**Q.2** a. Define the following terms giving examples for each of them:
Entity, attribute, role and relationship between the entities

**Answer:**

**Ans2a. Entity:** An entity is a "thing" or "object" in the real world that is distinguishable from other objects. For example, a person and bank account can be considered as entities.

**Attribute:** Entities are described in a database by a set of attributes, i.e., the characteristics of an entity are known as attributes. For example, name, age, date of birth, etc are attributes of the entity person. Similarly, account number, balance, nature of account, etc are attributes of the entity bank account.

**Relationship:** A relationship is an association among the several entities. For example, a depositor relationship associates the entity person with a bank account.

**Role:** The function that an entity plays in a relationship is called that entity's role. For example, in the relationship depositor mentioned above, the entity person plays the role of a customer in the relationship.

b. Describe any four main functions of a database administrator.

**Answer:**

**Ans2b.** A **database administrator** (**DBA**) is a person who is responsible for the environmental aspects of a database. In general, these include **(any four of these):**

- **Recoverability** - Creating and testing Backups
- **Integrity** - Verifying or helping to verify data integrity
- **Security** - Defining and/or implementing access controls to the data
- ***Availability** - Ensuring maximum uptime
- **Performance** - Ensuring maximum performance
- **Development and testing support** - Helping programmers and engineers to efficiently utilize the database

c. What is a weak entity set? What are two principles sources of weak entity sets? Give examples to explain.

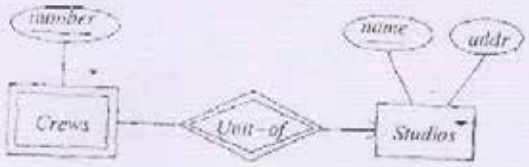
**Answer:**

**Ans2c.** There is an occasional condition in which an entity set's key is composed of attributes some or all, of which belong to another entity set. Such an entity set is called a weak entity set.

**Causes of Weak Entity Sets**

There are two principal sources of weak entity sets. First, sometimes entity sets fall into a hierarchy based on classifications unrelated to the "isa hierarchy". If entities of set E are subunits of entities in set F, then it is possible that the names of E entities are not unique until we take into account the name of the F entity to which the E entity is subordinate. For example

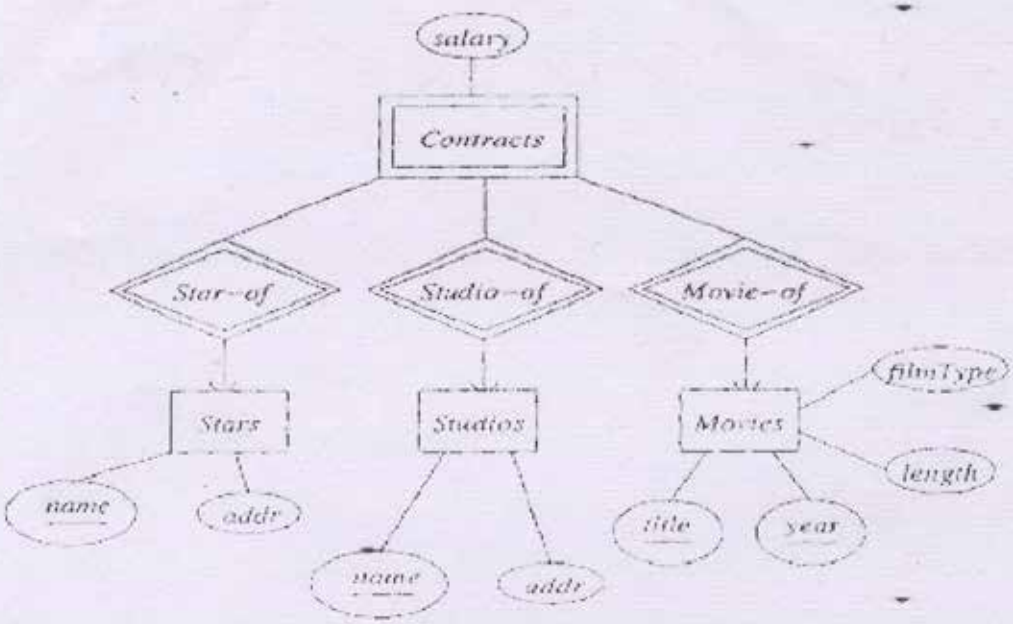
A movie studio might have several film crews. The crews might be designated by a given studio as crew 1, crew 2, and so on. However, other studios might use the same designations for crews, so the attribute number is not a key for crews. Rather, to name a crew uniquely, we need to give both the name of the studio to which it belongs and the number of the crew. The situation is suggested as shown in the figure The key for weak entity set Crews is its own ,lumber attribute and the name attribute of the unique studio to which the crew is related by the many-one Unit-of relations




**(any such example can be given)**

The second source of weak entity sets is the connecting entity sets. These entity sets often have no attributes of their own. Their key is formed from the attributes that are the key attributes for the entity sets they connect.

In the following figure we see a connecting entity set. 'Contracts' has an attribute salary, but this attribute does not contribute to the key. Rather, the key for a contract consists of the name of the studio and the star involved, plus the title and year of the movie involved.



**Q.3** a. Consider the following relation

Movie(title, year, length, incolor, studioName, producerC#)
StarsIn(movieTitle, movieyear, starName)
MovieStar(name, address, gender, birthdate)
MovieExec(name, address, cert#, networth)
Studio(name, address, presC#)

Write SQL quesries for

(i) The titles of movies made by 'MGM' Studios that either were made after 1970 or were less than 90 minutes long.

(ii) The names and addresses of all female movie stars that are also movie executives with a net worth over $10,000,000.

(iii) Finding the producer of 'Star Wars' by using a nested subquery.

(iv) Finding the producers of Harrison Ford's movies.

**Answer:**

**Ans3a.**

Movie(title, year, length, incolor, studioName, producerC#)
StarsIn(movieTitle, movieyear, starName)
MovieStar(name, address, gender, birthdate)
MovieExec(name, address, cert#, networth)
Stndio(name, address, presC#)
SQL quesries for

(i) the titles of movies made by 'MGM' Studios that either were made after 1970 or were less than 90 minutes long.

```
SELECT title
FROM Movie
WHERE (year > 1970 OR length < 90) AND studioName = 'MGM';
```

(2 ½

(ii) the names and addresses of all female movie stars that are also movie executives with a net worth over $10,000,000.

```
(SELECT name, address
 FROM MovieStar
 WHERE gender = 'F')
     INTERSECT
(SELECT name, address
 FROM MovieExec
 WHERE netWorth > 10000000);
```

(2½

(iii) Finding the producer of Star Wars by using a nested subquery.

b. Explain how Relational Calculus is different from Relational Algebra.

**Answer:**

**Ans3b. Relational Algebra and Relational calculus( RA and RC)** major differences are that RC is non procedural while RA is procedural and rigid as such. However, another notable difference is that it is easy to implement RC expressions directly to SQL language **(Use a suitable example to explain)**

**Q.4** a. SQL allows attributes to have a special value NULL, which is called the null value. What are three common interpretations that can be put on null values?

**Answer:**

**Ans4a.**

1. Value unknown: that is, 'I know there is some value that belongs here but 1 don't know what it is."
2. Value inapplicable: "There is no value that makes sense here."
3. Value withheld: "We are not entitled to know the value that belongs here."

b. Define and explain the following constraints:
Key Constraints, Referential Integrity Constraints, Attribute-Based Check Constraints and Tuple-Based Check Constraints
How does SQL allow specification of these?

**Answer:**

**Ans4b. (Elaborate following)**

**Key Constraints:** We can declare an attribute or set of attributes to be a key with a UNIQUE or PRIMARY KEY declaration in a relation schema.

**Referential Integrity Constraints:** We can declare that a value appearing in some attribute or set of attributes must also appear in the corresponding attributes of some tuple of another relation with a REFERENCES or FOREIGN KEY declaration in a relation schema.

**Attribute-Based Check Constraints:** We can place a constraint on the value of an attribute by adding the keyword CHECK and the condition to be checked after the declaration of that attribute in its relation schema.

**Tuple-Based Check Constraints:** We can place a constraint on the tuples of a relation by adding the keyword CHECK and the condition to be checked to the declaration of the relation itself.

**Q.5** a. What are Multi-valued Dependencies? When we can say that a constraint X is multi-determined? Does the following relation satisfy MVDs with 4NF? Decompose the following relation into 5NF.

| SUPPLY SNAME | PARTNAME | PROJNAME |
|---|---|---|
| Dev | Bolt | X |
| Dev | Nut | Y |
| Sanju | Bolt | Y |
| Sainyam | Nut | Z |
| Sanju | Nail | X |
| Sanju | Bolt | X |
| Dev | Bolt | Y |

**Answer:**

**Ans5a.**

An MVD is a constraint due to multi-valued attributes. A relational must have at least 3 attributes out of which two should be Multi-valued.

No.

R1

| SNAME | PARTNAME |
|---|---|
| Dev | Bolt |
| Dev | Nut |
| Sanju | Bolt |
| Sainyam | Nut |
| Sanju | Nail |

R2

| SNAME | PROJNAME |
|---|---|
| Dev | X |
| Dev | Y |
| Sanju | Y |
| Sainyam | Z |
| Sanju | X |

R3

| PARTNAME | PROJNAME |
|---|---|
| Bolt | X |
| Nut | Y |
| Bolt | Y |
| Nut | Z |
| Nail | X |

b. Describe four desirable properties of relational decompositions using suitable examples.

**Answer:**

**Ans5b.** Describe following four desirable properties of decomposition:

- Attribute preservation
- Lossless-join decomposition
- Dependency preservation
- Lack of redundancy

**Q.6** a. An ordered student file (ordering field is enrolment number) has 20,000 records stored on a disk having the Block size as 1 K. Assume that each student record is of 100 bytes, the ordering field is of 8 bytes, and block pointer is also of 8 bytes, find how many block accesses on average may be saved on using primary index.

**Answer:**

Number of records in the file = 20000
Block size = 1024 bytes
Record size = 100 bytes
Number of records per block = integer value of [1024 / 100] = 10
Number of disk blocks acquired by the file = [Number of records / records per block]
= [20000/10] = 2000
Assuming a block level binary search, it would require $\log_2 2000$ = about11 block accesses.

**Number of accesses with Primary Index:**

Size of an index entry = 8+8 = 16 bytes
Number of index entries that can be stored per block
= integer value of [1024 / 16] = 64
Number of index entries = number of disk blocks = 2000
Number of index blocks = ceiling of [2000/ 64] = 32
Number of index block transfers to find the value in index blocks = $\log_2 32 = 5$
One block transfer will be required to get the data records using the index pointer after the required index value has been located. So total number of block transfers with primary index = 5 + 1 = 6.
**Thus, the Primary index would save about 5 block transfers for the given case.**

b. What is the order of a B-tree? Describe the structure of B-tree nodes. Why is a B+ tree a better structure than a B-tree for implementation of an indexed sequential file?

**Answer:** Page Number 527 of Text Book

**Q.7** a. Consider the following information to elaborate Nested-Loop Join algorithm.
MARKS (enroll no, subject code, marks):10000 rows, 500 blocks
STUDENT (enroll no, name, dob): 2000 rows, 100 blocks
Also discuss the cost for the worst case.

**Answer:**

**Ans7a.** To compute the theta or equi-join

```
for each tuple s in STUDENT
{
for each tuple m in MARKS
{
        test s.enrolno = m.enrolno to see if they satisfy the join
condition if they do, output joined tuple to the result. }; };
```

- In the nested loop join there is one outer relation and one inner relation.
- It does not use or require indices. It can be used with any kind of join condition. However, it is expensive as it examines every pair of tuples in the two relations.
- If there is only enough memory to hold one block of each relation, the number of disk accesses can be calculated as:

For each tuple of STUDENT, all the MARKS tuples (blocks) that need to be accessed. However, if both or one of the relations fit entirely in the memory, block transfer will be needed only once, so the total number of transfers in such a case, may be calculated as:

= Number of blocks of STUDENT + Number of blocks of MARKS
= 100 + 500 = 600.

If only the smaller of the two relations fits entirely in the memory then use that as the inner relation and the bound still holds.

**Cost for the worst case:**

Number of tuples of outer relation × Number of blocks of inner relation + Number of blocks of outer relation.

2000 * 500 + 100 = 1,000,100 with STUDENT as outer relation.

There is one more possible bad case when MARKS is on outer loop and STUDENT in the inner loop. In this case, the number of Block transfer will be:

10000 * 100+500 = 1,000,500 with MARKS as the outer relation.

b. What are the various steps involved in the generation of query-evaluation plans for an expression, elaborate using suitable examples.

**Answer:**

**Ans7b.** Generation of query-evaluation plans for an expression involves several steps:

1) Generating logically equivalent expressions using **equivalence rules**
2) Annotating resultant expressions to get alternative query plans
3) Choosing the cheapest plan based on **estimated cost.**

The overall process is called **cost based optimization. (Explain)**

**Q.8** a. Describe using suitable examples, problems of concurrent transactions. Can these problems occur in transactions which do not read the same data values?

**Answer:**

- Lost updates: An update is overwritten
- Unrepeatable read: On reading a value later again an inconsistent value is found.
- Dirty read: Reading an uncommitted value
- Inconsistent analysis: Due to reading partially updated value.

b. Briefly describe Optimistic Concurrency Control.

**Answer:**

**Ans8b.** The basic philosophy for optimistic concurrency control is the optimism that nothing will go wrong so let the transaction interleave in any fashion, but to avoid any concurrency related problem we just validate our assumption before we make changes permanent. This is a good model for situations having a low rate of transactions 3 marks

c. Consider the following two transactions, given two bank accounts having a balance A and B.
Transaction T1: Transfer Rs. 100 from A to B
Transaction T2: Find the multiple of A and B
Add lock and unlock instructions (exclusive or shared) to transactions T1 and T2 so that they observe the serialisable schedule. Make a valid schedule.

**Answer:**

**Ans8c.**

| Schedule | T1 | T2 |
|---|---|---|
| Lock A | Lock A | |
| Lock B | Lock B | |
| Read A | Read A | |
| A = A - 100 | A = A - 100 | |
| Write A | Write A | |
| Unlock A | Unlock A | |
| Lock A | | Lock A: Granted |
| Lock B | | Lock B: Waits |
| Read B | Read B | |
| B = B + 100 | B = B + 100 | |
| Write B | Write B | |
| Unlock B | Unlock B | |
| Read A | | Read A |
| Read B | | Read B |
| Result = A * B | | Result = A * B |
| Display Result | | Display Result |
| Unlock A | | Unlock A |
| Unlock B | | Unlock B |

**Q.9** a. Compare shadow paging with log based recovery methods. Write the advantages and disadvantages of shadow paging over log-based schemes.

**Answer:**

**Ans9a. Advantages of shadow-paging over log-based schemes:**

- It has no overhead of writing log records,
- The recovery is trivial.

**Disadvantages:**

- Copying the entire page table is very expensive, it can be reduced by using a page table structured like a B+-tree (no need to copy entire tree, only need to copy paths in the tree that lead to updated leaf nodes).
- Commit overhead is high even with the above extension (Need to flush every updated page, and page table).
- Data gets fragmented (related pages get separated on disk).
- After every transaction is completed, the database pages containing old versions is completed, of modified data need to be garbage collected/freed.
- Hard to extend algorithm to allow transactions to run concurrently (easier to extend log based schemes).

b. Discuss deferred update technique of recovery. What are the advantages and disadvantages of this technique over immediate update technique?

**Answer:**

**Ans9b. Deferred Update**

Deferred update also called NO-UNDO/REDO is a technique used to recover/support transaction failures that occur due to operating system, power, memory or machine failures. When a transaction runs, any updates or alterations made to the database by the transaction are not done immediately. They are recorded in the log file. Data changes recorded in the log file are applied to the database on commit. This process is called "Re-doing". On rollback, any changes to data recorded in the log file are discarded; hence no changes will be applied to the database. If a transaction fails and it is not committed due to any of the reasons mentioned above, the records in the log file are discarded and the transaction is restarted. If the changes in a transaction are committed before crashing, then after the system restarts, changes recorded in the log file are applied to the database.

Advantages:

1. Recovery is made easier:
   Any transaction that reached the commit point (from the log) has its writes applied to the database (REDO).
   All other transactions are ignored.
2. Cascading rollback does not occur because no other transaction sees the work of another until it is committed (no stale reads).

Disadvantages:

1. Concurrency is limited: Must employ Strict 2PL which limits concurrency

c. Define check point and give its impact on data base recovery.

**Answer:**

**Ans9c. Checkpoint:** A recovery point in the logs where all current transactions have terminated and all updates have been written to disk. Consists of 4 steps:

1. Cease accepting new transactions
2. Allow all unfinished transactions to complete (commit or abort)
3. Write all pending changes to disk and to logs
4. Resume accepting new transactions

- In many environments, it is possible to take checkpoints each 15 minutes or half hour, etc.
- Recovery must then only be done from the time of the last checkpoint (assuming no damage to media).

## TEXT BOOK

Fundamental of Database Systems, Elmasri, Somayajulu, Gupta, Pearson Education, 2006