CHAPTER 14

Recovery System

Module VI

Syllabus

Recovery System : Recovery Concepts, Log based recovery.

14.1 Database Recovery Concepts - System Failure

Q. Explain various types of system failure. (4 Marks)

- Like any other real world database systems also suffer from failure from a variety of causes: disk crash, power outage, software error, a fire in the machine room even rain. The result of any failure is loss of information.
- Therefore, the database system must take actions in advance to ensure that the atomicity and durability properties of transactions are preserved.
- An integral part of a database system is a recovery scheme that can restore the database to the consistent state that existed before the failure.
- Provide high availability; that is, it must minimize the time for which the database is not usable after a crash.

14.1.1 Failure Classification

Q. Explain system failure classification. (4 Marks)

A computer system, like any other electrical or mechanical system tends to fail. There are many causes, including disk crash, power failure, software errors, a fire in the machine room, or even sabotage. Whatever the cause, information may be lost. There are various types of failure that may occur in a system, each of these needs to be dealt in a different manner.

1. Hardware Failure / System crash

- There is a hardware malfunction that causes the loss of the content of volatile storage, and brings transaction processing to a halt.
- The content of non volatile storage remains intact, and is not corrupted or changed.

2. Software Failure

The database software or the operating system may be corrupted or failed to work correctly, that may cause the loss of the content of volatile storage, and brings about database failure.

3. Media failure

- A disk block loses its content as a result of either a head crash or failure during a data-transfer operation.
- Copies of the data on other disks such as tapes, CDs are used to recover from the failure.

4. Network Failure

- The problem with network interface card can cause network failure.
- There may be problem with network connection.

5. Transaction failure

There are two types of errors that may cause a transaction to fail :

a. Logical error

The transaction can no longer continue with its normal execution because of some internal condition, such as wrong input values, data not found in database, data overflow, or resource limit exceeded etc.

b. System error

The system has entered an undesirable state like deadlock; as a result transaction cannot continue with its normal execution.

6. Application software error

- The problem with software accessing the data from database.
- This may cause database failure as data cannot be updated using such application to it..

7. Physical disasters

The problem caused due to flood, fire, earthquake etc..

8. Application software error

These are some logical errors in the program that is accessing database, which cause one or more transaction failure.

14.2 Database Recovery Concepts - System Recovery

Q. Explain concept of database recovery and also explain the techniques. **(4 Marks)**

1. Database recovery is the process of restoring the database to original (correct) state as it was before database failure occurs.
2. The process of solving any type of database failures, quickly and without data loss and keep database available is called database recovery.
3. The main element of database recovery is the most recent database backup. If you maintains database backup efficiently, then database recovery is very straight forward process.

Example

To recover data from system having full backup option is as below,

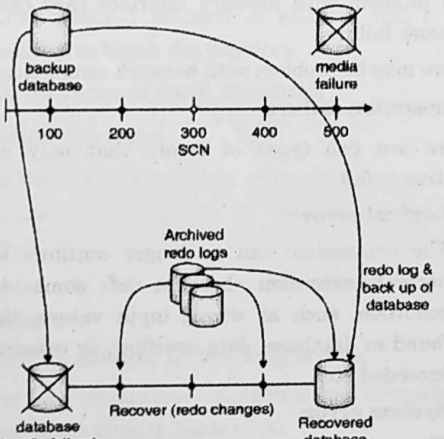

Fig. 14.2.1

Restoring entire system to a certain point may require time, depends on when last full backup taken and incremental backups which covers the period of time between the full backup and restore point.

5. Database recovery algorithms

a. During normal transaction executions ensure that enough information is backed up to allow recovery from possible failures.

b. Data should ensure that database consistency, transaction atomicity, and durability.

6. Types of database recovery / recovery phases

a. Forward database recovery

b. Backward database recovery

7. Database recovery techniques

a. Log based recovery

b. Shadow paging recovery

c. Checkpoints

14.3 Recovery and Atomicity

1. Introduction

a. Regardless of the number of concurrent transactions, the system has a single disk buffer and a single log. All transactions share this common buffer.

b. We allow immediate modification, and allow a buffer block to have data items updated by one or more transactions.

2. Concept

a. The recovery scheme depends on the concurrency-control scheme that is used in system.

b. To roll back a failed transaction, we must undo the updates performed by the transaction.

c. Suppose that a transaction T_0 has to be rolled back, and a data item Q that was updated by T_0 has to be restored to its old value.

d. Using the log-based schemes for recovery, we restore the value by using the undo information in a log record.

e. Suppose now that a second transaction T_1 has performed yet another update on Q before T_0 is rolled back.

f. Then, the update performed by T_1 will be lost if T_0 is rolled back.

g. Therefore, we require that, if a transaction T has updated a data item Q, no other transaction may update the same data item until T has committed or been rolled back.

h. We can ensure this requirement easily by using **strict two-phase locking**-that is, two-phase locking with exclusive locks held until the end of the transaction.

i. We roll back a failed transaction, T_n, by using the log. The system scans the log backward; for every log record of the form $<T_n, X_j, V_1, V_2>$ found in the log, the

system restores the data item X_j to its old value V. Scanning of the log terminates when the log record $<T_n$, start> is found.

j. Scanning the log backward is important, since a transaction may have updated a data item more than once.

3. Example

a. If **strict two-phase locking** is used for concurrency control, locks held by a transaction may be released lock only after the transaction has been rolled back. Once transaction T (that is being rolled back) has updated a data item, no other transaction could have updated the same data, because of the concurrency-control, restoring the old value of the data item will not delete any other transactions effect.

b. If we used **checkpoints** to reduce the number of log records that the system must scan to recovers from a possible crash. Since we assumed no concurrency of transactions, it was necessary to consider only the following transactions during recovery :

Transactions that started after the most recent checkpoint.

Transaction, if any, that was active at the time of the most recent checkpoint.

14.4 Log-Based Recovery

Q. Explain concept of log and log based recovery algorithm. (4 Marks)

1. Introduction

- There can be problem in accessing database due to any reason causes a database system failure.
- The most widely used structure for recording database modifications is **Transaction log (or log).**
- The log is a sequence of log records, recording all the update activities done on the database by all database users.

2. Transaction log

- Transaction log records are maintained to record various events during transaction processing.
- Transaction log file is recorded when transaction performs a write operation to record data changes.
- Log records to be useful for recovery from system and disk failures, the log must reside in stable storage.
- We assume that every log record is written to the end of the operation on stable storage as soon as log is created.
- Transactional log contains following data :
  - **a. Transaction Identifier (T_i) :** This is the unique identifier of the transaction that is recorded at write operation.
  - **b. Data item Identifier(X) :** unique identifier to recognize data item written.
  - **c. Old Value (V_1) :** Value of the old data item.
  - **d. New Value (V_2) :** Value of new data item after the write is done.

There are several types of log records.

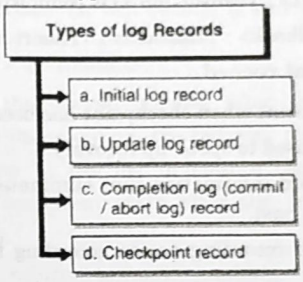

Fig. 14.4.1 : Types of log records

a. Initial log record

- To start a transaction initial transaction log is recorded.
- This log indicates that recording log file is started.
- Log Record :< T_n Start>
- Transaction (T_n) has started.

Example

<T_1 Start> : Transaction T_1 is started.

b. Update log record

- An update log record describes a single database write.
- It also includes the value of the bytes of page before and after the page change.
- **Log record :**< T_n, X, V_1, V_2>
- Transaction (T_n) has performed a write on data item X.
- It modify current value V_1 of data item X to updated value V_2 after the write.

Example

<T_1 , A, 100,500> : Transaction T_1 changed value of A to 500.

Or <T_1 , A, 500> : Transaction T_1 changed value of A to 500.

c. Completion log (commit / abort log) record

- If transaction completes operations successfully then commit a transaction or if any problem while executing transaction decision to abort and hence rollback a transaction.
- **Operational Update Log :** < T_n Commit > or < T_n Abort >
- It commits or rolls back the operations performed by transaction.

Example

<T_1 Commit> : Transaction T_1 is committed to server.

Or <T_1 Rollback> : Transaction T_1 abort its operations.

d. Checkpoint record

- Records a point when checkpoint has been made.
- These are used to speed up recovery.
- It also record information that eliminates the need to read a log's past.
- The time of recording varies according to checkpoint algorithm.
- **Log record :** < T_n Checkpoint A >
- It marks transaction status.

Example

<T_1 Checkpoint A>: Transaction T_1 is committed to server.

3. Recovery actions

Redo operation

- If a transaction has recorded commit operation before failure occurs, then transaction must be redone.
- Recovery Action : **REDO** (T_i)

Example

REDO (T_1)

Undo operation

- If a transaction has not recorded commit operation before failure occurs, then transaction must be undone.
- Recovery Action : **UNDO** (T_i)

Example : UNDO (T_1)

14.4.1 Deferred-Modification Technique (REDO Algorithm)

Q. Explain Deferred log based in detail. **(4 Marks)**

1. Introduction

- The deferred-modification technique will record database modifications in the transaction log files after transaction partially commits.
- **Partial commit :** When the final action of the transaction has been executed a transaction said to be partially committed.
- The version of the deferred-modification technique that we describe assumes that serial execution of transactions.
- Transaction perform partially commit,
  - Transaction log is recorded.
  - If the system crashes before the transaction commit or if the transaction aborts, then the information on the log will not be recorded.

2. Working

The execution log based recovery of transaction T_n as follows,

a. T_n starts its execution by writing < T_n **start>** to transaction log file.

b. If Transaction performs (T_n) performs write(X) operation it will record a **new operational log**.

c. If T_n **commits data to server**, than a record < T_n commit> is written to transaction log.

d. Ensure that all the transactional log record are written to stable storage. As it is possible to that database may fail at any point of time.

e. Once they have been written, the actual updating takes place to server then it will be in committed state.

3. Example

- Consider a simple banking system. Let T_0 be a transaction that transfers Rs.50 from account A to account B :

T_0
read(A)
A :=A -50
write(A)
read(B)
B:= B + 50
write(B)

- Let T_1 be a transaction that withdraws Rs. 100 from account C.

T_1
read(C)
C:= C + 100
write(C)

- Suppose that these transactions are executed serially, in the order T_0 followed by T_1, and that the values of accounts A, B, and C before the execution took place were Rs.100, Rs.100, and Rs.100, respectively.
- The portion of the log containing the relevant information on these two transactions appears as below,

<T_0 start>
<T_0, A, 50>
<T_0,B, 150>
<T_0 commit>
<T_1 start>
<T_1, C, 200>
<T_1 commit>

Fig. 14.4.2 : Database log corresponding to T_0 an T_1

- There are various orders in which the actual outputs can take place to both the database system and the log as a result of the execution of T_0 and T_1. as shown above.
- The value of A is changed in the database only after the record <T_0, A, 50> has been placed in the log.

Log	Database
<T_0 start >	
<T_0, A, 50 >	
<T_0, B 150 >	
<T_0 Commit>	
	A = 50
	B = 150
<T_1 Start>	
<T_1 C, 200>	
<T_1 Commit>	
	C = 200

Fig. 14.4.3

- Transaction log are used to handle any failure that results in the loss of data.
- The recovery scheme uses the following recovery procedure.

1. Method 1

- Use two lists of transactions: the committed transaction since the last checkpoint and the active transaction.
- Apply the REDO operation to all the WRITE operations of the committed transactions from the log in the order in which they were to the log.
  - Restart the active transaction. REDO(Tn) sets the value of all data items updated by transaction Tn to the new values.
  - The data items updated by Tn and their new value is updated to the log file.
  - The redo operation must be executing once.
  - After a failure, the recovery system contacts log based recovery to check for recovery action.
  - Transaction Tn will redone if and only if the log contains both the record < Tn start> and the record < Tn commit>.
- Thus, if the system crashes after the transaction completes its execution, the recovery scheme uses the information in the log to restore the system to a previous consistent state after the transaction had completed.
- Let us return to our banking example with transactions T_0 and T_1 executed one after the other in the order T_0 followed by T_1. Fig. 14.4.3 shows the log those results from the complete execution of T_0 and T_1.

<T_0 start>
<T_0,A, 50>
<T_0, B, 150>

Fig. 14.4.4(a) : Same log at different times - no commit recorded

2. Method 2

A. System crashes before the completion of the transactions

a. When the system starts after failure, no redo action is required. As, no commit record appears in the log.

b. All data values of A & B remain same.

c. The log records of the incomplete T_0 can be deleted from the log.

<T_0 start>
<T_0, A, 50>
<T_0, B, 150>
<T_0 commit>
<T_1 start>
< T_1, C, 200>

Fig. 14.4.4(b) : < T_0 commit > recorded

B. The crash comes just after the log record

a. For the step WRITE(C) of transaction T_1 has been written to stable storage.

b. When the system comes back up, the operation redo (T_0) is performed since the record < commit> appears in the log on the disk.

c. After this operation is executed T_0, the values A and B are Rs.50 and Rs.150, respectively.

d. The value of account C remain before, the log records of the incomplete transaction T_1 can be deleted from the log.

<T_0 start>
<T_0, A, 50>
<T_0, B,150>
<T_0 commit>
<T_1 start>
<T_1 C, 200>
<T_1 commit>

Fig. 14.4.4(c)

C. A crash occurs just after the log record < T_1 commit > is written to is stable storage

a. The log at the time of this crash is as in Fig.14.4.4(c) the system comes back up.

b. Two commit records are in the log: one for T_1 and one for T_0.

c. Therefore, the system must perform operations redo (T_0) and redo (T_1) in the order in which their commit records appear in the log.

d. After the system executes these operations, the values of accounts A, B, and C, are Rs.50, Rs.150, and Rs.200.

Finally, let us consider a case in which a second system crash occurs after recovery from the first crash.

4. Summary

- For each commit record <T_n commit>found in the log, the system performs the operation redo (T_n). In other words, it restarts the recovery actions from the beginning.
- Since redo writes values to the database independent of the values currently in the database, the result of a successful second attempt at 'redo' is the same as though 'redo' had succeeded the first time.

14.4.2 Immediate-Modification Technique (UNDO Algorithm)

Q. Explain Immediate log base in detail. (4 Marks)

1. Introduction

- The **immediate-modification technique** writes database modifications to be written to the database as soon as it is performed or in the active state.
- In the event of transaction failure, the system makes use of old-value of the log records to restore the data items.

2. UNDO operation

- Before a transaction T_n starts its execution, the system writes the record <T_n start> to the log.
- During transaction execution, any WRITE(X) **operation by T_n** is preceded by the writing of the appropriate new update record to the log.
- When **T_n partially commits**, the system writes the record <T_n commit> to the log.
- Since the information in the log is used in restoring the original database state, we cannot allow the actual update to the database to take place before the corresponding log record is written out to stable storage.
- We therefore require that, before the execution of an output (B) operation its log records is written to stable storage.
- This procedure is defined as follows :

1. Use two lists of transactions maintained by the system;
   a. Committed transactions since the last checkpoint
   b. Active transactions
2. Undo all the WRITE operations of the active transaction from the log, using the UNDO procedure
3. Redo the WRITE operations of transaction which are committed.

UNDO Operation

Undo WRITE operation : It consists of check its log entry of write T and reading Old value and setting the value of item X in the database to old value.

3. Example

- Consider a simple banking system, with transactions T_0 and T_1 executed one after the other in the order T_0 followed by T_1.

- Fig. 14.4.5(a) shows possible order of execution in both the database and the log as a result of the execution of T_0 and T_1.

<T_0 start>
<T_0, A, 100, 50>
<T_0, B, 100, 150>
< T_0 commit>
<T_1 start>
<T_1, C, 100, 200>
<T_1 commit>

Fig. 14.4.5(a)

Log	Database
<T_0 start> <T_0, A, 1200, 950> <T_0, B, 2000,2050>	
	A = 950 B = 2050
<T_0 commit> <T_1 start>	
<T_1, C, 700, 800> < T_1 commit >	C = 800

Fig. 14.4.5(b)

- Using the log file, the system can handle any failure that does not result in the loss of information in non volatile storage.
- The recovery scheme uses two recovery procedures.

1. Method 1

a. Undo(T_n) restores the value of all data items updated by transaction T_n to the old values and redo(T_n) sets the value of all data items updated by transaction T_n to the new values.

b. The set of data items updated by T_n and their respective old and new values can be found in the log.

c. The undo and redo operations must guarantee to correct behaviour, even if a failure occurs during the recovery process.

d. After a failure has occurred, the recovery scheme consults the log to determine which transactions need to be redone, which need to be undone :

- Transaction T_n needs to be undone if the log contains the record < T_n start>, but does not contain the record <T_n commit>.
- Transaction T_n needs to be redone if the log contains both the record < T_n start> and the record <T_n commit>.
- In our banking example, with transaction T_0 and T_1 executed one after the other in the order T_0 followed by T_1.

<T_0 start>
<T_0, A, 100, 50>
<T_0, B, 100, 150>

Fig. 14.4.6(a)

2. Method 2

a. If system crashes before the completion of the transactions

- First, let us assume that the crash occurs just after the log record for the step of transaction T_0 has been written to stable storage.
- When the system comes back up, it finds the record <T_0 start> in the log, but no corresponding <T_0 commit> record.
- Thus, transaction T_0 must be undone, so an undo (T_0) is performed.
- As a result, the values in accounts A and B (on the disk) are restored to Rs.100 and Rs.100, respectively.

<T_0 start>
<T_0, A, 100, 50>
<T_0, B, 100, 150>
<T_0 commit>
<T_1 start>
<T_1, C, 300, 200>

Fig. 14.4.6(b)

b. If crash comes occurs after the log record for the step write (C) of transaction T_1 has been written to stable storage Recovery Action

a. UNDO(T_1)

Record <T_1 start> appears in the log, but there is no commit record for transaction <T_1 commit>. So, Transaction needs to be Undone its effects.

b. REDO(T_0)

- Log contains both the record <T_0 start> as well as <T_0 commit>. Than transaction should be written again to show its effect on transaction.
- In this example, the same outcome would result if the order were reversed.

<T_0 start>
<T_0, A, 100, 50>
<T_0, B, 100, 150>
<T_0 commit>
<T_1 start>
<T_1, C, 200, 300>
< T_1 ,commit>

Fig. 14.4.6(c)

c. If crash occurs just after the log record <T_1 commit> has been written to stable storage.

- When the system comes to backup, both T_0 and T_1 need to be redone, since the records <T_0 start> and <T_0 commit> appear in the log, as do the records <T_1 start> and <T_1 commit>.
- After the system performs the recovery procedures REDO(T_0) and REDO(T_1), the values in accounts A, B, and C, are Rs.50, Rs.150, and Rs.300, respectively.

14.5 Recovery Related Structures - Checkpoint

Q. Explain concept of checkpoint with suitable example. **(4 Marks)**

1. Introduction

- A database checkpoint is where all committed transactions are written to the redo/audit logs.
- The database administrator determines the frequency of the checkpoints based on volume of transactions.
- When a system fails, It checks log to determine recovery action.
- Too frequent checkpoints can affect the performance.

2. Problems in this approach

- The search process is time-consuming.
- Most of the transactions that, according to our algorithm, need to be redone as they have already written their updates into the database.

3. Need of check points

- To record status of transaction execution, the system maintains the log, using one of the two techniques.
- The system periodically performs checkpoints, with following sequence of actions :
  1. Output all log records onto stable storage which are currently stored in main memory.
  2. Output to the disk.
  3. Output onto stable storage a log record <checkpoint>.
- Transactions are not allowed to perform any update actions, such as writing to a buffer block or writing a log record, while a checkpoint is in working state.
- The presence of a <checkpoint> record in the log allows the system to restructure its recovery procedure.

4. Working

- Consider a transaction T_n that committed prior to the checkpoint.
- For such a transaction, the <T_n commit> record appears in the log before the <checkpoint> record.
- Any database modifications made by transaction T_n must have been written to the database either prior to the checkpoint or as part of the checkpoint itself. Thus, at recovery time, there is no need to perform a redo operation on T_n.
- After a database failure has occurred, the recovery scheme examines the log to determine the most recent transaction T_n that started executing before the most recent checkpoint took place.
- It can find such a transaction by searching the log backward, from the end of the log, until it finds the first <checkpoint> record (since we are searching backward, the record found is the final <checkpoint> record in the log); then it continues the search backward until it finds the next <T_n start> record. This record identifies a transaction T_n.
- Once the system has identified respective transaction T_n, the redo and undo operations need to be applied to only for transaction T_n and all transactions that started executing after transaction T_n.
- The exact recovery operations to be performed depend on the modification technique being used.
- For the immediate-modification technique, the recovery operations are :

- For all transactions T_k in T that have no <T_k commit> record in the log, execute UNDO(T_k).
- For all transactions T_k in T such that the record <T_k commit> appears in the log, execute REDO(T_k).
- Obviously, the undo operation does not need to be applied when the deferred-modification technique is being employed.
- Consider the set of transactions {T_0, T_1 .... , T_{10}} executed in the order of the subscripts. Suppose that the recent checkpoint took place during the execution of transaction T_6. Thus, only transactions T_6, T_7, ... , T_{10} need to be considered during the recovery scheme. Each of them needs to be redone if it has committed; otherwise, it needs to be undone.

5. Advantages

1. A database checkpoint keeps track of change information and enables incremental database backup.
2. A database storage checkpoint can be mounted, allowing regular file system operations to be performed.
3. Database checkpoints can be used for application solutions which include backup, recovery or database modifications.

6. Disadvantages

1. Database storage checkpoints can only be used to restore from logical errors (E.g. a human error).
2. Because all the data blocks are on the same physical device, database storage checkpoints cannot be used to restore files due to a media failure.

Review Questions

1. What is recovery? Explain types of recovery?
2. What is log based recovery? Explain all log based recovery technique with example?
3. What is checkpoint? How is checkpoint information is used in recovery operation following System crash explain with example?
4. Describe shadow paging recovery technique? Under what circumstances does it not require a transaction log? List advantages and disadvantage of shadow paging?
5. What is database buffer? Explain buffer management technique used in database recovery?
8. Explain deferred database modification and immediate database modification and their difference in the context of recovery.

❑❑❑

Appendix A

Solved University Question Papers of May 2022 & Dec. 2022

May 2022

Q. 1(1) Which of the following is true in DBMS **(1 Mark)**

(A) Mechanism to copy data

(B) Mechanism to copy and managed data.

(C) Mechanism to store and manage data

(D) Mechanism to paste data.

Ans. : (C)

Q. 1(2) Weak Entity set **(1 Mark)**

(A) Do not have sufficient attributes

(B) Do not have any relation.

(C) Do not have sufficient attributes to form primary key

(D) Do not have attributes at all

Ans. : (C)

Q. 1(3) In E-R notations relationship reprinted as **(1 Mark)**

(A) eclipse

(B) diamond

(C) triangle

(D) circle

Ans. : (B)

Q. 1(4) Which of the following is correct syntax to display all employee from employee table using SQL

(A) Select & from employee

(B) Select * from employee

(C) Select * from department

(D) Select nothing **(1 Mark)**

Ans. : (B)

Q. 1(5) Which SQL command use to sort the data in ascending or descending order **(1 Mark)**

(A) Group by

(B) Order by

(C) having

(D) Not having

Ans. : (B)

Q. 1(6) Which of the following provides the ability to query information from the database and insert tuples into, delete tuples from, and modify tuples in the database? **(1 Mark)**

(A) DDL

(B) DCL

(C) DML

(D) Entity

Ans. : (C)

Q. 1(7) The Projection operation in relational algebra is written as **(1 Mark)**

(A) Σ

(B) P

(C) Π

(D) Σ

Ans. : (C)

Q. 1(8) Deletion of an student from table also deletes that student from another table. This kind of delete is called **(1 Mark)**

(A) cascaded

(B) virtual

(C) related

(D) Simple

Ans. : (A)

Q. 1(9) Which one of the following is not a part of ACID Properties of database transactions **(1 Mark)**

(A) Atomicity

(B) Consistency

(C) Isolation

(D) Deadlock

Ans. : (D)

Q. 1(10) A Transaction complete its execution is said to be

(A) Saved

(B) Loaded

(C) rolled

(D) committed **(1 Mark)**

Ans. : (D)

Q. 2(A) Explain the DDL and DML with suitable Examples. **(Refer sections 5.3 and 5.10)** **(10 Marks)**

Q. 2(B) Explain all E-R Notations with examples. **(Refer section 2.4)** **(10 Marks)**

Q. 2(C) Explain relational algebra operations in details. **(Refer sections 4.3 and 4.4)** **(10 Marks)**

Q. 3(A) Draw E-R diagram for Hospital Management System. **(10 Marks)**

Ans.:

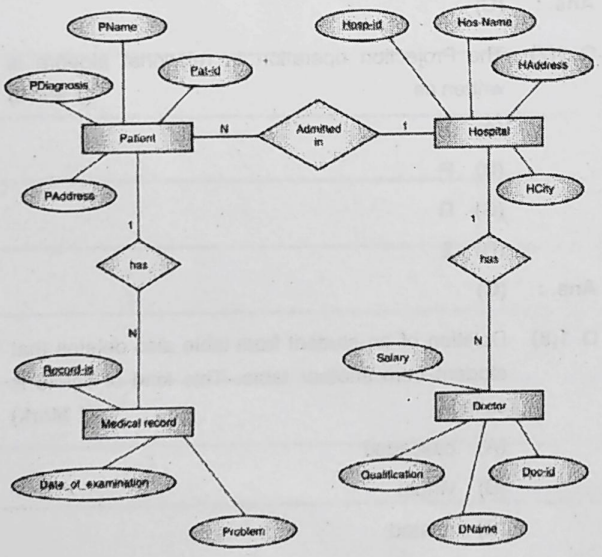

Fig.1-Q.3(a)

Q. 3(B) Explain deadlock with respect to database transactions. Also discuss deadlock handling. **(Refer section 13.9)** **(10 Marks)**

Q. 3(C) Explain database architecture in detail. **(Refer section 1.14)** **(10 Marks)**

Q. 4(A)(i) Compare traditional file system with database management system. **(Refer section 1.3)** **(5 Marks)**

Q. 4(A)(ii) Explain Log based recovery in transaction management. **(Refer section 14.4)** **(5 Marks)**

Q. 4(A)(iii) Explain the role of primary key and foreign key in DBMS. **(Refer section 2.4)** **(5 Marks)**

Q. 4(B)(i) Explain any five SQL aggregate functions with examples. **(Refer section 6.5)** **(10 Marks)**

Q. 4(B)(ii) Explain ACID Properties in Details. **(Refer section 12.2)** **(10 Marks)**

Dec. 2022

Q. 1(a) Write a short note on DDL Commands. **(Refer section 5.3)** **(5 Marks)**

Q. 1(b) Explain Characteristics of databases. **(Refer section 1.2)** **(5 Marks)**

Q. 1(c) Explain generalization and Specialization. **(Refer section 2.7.1)** **(5 Marks)**

Q. 1(d) Explain binary relational operations in relational algebra. **(Refer sections 4.6 and 4.7)** **(5 Marks)**

Q. 2(a) Draw and explain DBMS System architecture. **(Refer section 1.7)** **(10 Marks)**

Q. 2(b) Explain stored procedures and functions with example. **(Refer section 9.1)** **(10 Marks)**

Q. 3(a) Draw EER diagram for Library management System. **(Refer Ex. 1.1.1)** **(10 Marks)**

Q. 3(b) Explain join operations in relational algebra. **(Refer section 4.7)** **(10 Marks)**

Q. 4(a) Explain steps for Mapping the ER and EER Model to the Relational Model. **(Refer sections 3.8, 3.9 & 3.10)** **(10 Marks)**

Q. 4(b) Write SQL Syntax for **(10 Marks)**

Course Table

Cid	Course_Name	Staff_name	Duration (in weeks)	fees
1	DBMS	Menon	6	45000
2	PCPF	Rai	4	28000
3	JAVA	Rajput	2	16000
4	DSA	Govilkar	5	32000

Student Table

Sid	name	Location	Cid
1	Anaya	Thane	1
2	Rajiv	Navi_mumbai	4
3	Suyog	Dadar	2
4	Pari	Andheri	3
5	Dhariya	CST	1

(i) Create above course table also insert values.

(ii) Create student table with c_id as foreign key.

(iii) Arrange courses in descending order of fees.

(iv) Find name of course student name 'Rajiv 'has enrolled for.

(v) List down names of all students whose course duration is more than 3.

Ans. :

(i) Create above course table also insert values

```
Create tablecourse
(
Cid             varchar(10)Primary Key,
Course_Name     varchar(20),
Staff_name      varchar(30),
Duration        int,
Fees            number(10,2)
);
```

(ii) Create student table with c_id as foreign key

```
Create tableStudent
(
Sid             varchar(10 Primary Key,
Name            varchar(20),
Staff_name      varchar(30),
Location        varchar(50),
Cid             varchar(10) References course(Cid),
);
```

(iii) Arrange courses in descending order of fees

```
Select *
From course
Order by Fees DESC;
```

(iv) Find name of course student name 'Rajiv 'has enrolled for

```
Select      Course_Name
From        student s, course c
Where       s.cid=c.cid
AND         Name = "Rajiv";
```

(v) List down names of all students whose course duration is more than 3

```
Select      Name
From        student s, course c
Where       s.cid=c.cid
AND         c.Duration > 3;
```

Q. 5(a) Define normalization. Explain 1NF,2NF and 3NF with example.
(Refer sections 11.8, 11.9, 11.10 and 11.11) **(10 Marks)**

Q. 5(b) Explain Serializability with types. **(Refer Section 12.7)** **(10 Marks)**

Q. 6 Write short note on (Any four) **(20 Marks)**

(a) Role of DBA. **(Refer section 1.17.1)**

(b) Need of Normalization.(Refer Section 11.8)

(c) Primary key and Foreign key. **(Refer section 2.4)**

(d) ACID properties. **(Refer section 12.2)**

(e) Nested and Sub queries in SQL. **(Refer section 7.5)**

❑❑❑