June 2023 (2019 scheme)

1. Classify the following cases into logical data independence and physical data independence.

(a) Creating an index for a data file

-physical data independence.

(b) Changing the integrity constraint

- logical data independence

(c) Reorganizing the file

- physical data independence**.**

2. What is meant by a recursive relationship type? Give an example of recursive relationship type.

In some cases the *same entity type participates* more than once in a relationship type in *different roles. In such cases the role name* becomes essential for distinguishing the meaning of the role that each participating entity plays. Such relationship types are called recursive relationships.

The SUPERVISION relationship type relates an employee to a supervisor, where both employee and supervisor entities are members of the same EMPLOYEE entity set. Hence, the EMPLOYEE entity type *participates twice in* SUPERVISION: once in the role of *supervisor (or boss), and once in the role of supervisee (or subordinate).*

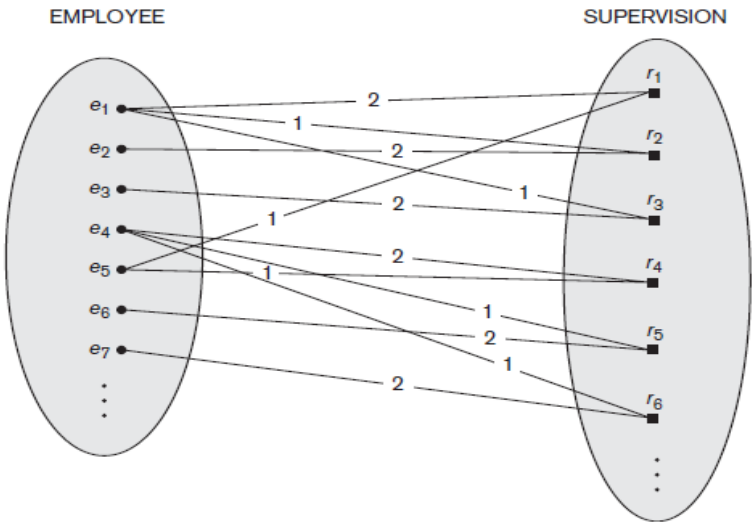


Figure 7.11
A recursive relationship SUPERVISION between EMPLOYEE in the *supervisor* role (1) and EMPLOYEE in the *subordinate* role (2).

3. Consider the relational model constraints: domain constraint, key constraint, entity integrity and referential integrity. Specify which of these constraints may be violated during the following modification operations: insert, update and delete.

Domain constraint :

Domain constraint gets violated only when a given value to the attribute does not appear in the corresponding domain or in case it is not of the appropriate datatype.

Entity Integrity constraint :

On inserting NULL values to any part of the primary key of a new tuple in the relation can cause violation of the Entity integrity constraint.

Key Constraints :

On inserting a value in the new tuple of a relation which is already existing in another tuple of the same relation, can cause violation of Key Constraints

Referential integrity :

On inserting a value in the foreign key of relation 1, for which there is no corresponding value in the Primary key which is referred to in relation 2, in such case Referential integrity is violated

Deletion operation:

On deleting the tuples in the relation, it may cause only violation of Referential integrity constraints.

Referential Integrity Constraints :

It causes violation only if the tuple in relation 1 is deleted which is referenced by foreign key from other tuples of table 2 in the database, if such deletion takes place then the values in the tuple of the foreign key in table 2 will become empty, which will eventually violate Referential Integrity constraint

4. What is meant by complete set of relational algebra? Show how join operation in relational algebra can be represented using this set.

In relational algebra, a complete set of operators refers to a set of fundamental operations that can be used to express any relational algebra expression. The complete set typically consists of five operations: selection, projection, Cartesian product, set difference, and union.

To represent the join operation using the complete set perform the Cartesian product (×) of the two relations .Apply a selection (σ) on the resulting relation, specifying the join condition.

EMPLOYEE

Employee ID	Name	Dept.ID
E1	Alice	D1
E2	Bob	D2
E3	Charlie	D1

DEPARTMENT

Department ID	Department Name
D1	Marketing
D2	Sales

EMPLOYEE X DEPARTMENT

Employee ID	Name	Dept.ID	Department ID	Department Name
E1	Alice	D1	D1	Marketing
E1	Alice	D1	D2	Sales
E2	Bob	D2	D1	Marketing

E2	Bob	D2	D2	Sales
E3	Charlie	D1	D1	Marketing
E3	Charlie	D1	D2	Sales

$\sigma_{\text{Dept.ID = Department ID}}$ (EMPLOYEE X DEPARTMENT)

Employee ID	Name	Dept.ID	Department ID	Department Name
E1	Alice	D1	D1	Marketing
E2	Bob	D2	D2	Sales
E3	Charlie	D1	D1	Marketing

5. What is meant by a correlated nested query? Give a suitable example.

Some queries require that existing values in the database be fetched and then used in a comparison condition. Such queries can be formulated by using nested queries, which are complete SELECT . . . FROM . . . WHERE . . . blocks within the WHERE-clause of another query. Whenever a condition in the WHERE-clause of a nested query references some attribute of a relation declared in the outer query, the two queries are said to be correlated

Retrieve the name of each employee who has a dependent with the same first name and same sex as the employee

```
Q16: SELECT  E.FNAME, E.LNAME
     FROM    EMPLOYEE AS E
     WHERE   E.SSN IN    (SELECT  ESSN
                          FROM    DEPENDENT
                          WHERE   E.FNAME=
                                  DEPENDENT_NAME AND
                                  E.SEX=SEX);
```

6. Explain the advantage of a multilevel index.

Multilevel indexing improve the efficiency of searching an index file in following way:

1) A multilevel index reduces the number of blocks accessed when searching for a record, given its indexing field value.

2) To retain the benefits of using multilevel indexing while reducing index insertion and deletion problems,

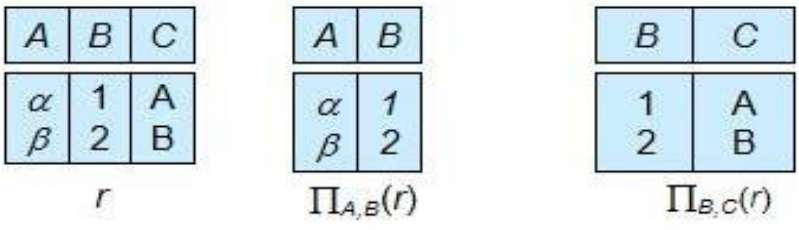
3) Designers adopted a multilevel index that leaves some space in each of its blocks for inserting new entries. This is called a dynamic multilevel index and is often implemented by using data structures called B-trees and B+-trees

7. Why Armstrong's axioms are said to sound and complete?

The axioms are sound in generating only functional dependencies in the closure of a set of functional dependencies when applied to that set .They are also complete in that repeated application of these rules will generate all functional dependencies in the closure.

8. What is meant by lossless join property?

The word loss in lossless refers to loss of information, not to loss of tuples. If a decomposition does not have the lossless join property, we may get additional spurious tuples after the PROJECT (π) and NATURAL JOIN (*) operations are applied; these additional tuples represent erroneous or invalid information. The Lossless join(nonadditive join) property ensures that no spurious tuples result after the application of PROJECT and JOIN operations.



$\Pi_A(r) \bowtie \Pi_B(r)$

A	B	C
α	1	A
β	2	B

9. List and explain the desirable properties of a transaction.

Atomicity: A transaction is an atomic unit of processing; it should either be performed in its entirety or not performed at all.

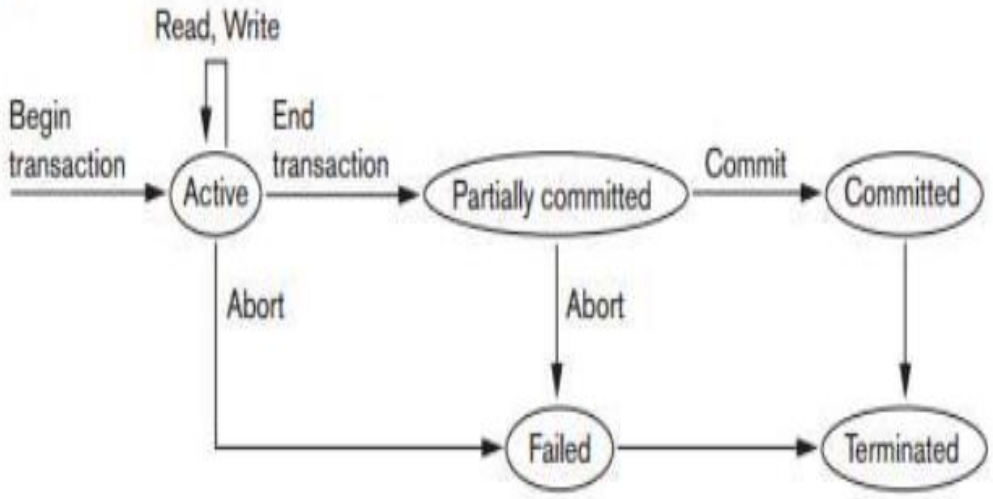
Consistency preservation: A transaction should be consistency preserving, meaning that if it is completely executed from beginning to end without interference from other transactions, it should take the database from one consistent state to another.

Isolation: A transaction should appear as though it is being executed in isolation from other transactions, even though many transactions are executing concurrently. That is, the execution of a transaction should not be interfered with by any other transactions executing concurrently.

Durability or permanency: The changes applied to the database by a committed transaction must persist in the database. These changes must not be lost because of any failure.

10. Illustrate the states for transaction execution.

Figure 21.4
State transition diagram illustrating the states for transaction execution.



1. BEGIN_TRANSACTION → This marks the beginning of transaction execution.

2. READ or WRITE → These specify read or write operations on the database items that are executed as part of a transaction.

3. END_TRANSACTION → This specifies that READ and WRITE transaction operations have ended and marks the end of transaction execution.

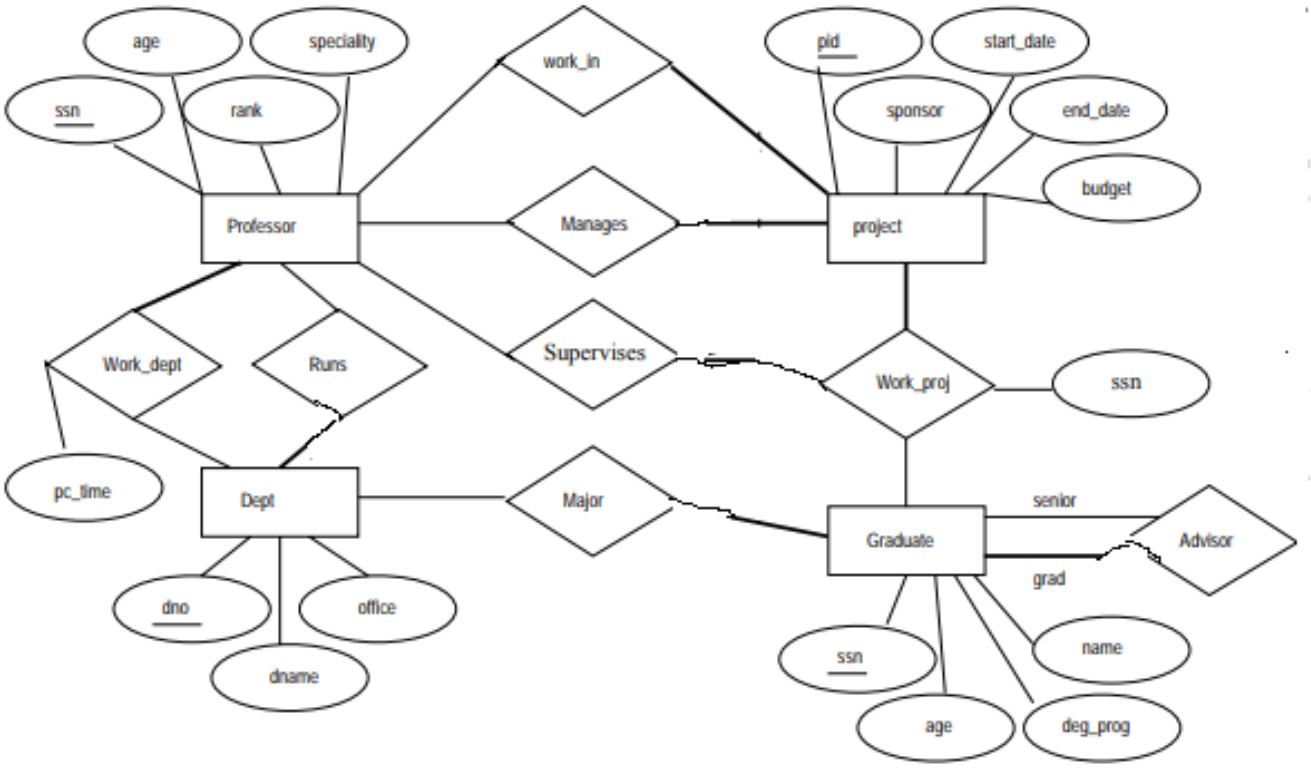
4. COMMIT_TRANSACTION → This signals a successful end of the transaction so that any changes (updates) executed by the transaction can be safely committed to the database and will not be undone.

5. ROLLBACK (or ABORT) → This signals that the transaction has ended unsuccessfully, so that any changes or effects that the transaction may have applied to the database must be undone.

PART B

11. Consider the following information about a university database:

Professors have a ssn, a name,an age, a rank, and a research specialty. Projects have a project number, a sponsor name, a starting date, an ending date, and a budget. Graduate students have ssn, a name, an age, and a degree program (e.g., M.S. or Ph.D.). Each project is managed by one professor (known as the project's principal investigator). Each project is worked on by one or more professors (known as the project's co-investigators). Professors can manage and/or work on multiple projects. Each project is worked on by one or more graduate students (known as the project's research assistants). When graduate students work on a project, a professor must supervise their work on the project. Departments have a department number, a department name, and a location.Departments have a professor (known as the chairman) who manages the department.Professors work in one or more departments, and for each department that they work percentage is associated with their job. Graduate students have one major department in which they are working on their degree. Each graduate student has another, more senior graduate student (known as a student advisor) who advises him or her on what courses to take. Design and draw an ER diagram that captures the information about the university.

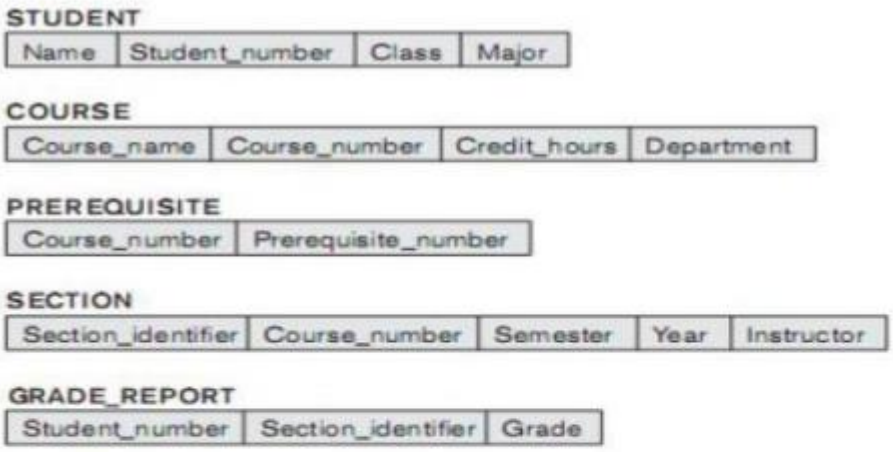


12 a) Explain the difference between database schema and database state with suitable example. Specify the role of schema in a DBMS.

The description of a database is called the database schema, which is specified during database design and is not expected to change frequently .A displayed schema is called a schema diagram. We call each object in the schema a schema construct. In a given database state, each schema construct has its own current set of instances

for example, the STUDENT construct will contain the set of individual student entities (records) as its instances.

The data in database at particular instant or moment of time is called database state or snapshot The schema is not supposed to change frequently, but it is not uncommon that changes occasionally need to be applied to the schema as the application requirements change. It is called schema evolution.



Database schema

Name	Student_number	Class	Major
Ram	CS001	R4	CSE
Shyam	CS002	R4	CSE

Database state

b) With a neat diagram explain the Three Schema Architecture of a DBMS

Internal level

The internal level has an internal schema, which describes the physical storage structure of the database.The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.

Conceptual level

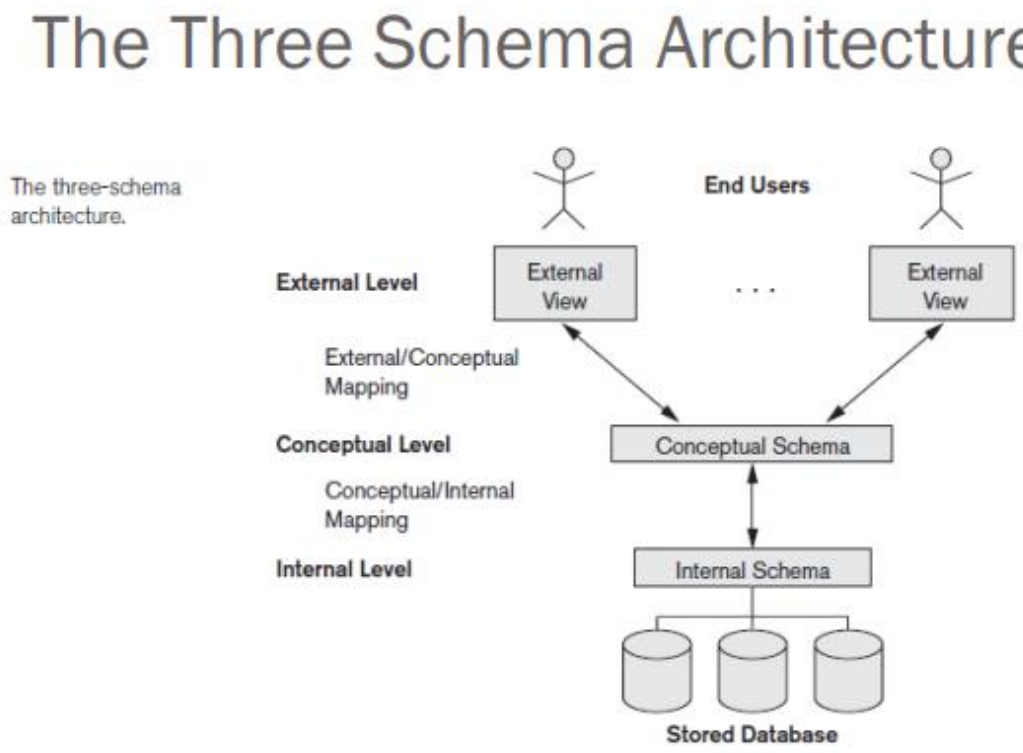
Describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints. The implementation conceptual schema is often based on a conceptual schema design in a high-level data model.

External or view level

The external or view level includes a number of external schemas or user views. Each external schema describes the part of the database that a particular user group is interested in

and hides the rest of the database from that user group. Each external schema is typically implemented using a representational data model, possibly based on an external schema design in a high-level data model.

In a DBMS based on the three-schema architecture, each user group refers only to its own external schema. Hence, the DBMS must transform a request specified on an external schema into a request against the conceptual schema, and then into a request on the internal schema for processing over the stored database. If the request is a database retrieval, the data extracted from the stored database must be reformatted to match the user's external view.



13 a) Consider the two relations T1 and T2 shown below. Show the results of the following operations.

Relation **T1**

P	Q	R
30	Ac	25
35	Bc	28
45	Ac	26

Relation **T2**

A	B	C
30	Bc	26
45	Cc	23
30	Bc	25

i) $T1 \bowtie_{T1.Q=T2.B} T2$

ii) $T1 \bowtie_{T1.P=T2.A} T2$

iii) $T1 \cup T2$

iv) $T1 \bowtie_{(T1.P=T2.A \text{ AND } T1.R=T2.C)} T2$

$T1 \bowtie_{T1.Q=T2.B} T2$

P	Q	R	A	B	C
35	Bc	28	30	Bc	26
35	Bc	28	30	Bc	25

b) An Employee relation has attributes: Employee-Id (numeric type), Name (character type),Salary (numeric type) and Dep-No (numeric type).A Department relation has attributes: Department-Number (numeric type), DepartmentName(character type), Dep-Manager-Id (numeric type).Employee-Id is the primary key of Employee relation. Department-Number is the primary keyof the Department relation. Dep-No attribute of Employee relation refers to the DepartmentNumber attribute of Department relation and Dep-Manager-Id attribute of Department relation refers to the Employee-Id attribute of Employee relation.

(i) Write create table statements by specifying necessary integrity constraints for creating these two relations in SQL.

```
CREATE TABLE employee
        ( employee_id  INTEGER PRIMARY KEY,
```

```
    name VARCHAR(10),
    salary INTEGER,
  department_no INTEGER);

CREATE TABLE department
  (department_no INTEGER PRIMARY KEY,
    department_ name VARCHAR(10),
   department_manager_id  INTEGER);
```

ALTER TABLE employee ADD FOREIGN KEY (department_no)
REFERENCES department (department_no)

ALTER TABLE department ADD FOREIGN KEY (department_manager_id)
REFERENCES employee (employee_id)

(ii) Write SQL statement to insert the details of an employee John with id 101 with salary 5000 and working in department number 5.

INSERT INTO employee VALUES(101,'John',5000,5);

(iii) Insert the details of a Research Department with Department Number 1 and it has not been assigned any manager.

INSERT INTO department VALUES(1,' Research',NULL);

(iv) Assume that a department with employees working in it is to be deleted. Specify the two options to manage this scenario.

Option1:

DELETE FROM employee WHERE department_no=2;

DELETE FROM department WHERE department_no=2;

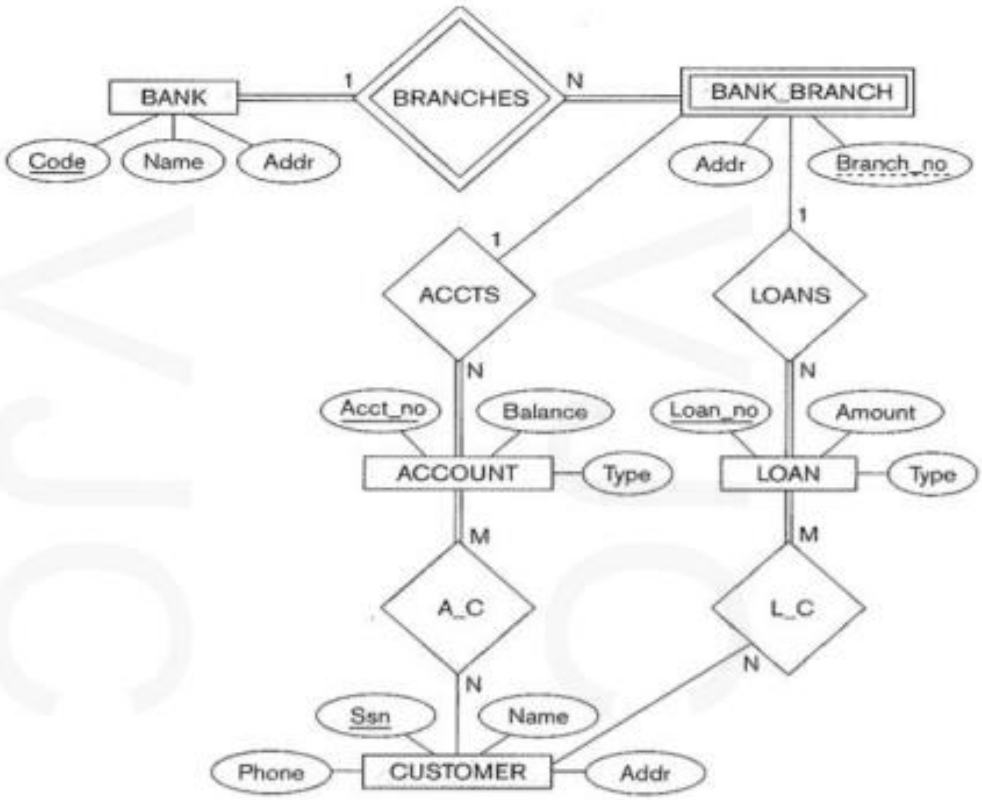
Option 2:

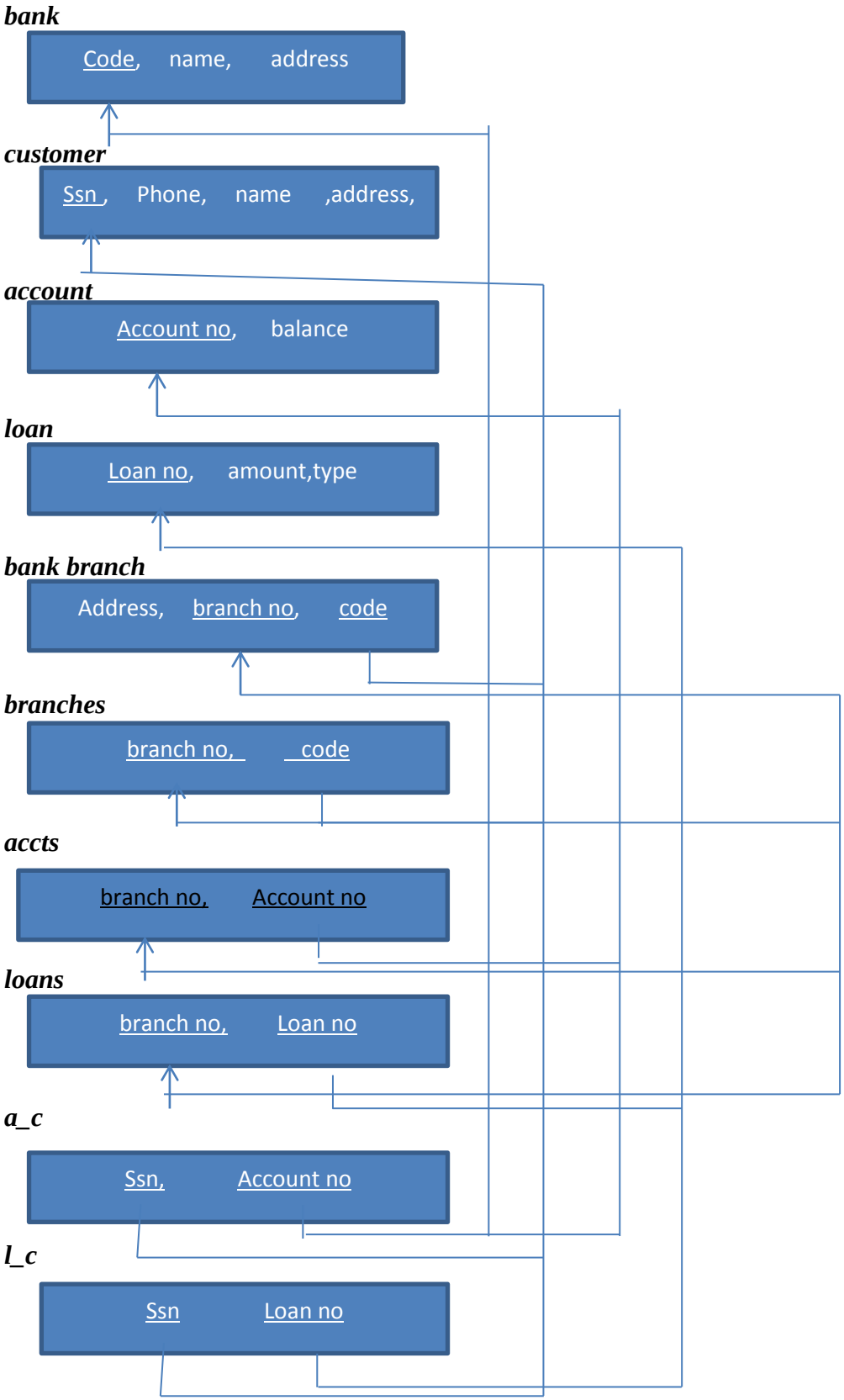
ALTER TABLE employee ADD FOREIGN KEY (department_no)
REFERENCES department (department_no) ON DELETE CASCADE;

DELETE FROM department WHERE department_no=2;

As you delete the contents of department_no=2 in the department table it automatically deletes the details of department_no=2 from the employee table also. This works out because the foreign key constraint ON DELETE CASCADE is specified

14. Convert the ER schema for Bank database given below into a relational schema.




bank
Code, name, address
customer
Ssn , Phone, name ,address,
account
Account no, balance
loan
Loan no, amount,type
bank branch
Address, branch no, code
branches
branch no, code
accts
branch no, Account no
loans
branch no, Loan no
a_c
Ssn, Account no
l_c
Ssn Loan no

15. a) Consider the following relations:

Employee (Employee-Id, Employee-Name, Salary, Department-No)

Department (Department-No, Department-Name)

Write SQL queries for the following:

(i) Retrieve the employee names and their department names

```
SELECT E.Employee_Name, D.Department_Name
FROM Employee E,Department D
WHERE E.Department_No = D.Department_No;
```

(ii) Retrieve department names and the average salary given by them

```
SELECT D.Department_Name, AVG(E.Salary)
FROM Employee E,Department D
WHERE E.Department_No = D.Department_No
GROUP BY D.Department_Name;
```

(iii) Retrieve the ids of employees getting salary greater than the average salary of their department

```
SELECT E.Employee_Id
FROM Employee E,Department D
WHERE E.Department_No = D.Department_No
AND E.Salary >
(SELECT AVG(Salary) FROM Employee
 WHERE E.Department_No = D.Department_No
GROUP BY D.Department_Name);
```

(iv) For each department that has more than 4 employees, retrieve the department-No and the number of employees getting salary more than Rs. 50000

```
SELECT D.Department_No, COUNT(*)
```

```
FROM Employee E,Department D

WHERE E.Department_No = D.Department_No
AND  E.Salary > 50000
GROUP BY D.Department_Name
HAVING COUNT(*) > 4;
```

b) What is meant by a heap file? Explain how insert, update, delete and search operations can be performed in a heap file.

A heap file is a type of file organization used to store records or tuples in an unordered manner. It is called a "heap" because the records are placed randomly throughout the file, similar to a heap of objects without any specific order or structure.The key characteristic of a heap file is that it allows for efficient insertion of new records at the end of the file. Since there is no predetermined order, new records can be easily appended to the file without the need for rearranging or reshuffling the existing data.

Insertion:

To insert a new record in a heap file, the new record is simply appended at the end of the file.The file's metadata, such as the total number of records or the file's size, is typically updated to reflect the addition of the new record.No special considerations are required for maintaining any particular order or index structure.

Update:

Updating a record in a heap file involves searching for the specific record to be updated.A sequential scan of the entire file is performed until the desired record is found.Once the record is located, the necessary modifications or updates are made directly to the record in place.

Deletion:

Deleting a record from a heap file also requires a search operation to locate the specific record to be deleted.Similar to the update operation, a sequential scan of the entire file is performed until the record is found.Once the record is located, it can be physically removed

from the file by either marking it as deleted or shifting the subsequent records to fill the gap left by the deletion.

Search:

Searching for a specific record in a heap file requires a sequential scan of the entire file.Each record is examined one by one until the desired record is found or until the end of the file is reached.Since there is no predefined order or indexing structure in a heap file, sequential scanning is the only method to search for a record efficiently.

16 a) What are the advantages of Views? Explain two view implementation techniques.

Views in a database management system provide several advantages, including:

Data Abstraction: Views allow for data abstraction by presenting a customized or simplified view of the data to the users. Instead of accessing the underlying tables directly, users can interact with the view, which can hide complexity, enforce security restrictions, and provide a tailored perspective of the data.

Security and Access Control: Views can be used to enforce security policies and access controls on the underlying tables. By granting permissions to access specific views instead of the underlying tables, database administrators can ensure that users only have access to the data they need, while restricting access to sensitive information.

Simplified Data Manipulation: Views can simplify data manipulation operations by providing predefined queries, aggregations, joins, or complex calculations. Users can perform operations on the view without needing to understand the underlying table structures or writing complex queries, thus enhancing productivity and reducing the risk of errors.

Data Integrity and Consistency: Views can be used to enforce data integrity and consistency rules. By defining views that include specific conditions or constraints, database administrators can ensure that only valid data is visible through the view, preventing data inconsistencies and maintaining the overall integrity of the database.

Two common techniques for implementing views are:

Materialized Views:

Materialized views store the results of a view query as a physical table in the database.The view is refreshed or updated periodically or when the underlying data changes, ensuring that the materialized view always reflects the most recent data.

Virtual or Query-Defined Views:

Virtual views, also known as query-defined views or simply views, do not store the results of a query as a physical table.Instead, the view is defined by a query, which is executed dynamically whenever the view is accessed.

Virtual views provide a logical representation of the data without duplicating storage space, making them suitable for situations where the underlying data is frequently updated or when storage space is a concern.

b) Consider a disk with block size 512 bytes. A block pointer is 6 bytes long, and a record pointer is 7 bytes long. A file has 30,000 EMPLOYEE records of fixed-length. Each record has the following fields: NAME (30 bytes), SSN (9bytes), DEPARTMENTCODE (9 bytes),ADDRESS (40 bytes), PHONE (9 bytes),BIRTHDATE (8 bytes), SEX (1 byte), JOBCODE (4bytes), SALARY (4 bytes, real number). An additional byte is used as a deletion marker.Assume that file is not ordered by the key field SSN and we need to create a secondary index on SSN.

(i) Find the number of levels needed, if we make it into a multilevel index.

Data file
No. of records r =30000
Block size B=512 bytes
Record size R=115 bytes
Blocking factor bfr=B/R=512/115=4.45=4
Number of file blocks ,b=r/bfr=30000/4=7500
Index file- 1st level
No. of records r =30000
Block size B=512 bytes
Record size R=9 bytes
Blocking factor bfr=B/R=512/9=56
Number of file blocks ,b=30000/56=536
Index file- 2nd level

Number of file blocks ,b=536/56=10

Index file- 3rd level

Number of file blocks ,b=1

(ii) Find the number of block accesses needed to retrieve a record from this file if we use the multilevel index.

Number of block accesses=4

17 a) Given relation R(A,B,C,D,E) and functional dependencies F={AB → C, CE → D, A → E}. Determine whether each functional dependency below is in F+ or not:

i) AB → D

ii) A → C

Both cannot be inferred

b) Consider the following relation:CAR_SALE(Car#, Date_sold, Salesperson#, Commission%, Discount_amt)Assume that a car may be sold by multiple salespeople, and hence {Car#,Salesperson#} is the primary key. Additional dependencies are :

Date_sold → Discount_amt and Salesperson# → Commission%

(i) Based on the given primary key and functional dependencies, is this relation in 1NF, 2NF, or 3NF? Why or why not?

(ii) How would you successively normalize it completely?

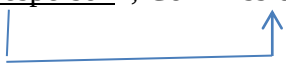
CAR_SALE(Car#, Date_sold, Salesperson#, Commission%, Discount_amt)

Since attributes are atomic relation is in 1NF

Decomposing into 2NF,to eliminate partial dependencies

Car#, Date_sold, Salesperson#, Discount_amt

Salesperson#, Commission%,



Since there is no transitive dependencies relation is in 3NF

18 a) Consider the following decompositions for the relation schema R into R1, R2 and R3.Determine whether the decomposition has the lossless join property with respect to the given F.

R={P, Q, R, S, T, U}

R1={P, Q}, R2= {R, S, T}, R3={P,R,U}

F = { P → Q, R → {S, T}, {P,R} → U}

P	**Q**	**R**	**S**	**T**	**U**
~~B11~~ a1	~~B12~~ a2	B13	B14	B15	B16
B21	B22	~~B23~~ a3	~~B24~~ a4	~~B25~~ a5	B26
~~B31~~ a1	~~B32~~ a2	~~B33~~ a3	~~B34~~ a4	~~B35~~ a5	~~B36~~ a6

Since there is a row of a's the decomposition has the lossless join property

b) Explain insert, update and delete anomalies with suitable examples.

Insertion Anomaly

Consider the relation:

EMP_PROJ(Emp#, Proj#, Ename, Pname, No_hours)

We cannot insert a project unless an employee is assigned to it. Conversely we cannot insert an employee unless an he/she is assigned to a project.

Deletion Anomaly

Consider the relation:

EMP_PROJ(Emp#, Proj#, Ename, Pname, No_hours)

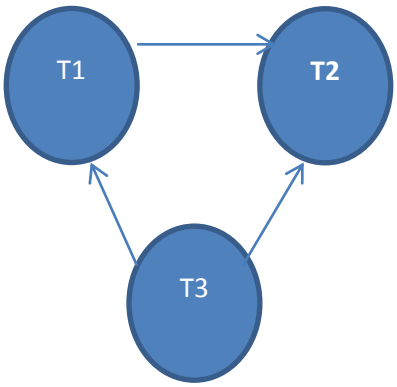
If we delete from EMP_DEPT an employee tuple that happens to represent the last employee working for a particular department, the information concerning that department is lost from the database.

Modification Anomaly

EMP_DEPT, if we change the value of one of the attributes of a particular department say,the manager of department 5 we must update the tuples of all employees who work in that department;otherwise, the database will become inconsistent.If we fail to update some tuples, the same department will be shown to have two different values for manager in different employee tuples, which would be wrong

19 a) Consider the schedule S of three transactions T1,T2 and T3 given below. State whether the schedule is serializable or not.

S: r3(Y), r3(Z), r1(X),w1(X), w3(Y), w3(Z),r2(Z), r1(Y), w1(Y), r2(Y), w2(Y), r2(X), w2(X)
(Hint: Interpret the notation r3(Y) as the operation read database item Y of transaction T3.)



Since there is no loop in precedence graph schedule is conflict serializable

b) Explain the lost update problem and temporary update problem that occur when concurrent execution is uncontrolled.

The lost update problem

This problem occurs when two transactions that access the same database items have their operations interleaved in a way that makes the value of some database items incorrect.

Suppose that transactions T1 and T2 are submitted at approximately the same time, and suppose that their operations are interleaved.Then the final value of item X is incorrect because T2 reads the value of X before T1 changes it in the database, and hence the updated value resulting from T1 is lost.

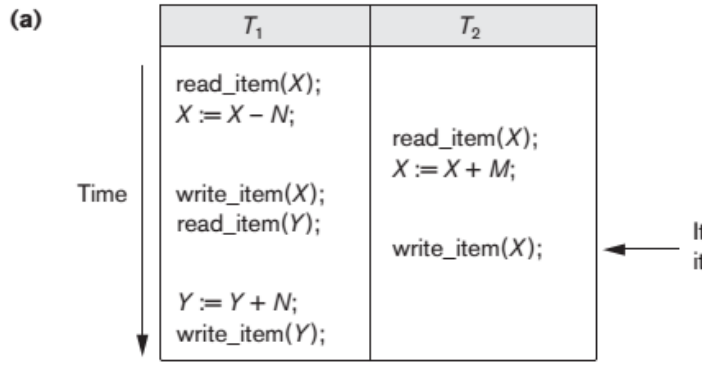


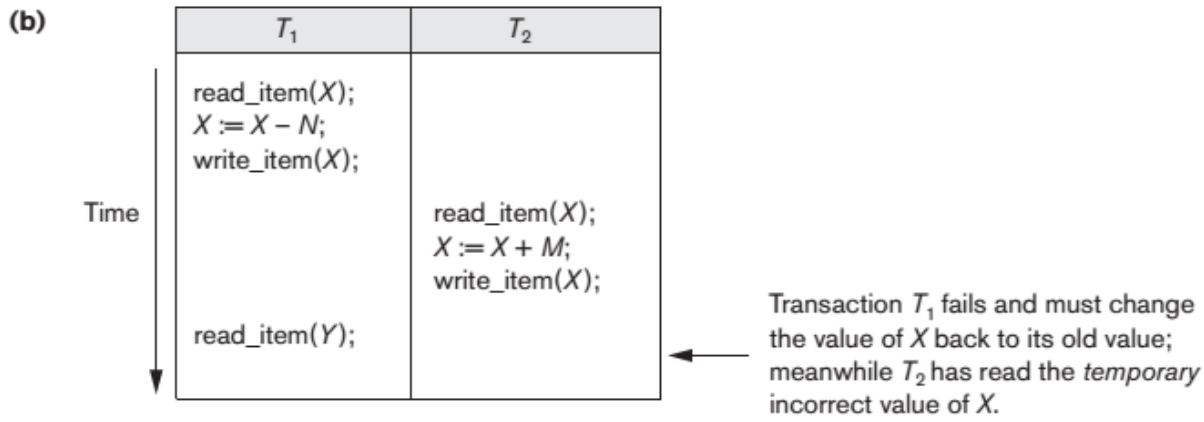
Figure 21.3
Some problems that occur when concurrent execution is uncontrolled. (a) The lost update problem. (b) The temporary update problem. (c) The incorrect summary problem.

If X = 80 at the start, N = 5 and M = 4 , then the final result should be X = 79. However, in the interleaving of operations shown in Figure 21.3(a), it is X = 8

because the update in T1that reduced 5 from X was lost

Dirty read (Temporary Update/ WR conflict) problem

This problem occurs when one transaction updates a database item and then the transaction fails for some reason. Meanwhile, the updated item is accessed (read) by another transaction before it is changed back to its original value.



T1 updates item X and then fails before completion, so the system must change X back to its original value. Before it can do so, however, transaction T2 reads the temporary value of X, which will not be recorded permanently in the database because of the failure of T1.

The value of item X that is read by T2 is called dirty data because it has been created by a transaction that has not completed and committed yet; hence, this problem is also known as the dirty read problem.

20 a) Explain conservative and strict two-phase locking techniques. Why strict 2PL is deadlock free?

Conservative Two-Phase Locking (2PL):

In conservative 2PL, transactions must acquire all the necessary locks they will need before they begin their execution.Each transaction requests all the required locks at the start, and if any of the requested locks are not available, the transaction is made to wait until all the locks can be acquired.Once a transaction acquires all its locks, it proceeds with its execution and holds the locks until it completes.Conservative 2PL ensures serializability by preventing conflicts between transactions but can lead to reduced concurrency as transactions may be waiting for locks even when there is no actual conflict.

Strict Two-Phase Locking (2PL):

In strict 2PL, transactions follow a strict protocol for acquiring and releasing locks.According to the protocol, transactions acquire all the necessary locks before accessing any data item and release all the locks only after the transaction has completed (either committed or rolled

back).Strict 2PL guarantees serializability and also ensures that the transaction's final outcome (commit or rollback) is not affected by any other transaction's actions.

Strict 2PL is deadlock-free, meaning it guarantees that deadlocks cannot occur during the execution of transactions.Deadlocks can occur when transactions wait indefinitely for resources held by other transactions, forming a cycle of waiting. Strict 2PL avoids deadlocks by requiring transactions to release all locks only at the end, thus eliminating the possibility of cyclic dependencies among transactions.

No Circular Wait: Strict 2PL ensures that transactions acquire all the required locks at the beginning and hold them until the end. This eliminates the possibility of circular wait, where one transaction is waiting for a resource held by another transaction while simultaneously holding a resource that the other transaction is waiting for. As a result, strict 2PL prevents the formation of deadlocks.

Locks Released at the End: In strict 2PL, locks are released only after a transaction has completed (committed or rolled back). By holding locks until the end, a transaction avoids the situation where it holds a lock and waits indefinitely for another lock held by a different transaction. This eliminates the potential for circular dependencies and breaks the conditions required for a deadlock to occur.

Serialization Guarantee: Strict 2PL ensures serializability, meaning that the concurrent execution of transactions produces the same result as if they were executed sequentially in some order. By enforcing a strict protocol for acquiring and releasing locks, strict 2PL ensures that the execution order of transactions is equivalent to a serial execution, which avoids the possibility of deadlocks.

b) Differentiate among recoverable, cascading rollback and strict schedules with suitable examples.

A schedule S is recoverable if no transaction T in S commits until all transactions T' that have written some item X that T reads have committed.

Sa = r1(X);r2(X);w1(X);r1(Y);w2(X);c2; w1(Y);c1;

Cascading rollback(or cascading abort): A phenomenon occurring in some recoverable schedules, where an uncommitted transaction has to be rolled back because it read an item from a transaction that failed.

Se: r1(X); w1(X);r2(X); r1(Y); w2(X); w1(Y); a1; a2;

In schedule Se, where transaction T2 has to be rolled back because it read item X from T1, and T1 then aborted.

A Strict Schedule, is in which transactions can neither read nor write an item X until the last transaction that wrote X has committed (or aborted).All strict schedules are cascadeless, and all cascadeless schedules are recoverable.

Sb = r1(X); w1(X); c1,r1(X);w2(X);c2;