# Answer Key

**Programme Name & Branch** : **B.Tech(CSE)**
**Course Code** : **BCSE302L**
**Course Name** : **Database Systems**
**Faculty Name(s)** : SWATHI J.N, MOHAN KUMAR P, NAGARAJA RAO A, LYDIA JANE G , NAVAMANI T M, ANAND BIHARI, SARASWATHI PRIYADHARSHINI A, VASANTHI P, KRISHNA RANI SAMAL K, KONATHAM SUMALATHA, THANGARAMYA K, DEEPIKA J, SAURABH AGRAWAL, DEEPA D, SUDHAKAR K, ARPITA GHOSH, MAHESWARI B, SARASWATHI U, ANUSHA R
**Class Number(s)** :
VL2025260101324/1328/1381/1385/1388/1373/ 1371/ 1396/1368/1363/1357/1351/1347/1343/1333/1375/1391/1378
**Date of Examination** : **18- 08 - 2025**
**Exam Duration** : **90 minutes** **Maximum Marks: 50**

1) a) Why would you choose database system instead of simply storing data in operating system files? When would it make sense not to use a database system? 6m

advantages of using a DBMS:

- Controlling Redundancy
- Restricting Unauthorized Access
- Providing Persistent Storage for Program Objects
- Providing Storage Structures and Search Techniques for Efficient Query Processing
- Providing Backup and Recovery
- Providing Multiple User Interfaces
- Representing Complex Relationships among Data
- Enforcing Integrity Constraints
- Permitting Inferencing and Actions Using Rules and Triggers

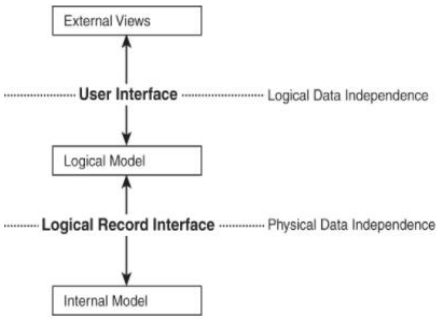
In spite of the advantages of using a DBMS, there are a few situations in which a DBMS may involve unnecessary overhead costs that would not be incurred in traditional file processing.

- For very small data where overhead of DBMS is unnecessary.
- For applications requiring extremely high speed with minimal data (e.g., embedded systems).
- For static data stored once and rarely updated (configuration files).

b) Explain the difference between logical and physical data independence. 4m

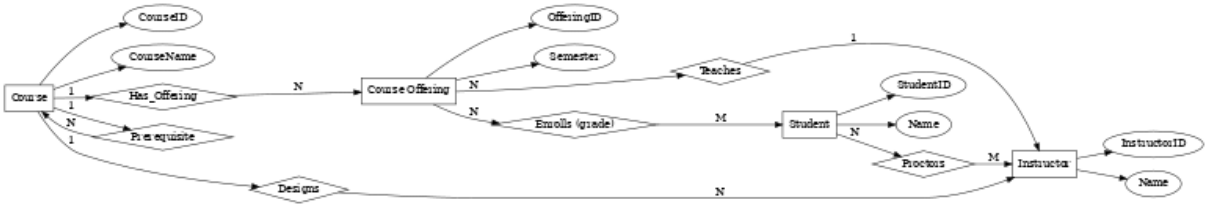
**Logical data independence:** It is the capacity to change the conceptual schema without having to change external schemas or application programs.

**Physical data independence:** It is the capacity to change the internal schema: without having to change the conceptual schema. Hence, the external schemas need not be changed as well.



# Answer Key

2) A university registrar's office maintains data about the following entities: (a) courses, with attributes including number, title, credits, syllabus, and prerequisites; (b) course offerings, is to model year, semester, section number, timings, and classroom; (c) students, with attributes student-id, name, and program; (d) instructors, with attributes identification number, name, department, qualifications and title. Further, the enrolment of students in courses and grades awarded to students in each course they are enrolled in must be appropriately modelled. Each student has exactly one instructor as a proctor. A proctor is a professor/instructor who proctors zero or more students. One or more courses are offered by an instructor, and one or more students register for a course offered by an instructor. A course can have zero or more prerequisites. The qualifications of an instructor can be many. One of the instructors is a HOD for a department. Also, the syllabus for each course is designed by one or more faculty members who are instructors. Construct an E-R diagram for the registrar's office as per the specifications. Document all other assumptions that you make about the mapping constraints (cardinality ratios).




Entities & Primary Keys:

• COURSE (PK: Course_ID; attrs: Title, Credits, Syllabus)

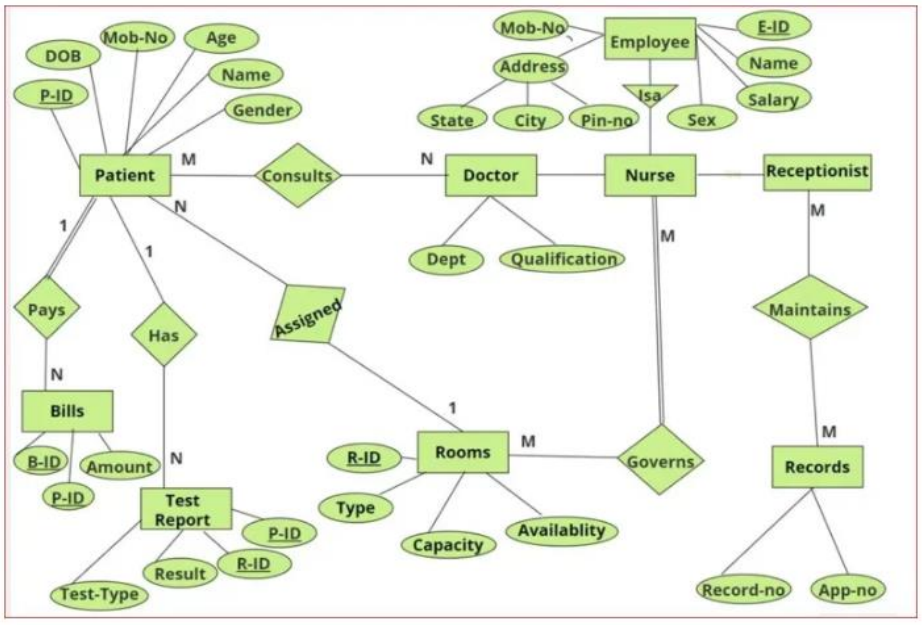
• COURSE_OFFERING (PK: Offering_ID; FK: Course_ID; attrs: Year, Semester, Section_No, Timings, Classroom)

• STUDENT (PK: Student_ID; attrs: Name, Program)

• INSTRUCTOR (PK: Instructor_ID; attrs: Name, Dept, Title)

• PROCTOR (PK/FK: Instructor_ID; subtype of INSTRUCTOR)

3) Map the given ER model to a relational schema. Identify and represent all the Constraints




Relational Schema:

PATIENT(P_ID, Name, DOB, Age, Gender, Mob_No)

# Answer Key

DOCTOR(D_ID, Name, Address, State, City, Pin_No, Dept, Qualification)
NURSE(N_ID, Name, Sex, Salary, Dept)
RECEPTIONIST(R_ID, Name, Salary)
ROOM(Room_ID, Type, Capacity, Availability)
BILL(Bill_ID, P_ID, Amount)
TEST_REPORT(Report_ID, P_ID, Test_Type, Result)
RECORD(Record_No, App_No, R_ID)

Relationships:
- PATIENT consults DOCTOR (CONSULTS(P_ID, D_ID), M:N)
- PATIENT assigned to ROOM (ASSIGNED(P_ID, Room_ID), 1:M)
- PATIENT pays BILL (1:M)
- RECORD maintained by RECEPTIONIST (1:M)
- DOCTOR governs ROOM (GOVERNS(D_ID, Room_ID), M:N)

Constraints:
- Primary Keys: P_ID, D_ID, N_ID, R_ID, Room_ID, Bill_ID, Report_ID, Record_No
- Foreign Keys: P_ID in BILL, TEST_REPORT; D_ID in CONSULTS; Room_ID in ASSIGNED

4) a) Discuss about super type and subtype classes of an EER model with one example. 6m

In many cases an entity type has numerous subgroupings or subtypes of its entities that are meaningful and need to be represented explicitly because of their significance to the database application. For example, the entities that are members of the EMPLOYEE entity type may be distinguished further into SECRETARY, ENGINEER, MANAGER, TECHNICIAN, SALARIED_EMPLOYEE, HOURLY_EMPLOYEE, and so on. The set or collection of entities in each of the latter groupings is a subset of the entities that belong to the EMPLOYEE entity set, meaning that every entity that is a member of one of these subgroupings is also an employee. We call each of these subgroupings a called the **superclass** or **supertype** for each of these subclasses. We call the relationship between a superclass and any one of its subclasses a **superclass/subclass** or **supertype/subtype** or simply **class/subclass relationship**. In our example, EMPLOYEE/SECRETARY and EMPLOYEE/TECHNICIAN are two class/subclass relationships. Notice that a member entity of the subclass represents the *same real-world entity* as some member of the superclass; for example, a SECRETARY entity 'Joan Logano' is also the EMPLOYEE 'Joan Logano.' Hence, the subclass member is the same as the entity in the superclass, but in a distinct *specific role.*

b) Why should nulls in a relation be avoided as far as possible? Discuss the problem of spurious tuples and about how we may prevent it. 4m

**Null Values in Tuples**

***Problem 1:*** If many of the attributes do not apply to all tuples in the relation: tuples will have many null values - Waste space at the storage level - Problem with understanding the meaning of the attributes - Problem with specifying JOIN operations at the logical level.

**Problem 2:** How to account for nulls when aggregate operations – COUNT/SUM are applied - Nulls can have multiple interpretations: *does not apply /unknown/ known but absent* -Having same representation for all nulls compromises the different meanings they may have.

# Answer Key

Design relation schemas so that they can be joined with equality conditions on attributes that are either primary keys or foreign keys in a way that guarantees that no spurious tuples are generated.

5) a) Consider a relational schema R (A B C D E F G H I J) and a set of functional dependencies as follows: F: {AB ->C, AD-> GH, BD ->EF, A-> I, H-> J}. Check out that relation R is in 3NF or not? If not decompose it into 3NF. 5m

Step 1: Candidate Key Calculation

- Closure of {A,B,D}: ABD+ = {A,B,D} ∪{C (AB→C), GH (AD→GH), EF (BD→EF), I (A→I), J (H→J)} = {A,B,C,D,E,F,G,H,I,J}. ABD is a candidate key.

Step 2: Check 3NF Conditions - A relation is in 3NF if for every FD X→Y: (i) X is a superkey OR (ii) Y is a prime attribute (part of candidate key).

- AB → C : AB is not a superkey, C is not prime → Violates 3NF.
- AD → GH : AD is not superkey, G,H not prime → Violates.
- BD → EF : BD not superkey, E,F not prime → Violates.
- A → I : A not superkey, I not prime → Violates.
- H → J : H not superkey, J not prime → Violates.

Step 3: Decomposition into 3NF

- R1(AB,C)
- R2(AD,GH)
- R3(BD,EF)
- R4(A,I)
- R5(H,J)
- R6(ABD) (to preserve candidate key)

b) Discuss about various types of anomalies with example. 5m

**1. Insertion Anomaly -** An insertion anomaly occurs when certain data cannot be added to the database without the presence of other data. This often happens when a table has attributes that are dependent on other attributes, leading to unnecessary data being inserted.

**2. Deletion Anomaly -** A deletion anomaly occurs when deleting data about one entity inadvertently leads to the loss of data about another entity. This typically happens when multiple pieces of information are stored in a single table, and deleting one record removes more information than intended.

**3. Update Anomaly -** An update anomaly occurs when changes to data require multiple updates to ensure consistency, and failing to update all instances leads to data inconsistency.