow AB$ (c). However, $AB \rightarrow D$ as given (b).

• 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. 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.

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

Q.4. (a) Explain the "ACID" properties in brief.

Ans. Refer Q.4. (b) Second Term Examination 2016. **(6)**

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:

$w_1(A), r_2(A), w_1(B), w_2(C), r_2(C), r_4(B), w_2(D), w_4(E), r_5(D), w_5(E)$

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 $T_1 \rightarrow T_4$

$w_2(C)$: C is subsequently read by T_2, so add edge $T_2 \rightarrow T_2$

$r_2(C)$: 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 $T_2 \rightarrow T_2$

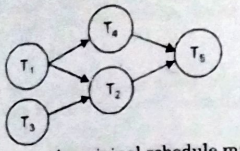
$w_4(E)$: E is subsequently written by T_5, so add edge $T_4 \rightarrow T_5$

$r_5(D)$: 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.

syllabusx.live

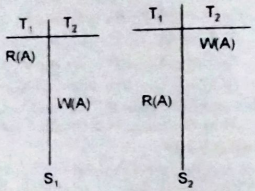


4. This graph has no cycles, so the original schedule must be serializable.

View Seriaizability: View equivalent schedule and View Serializable Schedule

View Equivalent Schedule : Consider two schedules S_1 and S_2, they are said to view equivalent if following conditions are true :

- **Initial read must be same.**

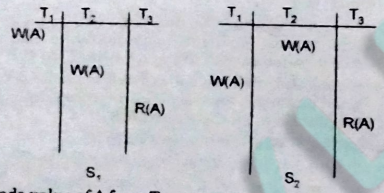


- S_1 : T_1 reads A from Database.

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



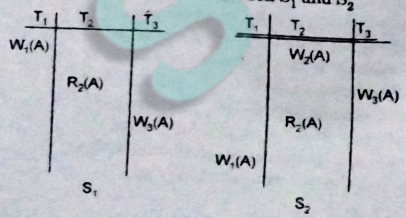
- S_1 : T_3 reads value of A from T_2.

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.

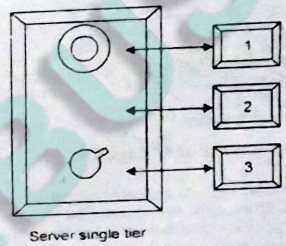


Fig. (A)

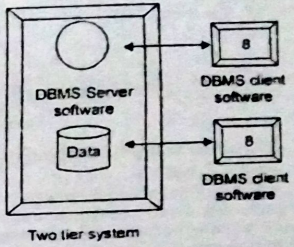


Fig. (B)

(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.

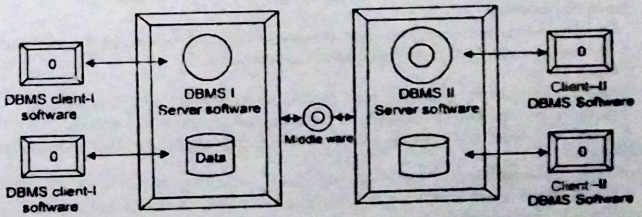
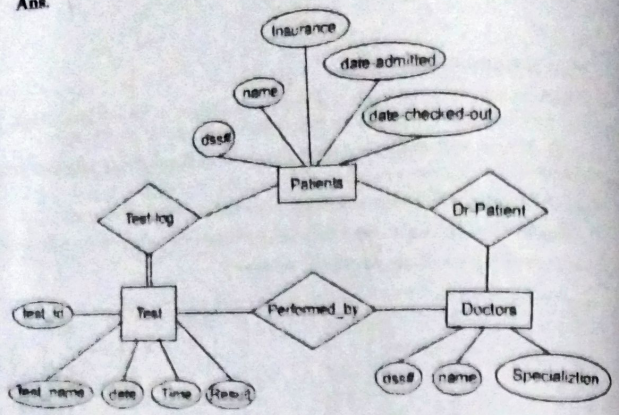


Fig. (c) Three Tier System

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)**

syllabusx.live

Ans.



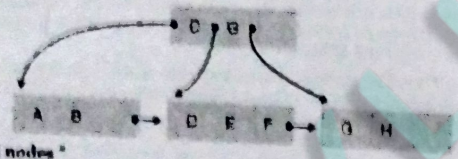
E-R diagram for a hospital.

Q.6. (a) How the B+ tree index files are maintained? Explain. **(8)**

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.

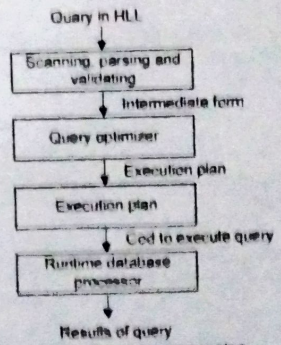
B⁺ Tree Deletion

- 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. **(8)**

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

$R1 \bowtie_\theta R2$

syllabusx.live

R1 and R2 are relations having attributes (A1, A2, .., An) and (B1, B2,.. Bn) 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.

syllabusx.live

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) $r \times s$ (iv) $\pi_{A,F}(\sigma_{C=D}(r \times s))$

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)

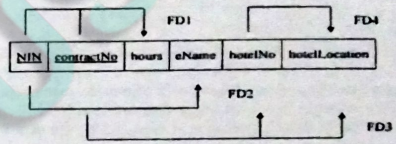
Ans. An agency called InstantCover supplies part-time/temporary staff to hotels throughout Scotland. The table shown in Figure lists the time spent by agency staff working at two hotels. The National Insurance Number (NIN) is unique for employee.

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

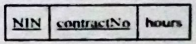
Describe and illustrate the process of normalizing the table. State any assumptions you make about the data shown in this table.

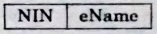
The table shown is susceptible to update anomalies. Provide examples of insertion, deletion, and modification anomalies.

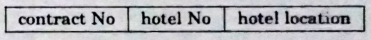
Answer: In the 1NF we remove all the repeating groups if any and assigned primary keys (candidate keys). Then we figure out the functional dependencies (FDs). By using dependency diagram we represent the table as shown below. (NF –stand for Normal Form)



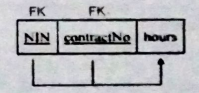
Obviously FD2 and FD3 violate teh 2NF (partial dependency) and FD4 violate 3NF (transitive dependency). In the 2NF we remove the partial dependency of the FD2 and FD3 by splitting it into new tables as shown below.

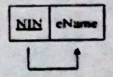


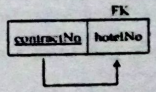


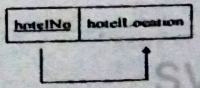


In 3NF we remove the transitive dependency of teh FD4 by splitting it into new table as shown below.









syllabusx.live

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

T' hold the lock		Read	Write
	Read	yes	no
	write	no	no

(b) read/write/certify locking scheme

T' hold the lock		Read	Write	certify
	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) $r_1(X); r_3(X); w_1(X); r_2(X); w_2(X);$ (ii) $r_1(X); r_2(X); w_2(X); w_1(X); r_3(X);$

(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_2 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_3 and T_1, $r_1(X)$ will happen after $w_3(X)$, 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, $\{A, C, D\}^+$ = $\{A, B, C, D, E, F\}$

abusx.live

(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 characteristic 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.

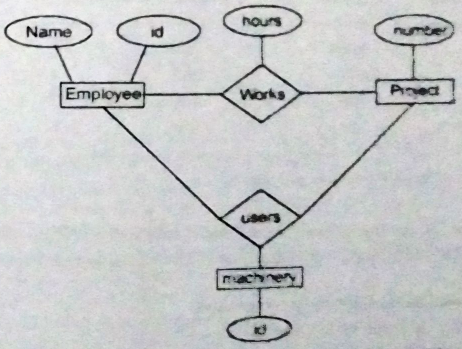


Fig. E.R. diagram with redundant relationship

syllabusx.live

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.

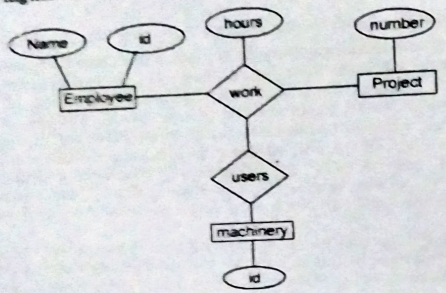


Fig. E.R. diagram with aggregation

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.

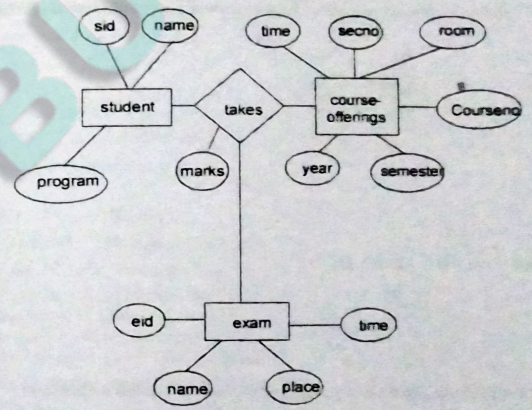
Q.2.(a) What is the difference between the terms 'relation' and 'relation schema'? Explain with an example. (3)

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.

SYLLABUSX
syllabusx.live

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*) $\pi_{S.\ roll\ no.,\ S.name}$ (student) $p_{s.branch}$ = "CSE"

(*ii*) $\pi_{S.name}$ ($\sigma_{B.Publisher = "ABC"}$ (Students $\bowtie$ Books B)).

(*iii*) $\pi_{s.name,}$ ($\sigma_{date\text{-}of\text{-}issue = "jan\text{-}1\text{-}2016"}$ (Books B $\bowtie$ Issue I)

$\pi_{I.Date_of_Issue}$ = '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

syllabusx.live

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 equij The above example corresponds to equijoin.

Q.4.(d) Explain the importance of defining view.

Ans. A view is nothing more than a SQL statement that is stored in the datab with an associated name. A view is actually a composition of a table in the form predefined SQL query.

A view can contain all rows of a table or select rows from a table. A view can created from one or many tables which depends on the written SQL query to crea 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 intuit
- Restrict access to the data such that a user can see and (sometimes) mod exactly what they need and no more.
- Summarize data from various tables which can be used to generate report

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.

syllabusx.live

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 eardinalities, 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:

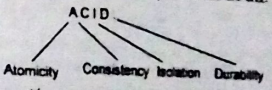
(i) They should have a minimal number of attributes necessary to suport the data requirements of the enterprise.

(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.

syllabusx.live

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.

syllabusx.live

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 $A \to D$?

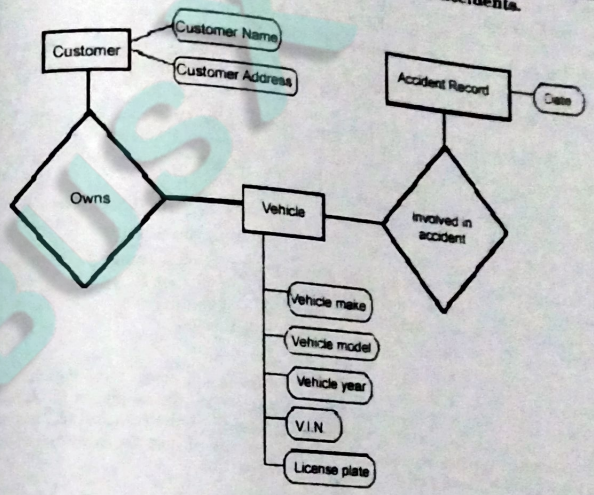
- Since $B \to A$, by augmenting with B on both sides we have $BB \to AB$, or $B \to AB$ i). However, $AB \to D$ as given (b).
- Hence by the transitive Rule, we get from (a) and (b), $B \to D$. Hence $AB \to D$ *may* be replaced by $B \to D$.
- We now have a self equivalent to original E, say E': $\{B \to A, D \to A, B \to D\}$. 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$\}$

syllabusx.live

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.

T1	T2
• Read (x)	
• Read (x)	
	• Read (x)
• Write (y)	

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:

T_1	T_2	T_3
• Read (x)		
• Read (y)		
• Write (x)		
	• Read (x)	
	• Write (x)	
		• Read (x)

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:

T1	T2
• Read(x)	
• Write(x)	
	• Read(y)
• Commit	
	• Read(x)

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

syllabusx.live

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.350, which is incorrect. The reason for this mistake is that the transaction T2 displays 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)

SYLLABUSX
syllabusx.live

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.

Timestamp pes 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.,

$$T_1 \rightarrow T_2 \text{ (Rollbacks)}$$

Now, if T_3 requests a data-item held by T_2, then T_3 waits

$$T_2 \leftarrow T_3 \text{ (waits)}$$

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_i, 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.

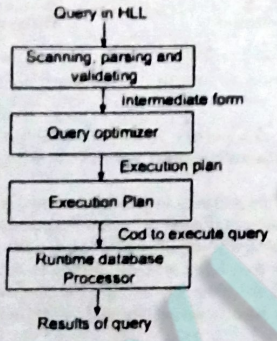
syllabusx.live

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 undesirable 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 the 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.

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: **(2.5x5 = 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 situations. 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 structures 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.

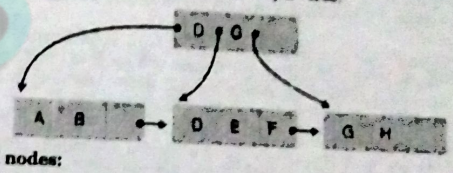
You should also pay particular attention to tuning the Uniface driver for the DBMS. Issues that play a role include a number of parameters for managing real memory required for communicating with the DBMS.

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 artual 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** 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.

syllabusx.live

- 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
Bolt	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.

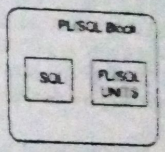
- **Architecture of PL/SQL**

The PL/SQL architecture mainly consists of following 3 components

1. PL/SQL block
2. PL/SQL Engine
3. Database Server

PL/SQL block:

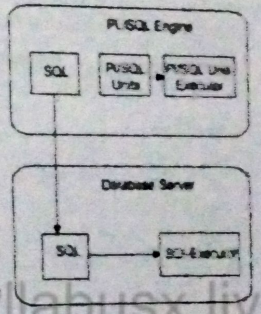
- This is the component which has the actual PL/SQL code.
- This consists of different sections to divide the code logically (declarative section for declaring purpose, execution section for processing statements, exception handling section for handling errors)
- It also contains the SQL instruction 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 Library
- o Package Body
- o Trigger
- o Type Body
- o Function
- o Procedure
- o Package Specification
- o Type

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.



syllabusx.live

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 example. (4)

Ans. Division: 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: 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

FIRST TERM EXAMINATION [FEB. 2018]
FOURTH SEMESTER [B.TECH]
DATABASE MANAGEMENT SYSTEM
[ETCS–208]

Time : 1½ hrs. **M.M. : 30**

Note: Attempt any three questions. Q.N. 1 is compulsory.

Q.1. Attempt Q. 1. and any two more questions.

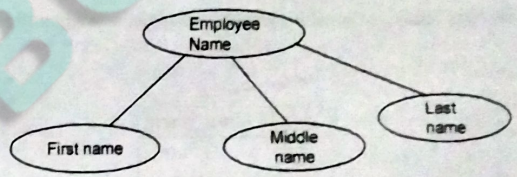
(a) Explain the following terms:

(i) Composite attributes

(ii) Data model

Ans. (i) Composite Attributes: An attribute is considered composite if it comprises two or more other attributes.

Example: Entity Employee Name can be divided into sub divisions like FName, MName, LName.



(ii) Data Model: Data models define how the logical structure of a database is modeled. Data Models are fundamental entities to introduce abstraction in a DBMS. They define how data is connected to each other and how they are processed and stored inside the system. Various data models in DBMS are:

a. Flat data models where all the data used are to be kept in the same plane.

b. Entity-Relationship (ER) Model is based on the notation of real-world entities and relationships among them.

c. Relational Model where data is stored in tables called **relations.**

Q.1. (b) Differentiate between physical and logical data independence.(2)

Ans. Logical data independence is difficult to achieve that the phyasical data independece. 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 shcema without causing application programs to be written.	It is the ability to modify the logical schema without causing application programs to be rewritten.

syllabusx.live

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.1. (c) Explain natural join and outer join in SQL.

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-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.1. (d) With the help of an example explain generalized projection using relational algebra. **(2)**

Ans. Refer Q.1.(d). FirstTerm Examination Feb -2017.[page no. 2-2017].

Q. 1. (e) Explain the function of following subsystems of database: **(2)**

(i) Query processor

(ii) Storage manager

(I). 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

(II). Storage Manager: A program module that provides the interface between the low level data stored in database and the application programs and queries submitted to the system. The storage manager is responsible for interaction with the file manager. It translates various DML statements into low-level file-system commands, thus the storage manager is responsible for storing, retrieving and updating data in database.

Q. 2. (a) What is the difference between a database schema and a database state? **(2)**

Ans. Difference between database schema and database state

Database Schema-

(i) It is the **overall Design of the Database**. It is the **skeleton structure** that **represents** the **logical view** of the entire database.

(ii) It **tells** how the data is organized and how the relations among them are associated.

(iii) This can be considered as a blueprint of a database, and gives a list of fields in the database with their data types

Database State-

(i) A **database state** provides the present state of the database and its data. It can be considered as an extension of the database schema.

(ii) When a database is newly defined, the corresponding database state is empty.

(iii) Factors that affect a database state are entering, deleting or modifying information in the database.

Q. 2. (b) Explain the three level architecture of a database system with the help of a diagram. **(3)**

Ans. Refer Q.5.(a). END Term Examination june -2016.[page no. – 2016-23].

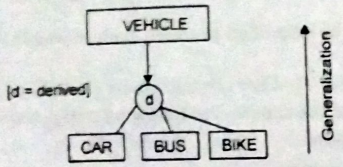
Q. 2. (c) Explain generalization and specialization in E-R diagrams. Support your answer with an example. **(3)**

syllabusx.live

Ans. Generalization: It is defined as the process of identifying some common characteristics of a collection of entiry sets and creating a new entity set that contains entities processing these common features.

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.

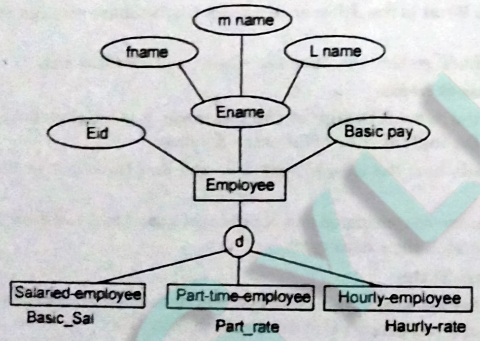
Example:



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 defining 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.

Example:



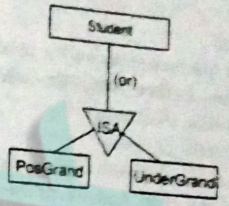
Q. 2. (d) Differentiate between disjoint and overlapping constaint. **(2)**

Ans. Difference between disjoint and overlapping constraints

Disjoint-

1. In a disjoint design constraint, an entity can be atmost one of the subclasses of the specialization. i.e. Employee is disjoint to secretery,technician,engineer etc which means that an employee can be anyone of the above mentioned designation

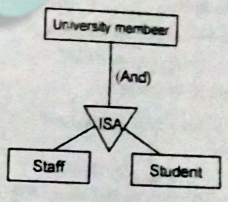
2. To represent a disjoint superclass/subclass relationship, Or is used.



Overlapping-

1. Overlapping constraints are revert in the sense that the superclass can belong to more than one subclass. for example person as a superclass and passport,driving license as its overlapping subclass

2. And is used to represent the overlapping specialization/generalization relationship in the ER diagram.



Q. 3. (a) What is a view? Why do we define a view? Write the syntax of creating a view using SQL. **(3)**

Ans. A VIEW is a virtual table, through which a selective portion of the data from one or more tables can be seen. Views do not contain data of their own. They are used to restrict access to the database or to hide data complexity. A view is stored as a SELECT statement in the database. DML operations on a view like INSERT, UPDATE, DELETE affects the data in the original table upon which the view is based.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

Syntax for creating sql view:

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

view_name is the name of the VIEW.

The SELECT statement is used to define the columns and rows that you want to display in the view

A view always shows up-to-date data the database engine recreates the data, using the view's SQL statement, every time a user queries a view.

For Example- to create a view on the product table the sql query would be like:

```
CREATE VIEW view_product
AS
SELECT product_id, product_name
```

syllabusx.live

FROM product
WHERE product_id> 10

Q. 3. (b) Consider the following schema for institute library:

Student (RollNo, Name, Father_Name, Branch)

Book (ISBN, Title, Author, Publisher)

Issue (RollNo, ISBN, Date_of_Issue)

Write the following queries in relational algebra:

(i) List RollNo and Name of all students of the branch 'CSE'. (1)

(ii) Find the name of students who have issued a book published by 'ABC' publisher. (2)

(iii) List title of all books and thier authors issued by a student 'XYZ'. (2)

(iv) List title of all books issued on or before Jan 1, 2014. (2)

Ans. (i) $\sigma_{S.roll\ no., S.Name}$ (Student)$\pi_{S.branch}$ = "CSE"

(ii) $\sigma_{S.Name}$ (Student $\bowtie$ Book B)$\pi_{B.publisher}$ = "ABC"

(iii) $\sigma_{B.Title, B.author}$ (Book B $\bowtie$ Issue I $\bowtie$ Student S)

$\pi_{s.name}$ = "XYZ"

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

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

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

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

Q. 4. (a) List any three differences between SQL and PL/SQL. (3)

Ans. Difference between SQL and PL/SQL

BASIS FOR COMPARISON	SQL	PL/SQL
Basic	In SQL you can execute a single query or a command at a time.	In PL/SQL you can execute a block of code at a time.
Full form	Structured Query Language	Procedural Language, extension of SQL.
Purpose	It is like a source of data that is to be displayed.	It is language that creates an application that display's the data acquired by SQL.
Writes	In SQL you can write queries and command using DDL, DML statements.	In PL/SQL you can write block of code that has procedures, functions, packages or variables, etc.
Use	Using SQL, you can retrieve, modify, add, delete, or manipulate the data in the database.	Using PL/SQL, you can create applications or server pages that display's the information obtained from SQL in a proper format.
Embed	You can embed SQL statement in PL/SQL.	You can not embed PL/SQL in SQL

Q. 4.(b) Explain PL/SQL block structure with an example. (4)

Ans. PL/SQL block structure

PL/SQL program units organize the code into blocks. A block without a name is known as an anonymous block. The anonymous block is the simplest unit in PL/SQL. It is called anonymous block because it is not saved in the Oracle database.

An anonymous block is an only one-time use and useful in certain situations such as creating test units. The following illustrates anonymous block syntax:

```
[DECLARE]
  Declaration statements;
BEGIN
  Execution statements;
[EXCEPTION]
    Exception handling statements;
END;
/
```

PL/SQL block structure follows divide-and-conquer approach to solve the problem stepwise.

The anonymous block has three basic sections that are the declaration, execution, and exception handling. Only the execution section is mandatory and the others are optional.

- The declaration section allows you to define data types, structures, and variables. You often declare variables in the declaration section by giving them names, data types, and initial values.
- The execution section is required in a block structure and it must have at least one statement. The execution section is the place where you put the execution code or business logic code. You can use both procedural and SQL statements inside the execution section.
- The exception handling section is starting with the EXCEPTION keyword. The exception section is the place that you put the code to handle exceptions. You can either catch or handle exceptions in the exception section.

```
• Example -
• DECLARE— DECLARE block, declare and initialize values
•    designation VARCHAR2(30);
• eno number(5) :=5;
•    id BOOLEAN;
•    inter INTERVAL YEAR(2)TO MONTH;
• BEGIN— BEGIN block, also assign values
•    designation := UPPER('Web Developer');
•    id :=TRUE;
•    inter := INTERVAL '45' YEAR;
• END;
• /
```

syllabusx.live

Q.4. (c) What do you mean by union of two tables? Explain when two tables are said to be union compatible. **(3)**

Ans. Union of two tables:

The UNION of the two tables gives you a virtual result table containing all the rows in the first table plus all the rows in the second table.

The UNION DISTINCT operator functions identically to the UNION operator without the DISTINCT keyword. In both cases, duplicate rows are eliminated from the result set.

On the other hand Union All operation preserves the duplicate rows.

Two relations R and S, are said to be UNION COMPATIBLE, only if they satisfy these two conditions:

1. Both the relations, i.e. R and S should be of same arity, i.e. they should have same number of attributes.

2. Domain of the similar attributes, i.e. domain of ith attribute of R and domain of ith attribute of S, should be same and this is applicable to all the attributes.

For example:

R (Sid, Sname) and S (Sid , Sname , Marks) are not UNION COMPATIBLE, as R has 2 attributes while S has 3 attributes.

R (Sid, Sname) and S (Sid , Marks) are not UNION COMPATIBLE, as domain of name and marks is different. Domain of names would be certain strings while domain of marks would be integer and float numbers.

R (Sid, Sname) and S (Sid , StudentName) are UNION COMPATIBLE, as R and S both have 2 attributes and their domains are also same, only the column names are different.

END TERM EXAMINATION [MAY-JUNE 2018]
FOURTH SEMESTER [B.TECH]
DATABASE MANAGEMENT SYSTEM
[ETCS–208]

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

Note: *Attempt any five questions including Q.1. which is compulsory.*

Q.1. (a) What is the use of data dictionary in a database system? How it is stored in a database? **(5)**

Ans. 1. Data Dictionary

A data dictionary is a file or a set of files that contains a database's metadata. The data dictionary contains records about other objects in the database, such as data ownership, data relationships to other objects, and other data.

The data dictionary is a crucial component of any relational database. Ironically, because of its importance, it is invisible to most database users. Typically, only database administrators interact with the data dictionary.

In a relational database, the metadata in the data dictionary includes the following:

- Names of all tables in the database and their owners
- Names of all indexes and the columns to which the tables in those indexes relate
- Constraints defined on tables, including primary keys, foreign-key relationships to other tables, and not-null constraints

For most relational database management systems (RDBMS), the database management system software needs the data dictionary to access the data within a database. For example, the Oracle DB software has to read and write to an Oracle DB. However, it can only do this via the data dictionary created for that particular database.

Here are the two types of data dictionary:

Active Data Dictionary

The DBMS software manages the active data dictionary automatically. The modification is an automatic task and most RDBMS has active data dictionary. It is also known as integrated data dictionary.

Passive Data Dictionary

Managed by the users and is modified manually when the database structure change. Also known as non-integrated data dictionary.

Q.1.(b) How are specialization and generalization used to define the conceptual schemes during conceptual database design? **(5)**

Ans. Refer Q.2.(c) First Term Examination 2018.

Q.1. (c) What is the difference between equi join and natural join operations? Explain with an example. **(5)**

Ans. Difference between equi join and natural join

Equi Join is a join using one common column (referred to in the "on" clause). This join is a equally comparison join, thus not allowing other comparison operator such as <, >=, etc. The joining id needs to be exact match and doesn't allow for any variations or filters like "having" clause. Here the join is explicitly mentioned.

SYLLABUSX
syllabusx.live

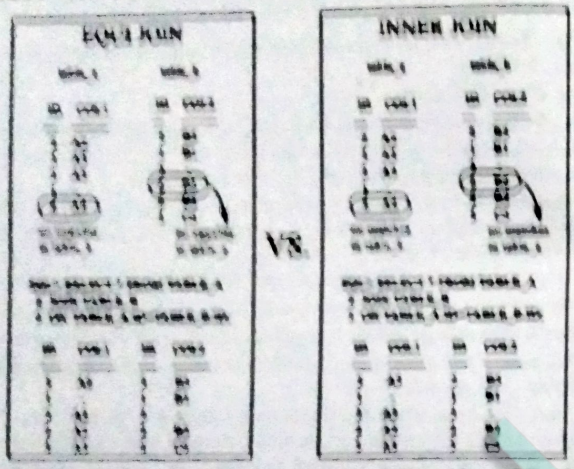
Example

SELECT * FROM table1 JOIN table2 ON (table1.id = table2.id)

Natural Join is an implicit join clause based on the common columns in the two tables being joined. Common columns are columns that have the same name in both tables. This is a much rarer join as the join here is implied. The query attempts to use all columns with common names to create a join. An asterisk qualified by a table name (for example, Table1.*) will be expanded to every column of that table that is not a common column.

Example

SELECT * FROM table1 NATURAL JOIN table2



Q.1 (d) What are the rules of decomposing a relation scheme? Explain. (8)

Ans. A functional **decomposition** is the process of breaking down the functions of an organization into progressively greater (finer and finer) levels of detail.

In decomposition, one function is described in greater detail by a set of other supporting functions.

Decomposition helps in eliminating some of the problems of bad design such as redundancy, inconsistencies and anomalies.

There are two types of decomposition :

1. Lossy Decomposition: "The decomposition of relation R into R1 and R2 is **lossy** when the join of R1 and R2 does not yield the same relation as in R."

One of the disadvantages of decomposition into two or more relational schemes (or tables) is that some information is lost during retrieval of original relation or table.

Consider that we have table STUDENT with three attribute roll_no ,sname and department.

STUDENT:

Roll_no	Sname	Dept
111	parimal	COMPUTER
222	parimal	ELECTRICAL

This relation is decomposed into two relation no_name and name_dept :

No_name:

Roll_no	Sname
111	parimal
222	parimal

Name_dept:

Sname	Dept
parimal	COMPUTER
parimal	ELECTRICAL

In lossy decomposition, spurious tuples are generated when a natural join is applied to the relations in the decomposition.

stu_joined :

Roll_no	Sname	Dept
111	parimal	COMPUTER
111	parimal	ELECTRICAL
000	parimal	COMPUTER
000	parimal	ELECTRICAL

The above decomposition is a bad decomposition or Lossy decomposition.

2. Lossless Join Decomposition: "The decomposition of relation R into R1 and R2 is **lossless** when the join of R1 and R2 yield the same relation as in R."

A relational table is decomposed (or factored) into two or more smaller tables, in such a way that the designer can capture the precise content of the original table by joining the decomposed parts. This is called lossless join (or non additive join) decomposition.

This is also reffered as non additive decomposition.

The lossless join decomposition is always defined with respect to a specific set F of dependencies.

Consider that we have table STUDENT with three attribute roll_no ,sname and department.

STUDENT :

Roll_no	Sname	Dept
111	parimal	COMPUTER
222	parimal	ELECTRICAL

This relation is decomposed into two relation Stu_name and Stu_dept :

Stu_name:

Roll_no	Sname
111	parimal
222	parimal

Stu_dept :

Roll_no	Dept
111	COMPUTER
222	ELECTRICAL

Now ,when these two relations are joined on the comman column 'roll_no' ,the resultant relation will look like stu_joined.

stu_joined :

Roll_no	Sname	Dept
111	parimal	COMPUTER
222	parimal	ELECTRICAL

syllabusx.live

In lossless decomposition, not any spurious tuples are generated when a natural joined is applied to the relations in the decomposition.

Q. 1.(e) When a transaction is said to follow two phase locking protocol? Explain the growing and shrinking phase of two phase locking protocol. **(5)**

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.

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,

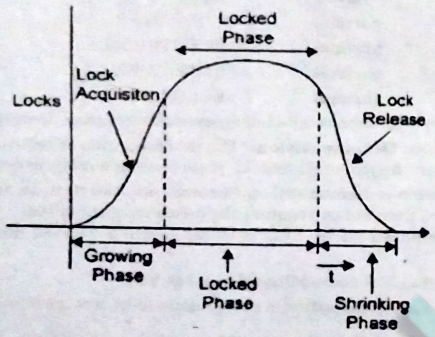


Fig. PL Scheme.

Note: As per 2PL protocol, the transaction cannot acquire a new lock after it has unlocked any of its existing locked items.

Q. 2.(a) When are functional dependencies said to be inferred from or implied by? **(4)**

Ans. Functional Dependency (FD): A functional dependency is an association 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 $\beta \leq \alpha$ then $\alpha \rightarrow \beta$ holds.

(ii) Augmentation Rule: If $\alpha \rightarrow \beta$ holds, and w is a set of attributes, then $v\alpha \rightarrow v\beta$ holds.

(iii) Transitivity Rule: If $\alpha \rightarrow \beta$ holds and $\beta \rightarrow v$ holds then $\alpha \rightarrow v$ holds.

(iv) Complementation Rule: If $\alpha \rightarrow \beta$ holds, then $\alpha \rightarrow \rightarrow R - \beta - \alpha$ holds

(v) Multivalued transtivity rule: If $\alpha \rightarrow \rightarrow \beta$ holds and $\beta \rightarrow \rightarrow v$ holds then $\alpha \rightarrow \rightarrow v - \beta$ holds then $\alpha \rightarrow \rightarrow v - \beta$ holds.

(vi) Replication Rule: If $\alpha \rightarrow \beta$ holds then $\alpha \rightarrow \rightarrow \beta$.

(vii) Coalescence Rule: If $\alpha \rightarrow \rightarrow \beta$ holds and $v \leq \beta$ and there is a s such that $\delta \leq R$ and $\delta \cap \beta = \phi$ and $\delta \rightarrow v$ then $\alpha \rightarrow v$ holds.

(viii) Intersection Rule: If $\alpha \rightarrow \rightarrow \beta$ holds and $\alpha \rightarrow \rightarrow v$ holds then $\alpha \rightarrow \rightarrow \beta \cap v$ holds.

(ix) Difference Rule: If $\alpha \rightarrow \rightarrow \beta$ holds and $\alpha \rightarrow \rightarrow v$ holds then $\alpha \rightarrow \rightarrow \beta - v$ holds and $\alpha \rightarrow \rightarrow v - \beta$ holds.

Q. 2.(b) Why is the domain key normal form known as the ultimate normal form? Explain with an example. **(4)**

Ans. Domain-key normal form (DK/NF) is a normal form used in database normalization which requires that the database contains no constraints other than domain constraints and key constraints.

A domain constraint specifies the permissible values for a given attribute, while a key constraint specifies the attributes that uniquely identify a row in a given table.

The domain/key normal form is achieved when every constraint on the relation is a logical consequence of the definition of keys and domains, and enforcing key and domain restraints and conditions causes all constraints to be met. Thus, it avoids all non-temporal anomalies.

The reason to use domain/key normal form is to avoid having general constraints in the database that are not clear domain or key constraints. Most databases can easily test domain and key constraints on attributes. General constraints however would normally require special database programming in the form of stored procedures (often of the trigger variety) that are expensive to maintain and expensive for the database to execute. Therefore, general constraints are split into domain and key constraints.

It's much easier to build a database in domain/key normal form than it is to convert lesser databases which may contain numerous anomalies. However, successfully building a domain/key normal form database remains a difficult task, even for experienced database programmers. Thus, while the domain/key normal form eliminates the problems found in most databases, it tends to be the most costly normal form to achieve. However, failing to achieve the domain/key normal form may carry long-term, hidden costs due to anomalies which appear in databases adhering only to lower normal forms over time.

The third normal form, Boyce–Codd normal form, fourth normal form and fifth normal form are special cases of the domain/key normal form. All have either functional, multi-valued or join dependencies that can be converted into (super)keys. The domains on those normal forms were unconstrained so all domain constraints are satisfied. However, transforming a higher normal form into domain/key normal form is not always a dependency-preserving transformation and therefore not always possible.

syllabusx.live

Example

Consider relations CAR (MAKE, vin#) and MANUFACTURE (vin#, country),

Where vin# represents the vehicle identification number 'country' represents the name of country where it is manufactured.

A general constraints may be of the following form:

If the MAKE is either 'HONDA' or 'MARUTI' then the first character of the vin# is a 'B' if the country of manufacture is 'INDIA'

If the MAKE is 'FORD' or 'ACCURA', the second character of the vin# is a 'B' if the country of manufacture is 'INDIA'

There is no simplified way to represent such constraints short of writing a procedure or general assertion to test them. Hence such procedure needs to enforce an appropriate integrity constraint. However, transforming a higher normal form into domain/key normal form is not always a dependency-preserving transformation and these are not possible always.

Q. 2.(c) State and explain BCNF. **(4.5)**

Ans. Boyce-Codd Normal Form (BCNF)

Boyce-Codd Normal form is a stronger generalization of third normal form. A table is in Boyce-Codd Normal form if and only if at least one of the following conditions are met for each functional dependency A → B:

- A is a superkey
- It is a trivial functional dependency.

Let us first understand what a superkey means. To understand BCNF in DBMS, consider the following BCNF example table:

Course code	Course venue	Instructor Name	Instructor's phone number
CS101	Lecture Hall 20	Prof. George	+1 6514821924
CS152	Lecture Hall 21	Prof. Atkins	+1 6519272918
CS154	CS Auditorium	Prof. George	+1 6514821924

Here, the first column (course code) is unique across various rows. So, it is a superkey. Consider the combination of columns (course code, professor name). It is also unique across various rows. So, it is also a superkey. A superkey is basically a set of columns such that the value of that set of columns is unique across various rows. That is, no 2 rows have the same set of values for those columns.

Some of the superkeys for the table above are:

- Course code
- Course code, professor name
- Course code, professor mobile number

A superkey whose size (number of columns) is the smallest is called as a candidate key. For instance, the first superkey above has just 1 column. The second one and the last one have 2 columns. So, the first superkey (Course code) is a candidate key.

Boyce-Codd Normal Form says that if there is a functional dependency A → B, then either A is a superkey or it is a trivial functional dependency. A trivial functional dependency means that all columns of B are contained in the columns of A. For instance, (course code, professor name) → (course code) is a trivial functional dependency because when we know the value of course code and professor name, we do know the value of course code and so, the dependency becomes trivial.

Let us understand what's going on:

A is a superkey: this means that only and only on a superkey column should it be the case that there is a dependency of other columns. Basically, if a set of columns (B) can be determined knowing some other set of columns (A), then A should be a superkey. Superkey basically determines each row uniquely.

It is a trivial functional dependency: this means that there should be no non-trivial dependency. For instance, we saw how the professor's department was dependent on the professor's name. This may create integrity issues since someone may edit the professor's name without changing the department. This may lead to an inconsistent database.

Q.3.(a) Discuss how each of the following constructs is used in SPL and discuss the various options of each constructs:

(a) Nested Queries **(2.5)**

Ans. Nested Queries- A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause.

A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.

There are a few rules that subqueries must follow

- Subqueries must be enclosed within parentheses.
- A subquery can have only one column in the SELECT clause, unless multiple columns are in the main query for the subquery to compare its selected columns.
- An ORDER BY command cannot be used in a subquery, although the main query can use an ORDER BY. The GROUP BY command can be used to perform the same function as the ORDER BY in a subquery.
- Subqueries that return more than one row can only be used with multiple value operators such as the IN operator.
- The SELECT list cannot include any references to values that evaluate to a BLOB, ARRAY, CLOB, or NCLOB.
- A subquery cannot be immediately enclosed in a set function.
- The BETWEEN operator cannot be used with a subquery. However, the BETWEEN operator can be used within the subquery.

Subqueries are most frequently used with the SELECT statement. The basic syntax is as follows

```
SELECT column_name [, column_name ]
FROM   table1 [, table2 ]
WHERE  column_name OPERATOR
  (SELECT column_name [, column_name ]
  FROM table1 [, table2 ]
  [WHERE])
```

syllabusx.live

Q.3.(b) SQL like clause **(2.5)**

Ans. SQL Like clause:

The SQL LIKE clause is used to compare a value to similar values using wildcard operators. There are two wildcards used in conjunction with the LIKE operator.

- The percent sign (%)
- The underscore (_)

The percent sign represents zero, one or multiple characters. The underscore represents a single number or character. These symbols can be used in combinations.

Syntax

The basic syntax of % and _ is as follows :-

SELECT FROM table_name

WHERE column LIKE 'XXXX%'

or

SELECT FROM table_name

WHERE column LIKE '%XXXX%'

or

SELECT FROM table_name

WHERE column LIKE '_XXXX_'

We can combine N number of conditions using AND or OR operators. Here, XXXX could be any numeric or string value.

Q.3.(c) SQL group by clause

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*;

Q.3. (d) In operator

Ans. IN Operator

The IN operator allows us to specify multiple values in a WHERE clause.

The IN operator is a shorthand for multiple OR conditions.

IN Syntax:

SELECT *column_name(s)*

FROM *table_name*

WHERE *column_name* IN (*value1*, *value2*, ...);

or

SELECT *column_name(s)*

FROM *table_name*

WHERE *column_name* IN (*SELECT STATEMENT*);

Example –

Select * From EmployeeAddressTable

Where FirstName IN ('Mary', 'Sam')

This would return all rows where the FirstName is either Mary or Sam. Because the * is used in the select, all fields from the EmployeeAddressTable table would appear in the result set.

Q. 3. (e) Triggers

Ans. Refer Q. 4.(*a*).First Term Feb -2016. [Page No. 6-2016].

Q. 4. (a) Why are duplicate tupples not allowed in a relation? **(3)**

Ans. Duplicate tuples are not allowed in relation

Because it violates the specifications of the relational integrity constraints, particularly the key constraint which states that no two **tuples** can have the same values for their attributes at any **relation** state of a database.

Q. 4.(b) Discuss the entity integrity and referential integrity constraints. **(3)**

Ans. Entity Integrity Constraint: Entity means any place, thing or person in database. An entity represents a real world object. Each table represents an entity and each row in a table is the instance of entity.Entity Integrity constraint uniquely identifies each row in table. Primary key assures the entity integrity constraint is applied to a table.

It also states that value in primary key fields cannot be null.

For example, Let us take example of table Employee having columns: Emp_Id, Name, Address, Pincode, Passport_Number, Salary.

emp_Id	Name	Address	Pincode	Passport_Number	Salary

Let's say Emp_Id is primary key in the table. Thus from the definition of Entity Integrity, the value of Emp_Id cannot be null as it unique identifies an employee record in the table.

Thus no primary key column of any row in a table can have a null value.

Referential Integrity Constraint: Referential integrity constraint refers to relationship between the tables. The relationship between tables is established by using foreign keys. This constraint states that a foreign key must have a matching primary key in another table or must be null.

For example, let's say we have table SC that tells us which student study in which class. We also have Student and Class tables.

Table SC

student_Id	Class_Id
1	2
2	1
3	4
4	1
5	3
6	4

syllabusx.live

Table-Student

Student_id	Student_Name	Address	Father_Name
1	Abhishek Kumar	Indira Colony	Ramesh Kumar
2	Mohit Sharma	MG Road	Neeraj Sharma
3	Rajan Gupta	Subash Colony	Raj Mohan Gupta
.....	.....	.....	.....

Table-Class

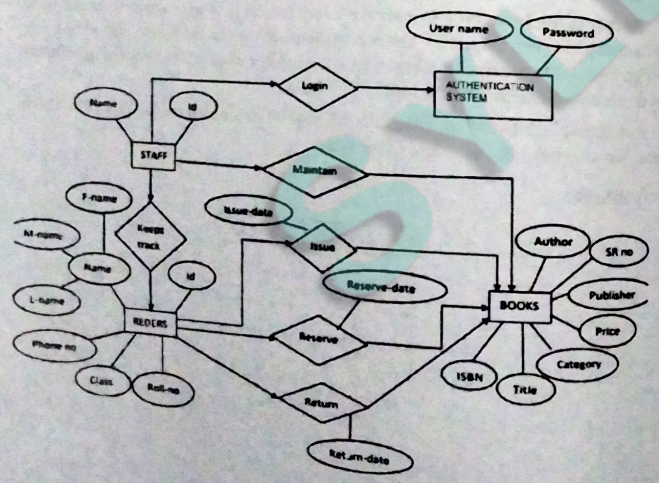
Class_Id	Class_Name
1	Class 1
2	Class X
3	Class XI
.....	.....

- The Student_Id in any row of the SC table corresponds to a Student_Id of the Student table.
- The Class_Id in any row of the SC table corresponds to a Class_Id of the Class table.

Thus every foreign key must contain a null value or a valid key reference in another table.

Q.4. (c) Draw an ER diagram for a Library management system. Take the entities and attributes to satisfy the requirements for such a system. Convert the ER diagram into tables. **(6.5)**

Ans. The entity relationship model is a popular high-level conceptual data model. It is a detailed, logical representation of data or an organization for a business area. The ER model is very useful for mapping the meanings and interactions of real world enterprises onto a conceptual schema the E- R diagram drawn below shows the entities in library management system, the relationship among the entities and attributes of both entities and their relationships



Tables Listing:

Issue: Table

Sr. No.	Field Name	Data Type	Size
1	Bno	int	25
2	Id	int	30
3	Issue_date	Date/time	10

Admin : Table

Sr. No.	Field Name	Data Type	Size
1	Name	Char	25
2	Date	int	10

Student : Table

Sr. No.	Field Name	Data Type	Size
1	Id	int	30
2	Roll_no	int	3

Teacher: Table

Sr. No.	Field Name	Data Type	Size
1	Id	int	30
2	Name	Char	25

Book : Table

Sr. No.	Field Name	Data Type	Size
1	B_no	int	10
2	ISBN	int	10
3	Subject	Char	10

Issue Return: Table

Sr. No.	Field Name	Data Type	Size
1	B_no	int	11
2	Id	int	20
3	Issue_date	Date/time	10

Q. 5. (a) How does a B tree differs from a B⁺ Tree? Why is a B⁺ Tree usually preferred as an access structure to a data file? **(6.5)**

Ans. 1- Difference between B tree and B+ tree

	B Tree	B⁺ Tree
Short web discription	A B tree is an organizational structure for information storage and retrieval in the form of a tree in which all terminal nodes are at the same distance from the base, and all non-terminal nodes have between n and 2 n sub-trees or pointers (where n is an integer).	B⁺ tree is an n-array tree with a variable but often large number of children per node. A B⁺ tree consists of a root, internal nodes and leaves. The root may be either a leaf or a node with two or more children.
Also known as Space Search Insert Delete	Balanced tree. O(n) O(log n) O(log n) O(log n)	B plus tree. O(n) $O(\log_b n)$ $O(\log_b n)$ $O(\log_b n)$

syllabusx.live

Storage	In a B tree, search keys and data stored in internal or leaf nodes.	In a B^+ tree, data stored only in leaf nodes.
Data	The leaf nodes of the three store pointers to records rather than actual records.	The leaf nodes of the tree stores the actual record rather than pointers to records.
Space	These trees waste space	There trees do not waste space.
Function of leaf nodes	In B tree, the leaf node cannot store using linked list.	In B^+ tree, leaf node data are ordered in a sequential linked list.
Searching	Here, searching becomes difficult in B- tree as data cannot be found in the leaf node.	Here, searching of any data in a B^+ tree is very easy because all data is found in leaf nodes.
Search accessibility	Here in B tree the search is not that easy as compared to a B^+ tree.	Here in B^+ tree the searching becomes easy.
Redundant key	They do not store redundant search key.	They store redundant search key.
Applications	They are an older version and are not that advantageous as compared to the B+ trees.	Many database system implementers prefer the structural simplicity of a B^+ tree.

The drawback of B-tree used for indexing, however is that it stores the data pointer (a pointer to the disk file block containing the key value), corresponding to a particular key value, along with that key value in the node of a B-tree. This technique, greatly reduces the number of entries that can be packed into a node of a B-tree, thereby contributing to the increase in the number of levels in the B-tree, hence increasing the search time of a record.

B^+ tree eliminates the above drawback by storing data pointers only at the leaf nodes of the tree. Thus, the structure of leaf nodes of a B^+ tree is quite different from the structure of internal nodes of the B^+ tree. It may be noted here that, since data pointers are present only at the leaf nodes, the leaf nodes must necessarily store all the key values along with their corresponding data pointers to the disk file block, in order to access them. Moreover, the leaf nodes are linked to provide ordered access to the records. The leaf nodes, therefore form the first level of index, with the internal nodes forming the other levels of a multilevel index.

Q.5.(b) What is partitioned hashing? How does it work? What are its limitations?

Ans. Partitioned Hashing **(6)**

Hash partitioning is a method to separate out information in a randomized way rather than putting the data in the form of groups. This partitioning system can be used efficiently to manage data on a particular platform. However, there are no performance benefits associated with hash partitioning, as it shuffles the data across the table space randomly.

The partitioning system can be used to efficiently match queries. It makes use of hashing algorithms to distribute the data across the device to space out the load. By

this method, the partitions are approximately the same size. The data that can be partitioned is not historical in nature, and thus this method is very easy to use.

As a general rule, use hash partitioning for the following purposes:

- To enable partial or full parallel partition-wise joins with likely equisized partitions.
- To distribute data evenly among the nodes of an MPP platform that uses Oracle Real Application Clusters. Consequently, you can minimize interconnect traffic when processing internode parallel statements.
- To use partition pruning and partition-wise joins according to a partitioning key that is mostly constrained by a distinct value or value list.
- To randomly distribute data to avoid I/O bottlenecks if you do not use a storage management technique that stripes and mirrors across all available devices.

Example –

Creating a table with Hash Partitioning:

```
CREATE TABLE sales_hash
(s_productid  NUMBER,
s_saledate   DATE,
s_custid     NUMBER,
s_totalprice NUMBER)
PARTITION BY HASH(s_productid)
( PARTITION p1 TABLESPACE tbs1
, PARTITION p2 TABLESPACE tbs2
, PARTITION p3 TABLESPACE tbs3
, PARTITION p4 TABLESPACE tbs4
);
```

Q. 6. (a) What do you understand by a transaction? In what situation a transaction is said to be committed or aborted? **(4)**

Ans. A transaction can be defined as a group of tasks. A single task is the minimum processing unit which cannot be divided further.

Let's take an example of a simple transaction. Suppose a bank employee transfers Rs 500 from A's account to B's account. This very simple and small transaction involves several low-level tasks.

A's Account

Open_Account(A)

Old_Balance = A.balance

New_Balance = Old_Balance - 500

A.balance = New_Balance

Close_Account(A)

B's Account

Open_Account(B)

Old_Balance = B.balance

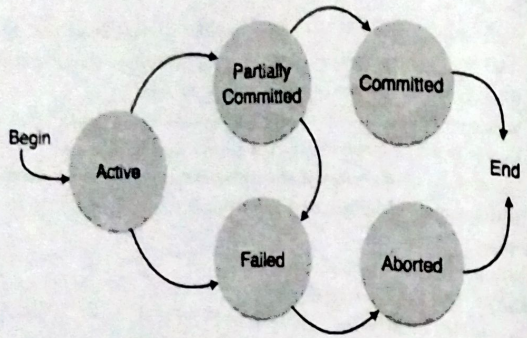
syllabusx.live

New_Balance = Old_Balance + 500

B.balance = New_Balance

Close_Account(B)

A transaction in a database can be in one of the following states–



- **Active** – In this state, the transaction is being executed. This is the initial state of every transaction.
- **Partially Committed** – When a transaction executes its final operation, it is said to be in a partially committed state.
- **Failed** – A transaction is said to be in a failed state if any of the checks made by the database recovery system fails. A failed transaction can no longer proceed further.
- **Aborted** – If any of the checks fails and the transaction has reached a failed state, then the recovery manager rolls back all its write operations on the database to bring the database back to its original state where it was prior to the execution of the transaction. Transactions in this state are called aborted. The database recovery module can select one of the two operations after a transaction aborts–
  - o Re-start the transaction
  - o Kill the transaction
- **Committed** – If a transaction executes all its operations successfully, it is said to be committed. All its effects are now permanently established on the database system.

Q.6.(b) What is a serializable schedule? How it differs from a serial schedule? **(4)**

Ans. Refer Q.5.(b). First Term Feb – 2017. [Page no - 2017-7]

Q.6.(c) Discuss the ACID properties for a transaction. **(4.5)**

Ans. Refer Q.1.(f). End Term june – 2017. [Page no - 2017-9]

Q. 7. Write short notes on the following: **(4)**

(a) Database tunning.

Ans. Refer Q.8.(a). End Term june – 2017. [Page no - 20-2017]

Q.7.(b) Cursors **(4)**

Ans. Cursors –

A cursor is a temporary work area created in the system memory when a SQL statement is executed. A cursor contains information on a select statement and the rows of data accessed by it.

This temporary work area is used to store the data retrieved from the database, and manipulate this data. A cursor can hold more than one row, but can process only one row at a time. The set of rows the cursor holds is called the *active* set.

There are two types of cursors in PL/SQL:

Implicit cursors

These are created by default when DML statements like, INSERT, UPDATE, and DELETE statements are executed. They are also created when a SELECT statement that returns just one row is executed.

Explicit cursors

They must be created when you are executing a SELECT statement that returns more than one row. Even though the cursor stores multiple records, only one record can be processed at a time, which is called as current row. When you fetch a row the current row position moves to next row.

Both implicit and explicit cursors have the same functionality, but they differ in the way they are accessed.

Example–

```
DECLARE var_rows number(5);
BEGIN
 UPDATE employee
 SET salary = salary + 1000;
 IF SQL%NOTFOUND THEN
dbms_output.put_line('None of the salaries where updated');
 ELSIF SQL%FOUND THEN
var_rows := SQL%ROWCOUNT;
dbms_output.put_line('Salaries for' || var_rows || 'employees are updated');
 END IF;
 END;
```

In the above PL/SQL Block, the salaries of all the employees in the 'employee' table are updated. If none of the employee's salary are updated we get a message 'None of the salaries where updated'. Else we get a message like for example, 'Salaries for 1000 employees are updated' if there are 1000 rows in 'employee' table.

Q.7.(c) Deadlocks

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 lack 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."

syllabusx.live

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:

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 sysem must maintain information about the current allocation of data items to transactions provide on appropriate algorithm to find if the system has entered 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.

FIRST TERM EXAMINATION [FEB. 2019]
FOURTH SEMESTER [B.TECH]
DATABASE MANAGEMENT SYSTEM
[ETCS–208]

Time : 1½ hrs. **M.M. : 30**

Note: Q.No. 1. is compulsory. Attempt any two more questions from the rest.

Q.1. (a) What is logical data independence and why is it important. (2.5)

Ans. 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.

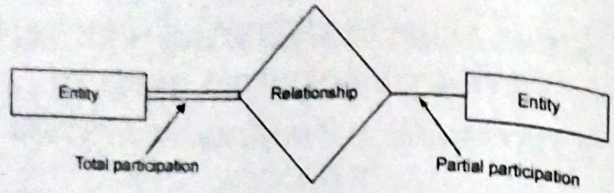
2. Logical data independence is the ability to modify the logical schema without causing application program to be rewritten. Modifications at the logical level are necessary whenever the logical structure of the database is altered (for example, when money-market accounts are added to banking system). 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.

Q. 1. (b) Explain the distinction between total and partial participation.(2.5)

Ans. • Total Participation: Each entity is involved in the relationship. Total participation is represented by double lines.

• Partial participation: Not all entities are involved in the relationship. Partial participation is represented by single lines.

syllabusx.live



Q. 1. (c) Define

(i) Entity Integrity. (2.5)

(ii) Referential Integrity.

Ans. Refer Q.4.(b), End Term Examination 2018.

Q. 1. (d) Differentiate between LEFT OUTER JOIN and RIGHT OUTER JOIN. (2.5)

Ans.

Basis for Comparison	Left outer join	Right Outer Join
Basic	All the tuples of the left table remain in the result.	All the tuples of the right table remain in the result.
NULL Extension	The tuples of left table that does not have a matching tuple in right table are extended with NULL value for attributes of the right table.	The tuples of right table that does not have a matching tuple in left table are extended with NULL value for attributes of the left table.

Q. 2. (a) Explain disadvantages of traditional file system. (5)

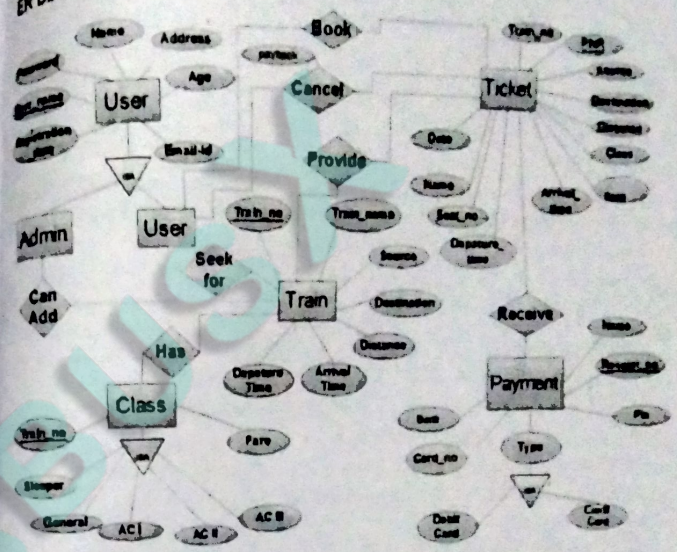
Ans. Disadvantages of Traditional File System

- Data redundancy (Each application has its own data file so, same data may have to be recorded and stored in many times).
- Data inconsistency (Due to the same data items that appear in more than one file do not get updated simultaneously in each and every file).
- Data dependence (Program and application in the file processing system are data dependent but, the problem is incompatible with file format).
- Limited data sharing.
- The problem with security.
- Retrieval (retrieval is not easy).
- Time consuming.
- Inefficient to maintain the record of the big firm having a large number of items.
- Required Lots of labor work to do.

Q. 2. (b) Draw E-R Diagram for railway Reservation System. (5)

Ans.

ER DIAGRAM



Q. 3. Consider the following relations containing airline flight information.

Flights (flno: integer, from : string, to: string, distance: integer, departs: time, arrives: time)

Aircraft(aid: integer, aname: string, cruisingrange: integer)

Certified(eid: integer, aid: integer)

Employees (eid: integer, ename: string, salary: integer)

Write the following queries in Relational algebra OR Structured Query Language (SQL).

Q.3. (a) Find the eids of the pilots certified for some Boeing aircraft. (2)

Ans. SELECT C.eid FROM Aircraft A, Certified C WHERE A.aid = C.aid AND A.aname = 'Boeing'

Q.3. (b) Find the name of the pilots certified for some Boeing aircraft. (2)

Ans. SELECT E.ename FROM Aircraft A, Certified C, Employees E WHERE A.aid = C.aid AND A.aname = 'Boeing' AND E.eid = C.eid

Q.3. (c) Find the aids of all the aircraft that can be used on non-stop flights from Bonn to Madrid. (2)

Ans. ρ(Bonn To Madrid, $\pi_{distance}(\sigma_{from = 'Bonn' \wedge to = 'Madrid'}$ Flight))

π_{aid}(Aircraft $\bowtie_{cruisingrange>distance}$ BonnToMadrid)

SELECT A.aid

FROM Aircraft A, Flight F

WHERE F.from='Bonn' AND F.to='Madrid', AND A.cruisingrange >=F.distance

syllabusx.live

Q.3. (d) Identify the flights that can be piloted by every pilot whose salary is more than Rs. 100,000. **(2)**

Ans. $\rho(Temp, \pi_{aid}((\sigma_{salary > 100,000} Employees) \triangleright\triangleleft Certified))$

$\pi_{Flno}(\sigma_{distance \geq cruisingrange}(Temp \triangleright\triangleleft Aircraft \triangleright\triangleleft Flight))$

```
SELECT F.flno
FROM Aircraft A, Certified C, Employees E, Flight F
WHERE E.eid = C.eid AND C.aid = A.aid AND F.distance<= A.cruisingrange AND
E.salary > 100,000
```

Q.3. (e) Find the name of the pilots who can operate planes with a range greater than 3,000 miles but are not certified on any Boing aircraft. **(2)**

Ans. $\rho(Temp1, \pi_{eid}(\sigma_{cruisingrange>3,000}(Certified \triangleright\triangleleft Aircraft)))$

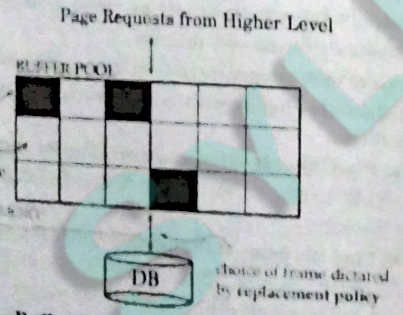
$\rho(Temp2, \pi_{eid}(\sigma_{aname = 'Boeing'}(Certified \triangleright\triangleleft Aircraft)))$

$\pi_{ename}((Temp1 - Temp2) Employees)$

```
SELECT E.ename
FROM Certified C, Employees E, Aircraft A
WHERE A.aid = C.aid AND E.eid = C.eid AND A.cruisingrange > 3000
AND E.eid NOT IN (SELECT C2.eid
FROM Certified C2, Aircraft A2
WHERE C2.aid = A2.aid AND A2.aname = 'Boeing')
```

Q. 4. (a) Define **(1)**

Ans. (i) Buffer Manager: The buffer manager is the software layer that is responsible for bringing pages from physical disk to main memory as needed. The buffer manages the available main memory by dividing the main memory into a collection of pages, which we called as buffer pool. The main memory pages in the buffer pool are called frames.



Buffer Management in a DBMS

Buffer Management in a DBMS

- Data must be in RAM for DBMS to operate on it!
- Buffer manager hides the fact that not all data is in RAM.

(ii) Transaction Manager: **(1)**

Ans. A transaction manager is a part of an application that controls the coordination of transactions over one or more resources. The transaction manager is responsible for

resting transaction objects and managing their durability and atomicity. Transaction managers keep track of all the resource managers that are enlisted in a transaction.

A transaction manager maintains a log in storage. This log is usually a sequential file, which is used to record the transaction events. Transaction managers use this log for recording transaction starts, commit decisions, enlistments, etc. At the time of normal processing, transaction managers simply write to the log, but if they fail, they read the log as soon as it restarts to restore the latest state.

(iii) Candidate Key **(1)**

Ans. Candidate Key: (*i*) When more than one or group of attributes serve as a uniqueness indentifier, they are called as candidate key.

(*ii*) Candidate key are those attributes of a relation which posses the properties of uniqueness an Irreduceability.

Uniqueness Property: It means that at any given time, no two tuples in the value R, at that time are duplicates of one another.

Ir-reduceability 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.

(iv) Metadata

Ans. Metadata is data that describes other data. Meta is a prefix that in most information technology usages means "an underlying definition or description."

Metadata summarizes basic information about data, which can make finding and working with particular instances of data easier. For example, author, date created and date modified and file size are examples of very basic document metadata. Having the abilty to filter through that metadata makes it much easier for someone to locate a specific document.

In addition to document files, metadata is used for images, videos, spreadsheets and web pages. The use of metadata on web pages can be very important. Metadata for web pages contain descriptions of the page's contents, as well as keywords linked to the content. These are usually expressed in the form of metatags. **(1)**

(v) Instances

Ans. The data stored in database at a particular moment of time is called instance of database. Database schema defines the variable declarations in tables that belong to a particular database; the value of these variables at a moment of time is called the instance of that database.

For example, lets say we have a single table student in the database, today the table has 100 records, so today the instance of the database has 100 records. Lets say we are going to add another 100 records in this table by tomorrow so the instance of database tomorrow will have 200 records in table. In short, at a particular moment the data stored in database is called the instance, that changes over time when we add or delete data from the database.

Q. 4. (b) Explain the concept of aggregation with the help of example. **(5)**

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 entities. This special kind of relationship is called as an aggregation.

syllabusx.live

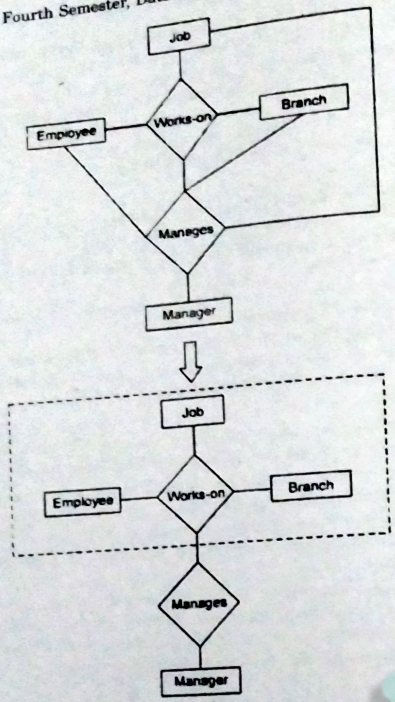


Fig. Aggregation concept.

END TERM EXAMINATION [MAY. 2019]
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 compulsoy.*

Q.1. Explain in brief (any five)

Q.1. (a)List the properties of Relations. **(5)**

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

• Order of columns is immaterial.

• In a table, there are no dupplicate 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.

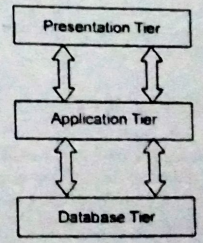
Q.1. (b) Explain Three level Database Abstraction. **(5)**

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.1. (c) Differentiate between weak entity and Strong Entity in ER diagram through an example. **(5)**

SYLLABUS
syllabusxlive

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. (d) Differentiate between 3NF and BCNF. **(5)**

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 BCXF 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. (e) What do you mean by recoverable and non-recoverable schedules? Give example of each. **(5)**

Ans. Recoverable Schedules:

• Schedules in which transactions commit only after all transactions whose changes they read commit are called recoverable schedules. In other words, if some transaction T_j is reading value updated or written by some other transaction T_i, then the commit of T_j must occur after the commit of T_i.

Example 1:

• S1: R1(x), **W1(x)**, R2(x), R1(y), R2(y),

W2(x), W1(y), **C1**, **C2**,

Given schedule follows order of T_i->T_j => **C1->C2**. Transaction T1 is executed before T2 hence there is no chances of conflict occur. R1(x) appears before W1(x) and transaction T1 is committed before T2 i.e. completion of first transaction performed first update on data item x, hence given schedule is recoverable.

Irrecoverable Schedule:

• The table below shows a schedule with two transactions, T1 reads and writes A and that value is read and written by T2. T2 commits. But later on, T1 fails. So we have

to rollback T1. Since T2 has read the value written by T1, it should also be rollbacked. But we have already committed that. So this schedule is irrecoverable schedule. When T_j is reading the value updated by T_i and T_j is committed before committing of T_i, the schedule will be irrecoverable.

T1	T1's buffer space	T2	T2's buffer Space	Database
				A=5000
R(A)	A=5000			A=5000
A=A-100	A=4000			A=5000
W(A)	A=4000			A=4000
		R(A)	A=4000	A=4000
		A=A+500	A=4500	A=4000
		W(A)	A=4500	A=4500
		Commit		
Failure Point				
Commit				

Q.1. (f) What do you mean by multi valued attributes? How they are represented? **(5)**

Ans. Multivalued Attributes

A multivalued attribute of an entity is an attribute that can have more than one value associated with the key of the entity. For example, a large company could have many divisions, some of them possibly in different cities. In this case, division or division-name would be classified as a multivalued attribute of the Company entity (and its key, company-name). The headquarters-address attribute of the company, on the other hand, would normally be a single-valued attribute.

To understand how MultiValued attributes are used, let's look at the structure of an ORDERS database created to capture the data on the sample form in the example below:

Purchase Order
No 03128
Date 3/18/98
Ship Via U.P.S.
Payment Cash

To John Swenson
14 Crossroads Place
Fremont, NV

Product	Quantity	Amount
x47-160	2	25.00
x47-181	2	32.00
x25-340	1	21.00
		78.00

MultiValued attributes can be used to describe line items on an order form. The highlighted area in the illustration shown below holds the line items on the printed form in the above illustration. A line item represents information about the purchase of

syllabusx.live

a particular product. The customer might place orders for several different products in different quantities with each one having a different price. The rest of the data fields on this form contain only a single piece of data, such as the customer's name and address, a credit card number or the amount of the order.

Each order has an order number (outlined in the top right hand corner of the form). This number serves as the item-ID to identify each order in the database file. Each item in the data file has the following sequence of attributes:

CUSTOMER-ID	DATE	PRODUCT	QTY	PAYMENT	SHIP
1	2	3	4	5	6

Q.2 (a) Explain through an example relation between Super key, Candidate key, Alternate key, Primary key and Foreign key. (4)

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 but not necessarily the inrreducibility property.

(iii) A candidate key is a special case of a super key,

For example:

R-No	Name	Class

A Superkey

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.

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 we can say that.

Candidate Key: A candidate key is a column, or set of columns, in a table that can uniquely identify any database record without referring to any other data. Each table may have one or more candidate keys, but one candidate key is unique, and it is called the primary key. This is usually the best among the candidate keys to use for identification.

Alternate Key: All the keys which are not primary key are called an alternate key. It is a candidate key which is currently not the primary key. However, A table may have single or multiple choices for the primary key.

Q.2. (b) What are cursors? What are the steps actions for explicit cursors? (4)

Ans. Refer Q.7. (b) End term examination 2018.

Q.2. (c) Write a trigger to count number of new tuples inserted using each insert statement. (4.5)

Ans.

```
Declare count int
Set count=0;
delimiter $$
CREATE TRIGGER Count_tupples
        AFTER INSERT ON employee
FOR EACH ROW
BEGIN
SET count = count + 1;
END; $$
delimiter;
```

Q.3. (a) What do you mean by cardinality in ER? How cardinality is handled while translating ER model to Relational model. Explain through example. (2.5,5)

Ans. 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.

(a) One-to-One (1 : 1): The relationship between a department and a Manager is usually ono-to-noe as shown in fig. (1)

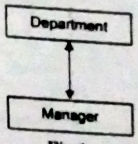


Fig.1.

Fig. (2) shows some instances of these relationships.

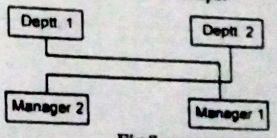


Fig.2.

(b) One-to-Many (1 : N): The relationship between manager and employee is one-to-many as swhon in fig. (3) because there are several employees reporting to the manager.

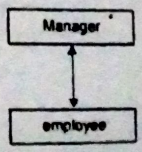


Fig.3.

syllabusx.live

Fig (4) shown some instance of these relationship.

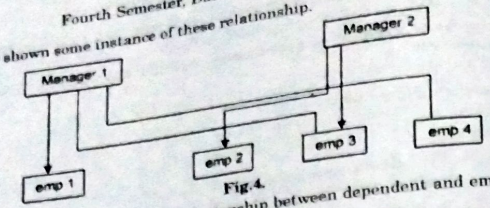


Fig.4.

(c) Many-to-One (N : 1): The relationship between dependent and employee is many-to-one as shown in fig. (5)

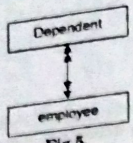


Fig.5.

(d) Many-to-Many (M : N): The relationship between Employee and project is many-to-many as shown fig. (6). Because each employee could be involved in a number of different projects and number of employees could be working on a given project.

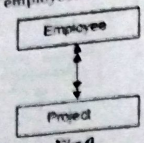


Fig.6.

Fig. (7) Given below shows some instance of these relationships.

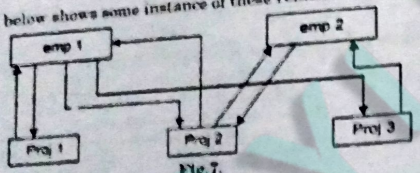


Fig.7.

Structural Constraints: Cardinality ratio and participation constraints are together framed as structural constraints on a reaction. **(5)**

Q.3. (b) How aggregation is represented in ER?

Ans. Refer Q 4. (b) of First Term Examination 2019.

Q.4. Write relational algebra/SQL for the following relational schema. (12.5)

Ans. Student (id, na ode)
enrolledIn (id, code)
subject (code, lecture)

(i) What are the names of the students in ETCS 205 or ETCS 207?

(ii) What are the codes of all the subjects taught?

(iii) What are the names of students who are taking a subject not taught by "Amit".

(iv) What are the names of students in at least two different subjects?

(v) Who teaches at least three different subjects?

Ans. (1) πname(σETCS205 = code(student ⋈ enrolledIn))

(2) πcode(subject)

(3) πname(σ lecturer = Amit(student ⋈ enrolledIn ⋈ subject))

(4) πname(student ⋈ (σ R.id = S.id AND R.code <> S.code(R ⋈ S)))

(5) (σ R.lecturer = S.lecturer AND R.code <> S.code(R ⋈ S))

Q.5 (a) Explain multivalued dependency through an example. (4)

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.5. (b) State and prove Pseudo transitive inference rule. (3)

Ans. In Pseudo transitive Rule, if X determines Y and YZ determines W, then XZ determines W.

1. If X → Y and YZ → W then XZ → W
1. X → Y (given)
2. WY → Z (given)
3. WX → WY (using IR_2 on 1 by augmenting with W)
4. WX → Z (using IR_3 on 3 and 2)

syllabusx.live

Q.5. (c) Find the highest normal form of relation R (A, B, C, D, E) with FD set as {BC-> D, AC-> BE, B->E}. **(8.5)**

Ans. Step 1. As we can see, $(AC)^+ = \{A,C,B,E,D\}$ but none of its subset can determine all attribute of relation, So AC will be candidate key. A or C can't be derived from any other attribute of the relation, so there will be only 1 candidate key {AC}.

Step 2. Prime attribute are those attribute which are part of candidate key {A,C} in this example and others will be non-prime {B,D,E} in this example.

Step 3. The relation R is in 1st normal form as a relational DBMS does not allow multi-valued or composite attribute.

The relation is in 2nd normal form because BC->D is in 2nd normal form (BC is not proper subset of candidate key AC) and AC->BE is in 2nd normal form (AC is candidate key) and B->E is in 2nd normal form (B is not a proper subset of candidate key AC).

The relation is not in 3rd normal form because in BC->D (neither BC is a super key nor D is a prime attribute) and in B->E (neither B is a super key nor E is a prime attribute) but to satisfy 3rd normal for, either LHS of an FD should be super key or RHS should be prime attribute.

So the highest normal form of relation will be 2nd Normal form.

Q.6. (a) Briefly explain transaction properties. **(4)**

Ans. • **Atomicity:** This property states that a transaction must be treated as an atomic unit, that is, either all of its operations are executed or none. There must be no state in a database where a transaction is left partially completed. States should be defined either before the execution of the transaction or after the execution/abortion/failure of the transaction.

• **Consistency:** The database must remain in a consistent state after any transaction. No transaction should have any adverse effect on the data residing in the database. If the database was in a consistent state before the execution of a transaction, it must remain consistent after the execution of the transaction as well.

• **Durability:** The database should be durable enough to hold all its latest updates even if the system fails or restarts. If a transaction updates a chunk of data in a database and commits, then the database will hold the modified data. If a transaction commits but the system fails before the data could be written on to the disk, then that data will be updated once the system springs back into action.

• **Isolation:** In a database system where more than one transaction are being executed simultaneously and in parallel, the property of isolation states that all the transactions will be carried out and executed as if it is the only transaction in the system. No transaction will affect the existence of any other transaction.

Q.6. (b) What is conflict serializable schedule? Check if the given schedule S1 is conflict serializable SI: R2(A), W2(A), R1(A), W1(A), R1(B), W1(b), R2(B), W2(B). **(5)**

Ans. If O_i and O_j are two operations in a transaction and $O_i < O_j$ (O_i is executed before O_j), same order will follow in schedule as well. Using this property, we can get two transactions of schedule S1 as:

T1: $R_1(A), W_1(A), R_1(B), W_1(B)$

T2: $R_2(A), W_2(A), R_2(B), W_2(B)$

Possible Serial Schedules are: T1->T2 or T2->T1

-> **Swapping non-conflicting operations** $R_2(A)$ and $R_1(B)$ in S1, the schedule becomes,

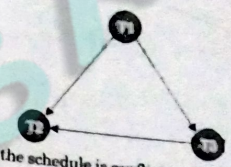
S11: $R_1(A), W_1(A), R_1(B), W_2(A), R_2(A), W_1(B), R_2(B), W_2(B)$

-> Similarly, **swapping non-conflicting operations** $W_2(A)$ and $W_1(B)$ in S11, the schedule becomes, **S12:** $R_1(A), W_1(A), R_1(B), W_1(B), R_2(A), W_2(A), R_2(B), W_2(B)$

S12 is a serial schedule in which all operations of T1 are performed before starting any operation of T2. Since S has been transformed into a serial schedule S12 by swapping non-conflicting operations of S1, S1 is conflict serializable.

Q.6. (c) Draw precedence graph of S1 to check for conflict serializable schedule. **(3.5)**

Ans. The graph for this schedule is :



Since the graph is acyclic, the schedule is conflict serializable. Performing Topological Sort on this graph would give us a possible serial schedule which is conflict equivalent to schedule S1.

In Topological Sort, we first select the node with indegree 0, which is T1. This would be followed by T3 and T2.

So, S1 is conflict serializable since it is conflict equivalent to the serial schedule T1 T3 T2.

Q.7. Write short note on following:

Q.7. (a) Materialized views. **(3)**

Ans. A materialized view is a replica of a target master from a single point in time. The master can be either a master table at a master site or a master materialized view at a materialized view site. Whereas in multimaster replication tables are continuously updated by other master sites, materialized views are updated from one or more masters through individual batch updates, known as a refreshes, from a single master site or master materialized view site.

Q.7. (b) Database Tuninig **(3)**

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.

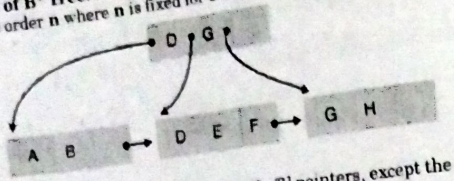
You should also pay particular attention to tuning the Uniface driver for the DBMS. Issues that play a role include a number of parameters for managing real memory required for communicating with the DBMS.

syllabusx.live

Q.7. (c) B+ tree index files

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.7. (d) Lossy Join (lossy decomposition) in dbms? (3.5)

Ans. Refer Q 1. (d) End Term Examination 2018

SYLLABUSX

syllabusx.live