Reg. Number:	

VIT
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

Continuous Assessment Test (CAT) – I - JAN 2026

Programme	:	B.Tech .(CSE) with all specializations	Semester	:	Winter Semester 2025-2026
Course Code & Course Title	:	BCS302L Database Systems	Class Number	:	CH2025260502991, CH2025260502994, CH2025260503004, CH2025260503009, CH2025260503015, CH2025260503018, CH2025260503022, CH2025260503033, CH2025260503062, CH2025260503074, CH2025260503070, CH2025260503066
Faculty	:	Dr Rama Parvathy L, Dr. Revathi M, Dr. Om Kumar CU, Dr. Sandhya, Dr. Sherly A, Dr. Rishikeshan CA, Dr. Parvathy AK, Dr. Helen Vijitha P, Dr. Rajesh M, Dr. Sankar P, Dr. Sivaranjani N, Dr. Vijayakumar K P.	Slot	:	D1+TD1
Duration	:	90 Minutes	Max. Mark		50

General Instructions:
Write only your registration number on the question paper in the box provided and do not write other information

Answer all questions

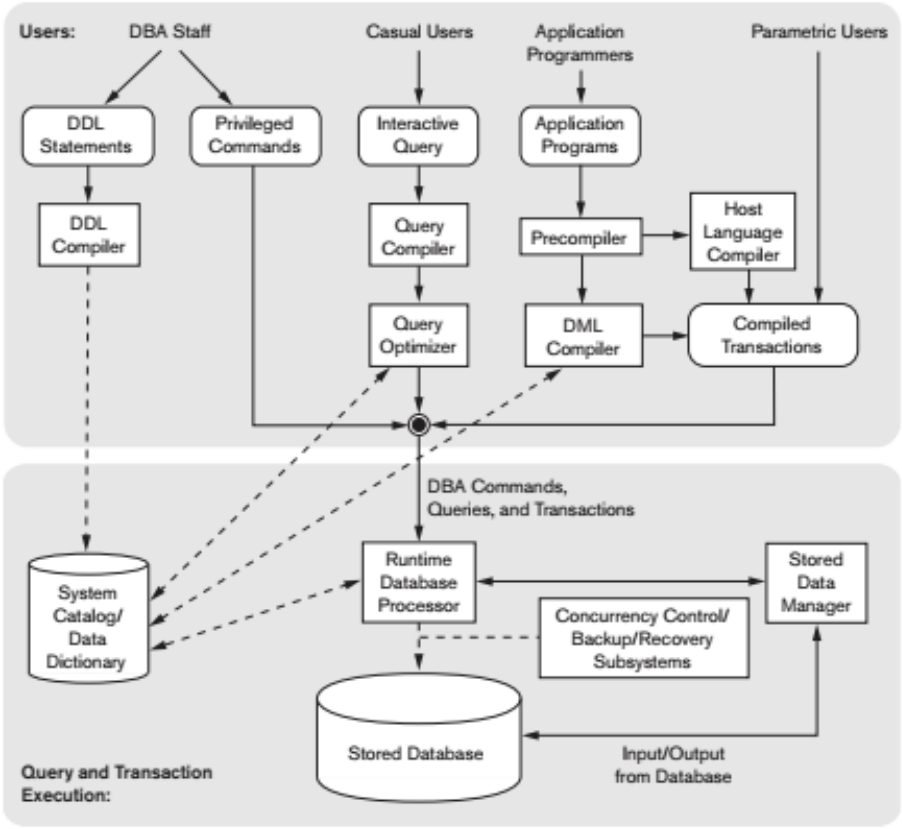
Q. No.	Description	Marks	CO	BT Level
1	a. An e-commerce website sells products that are identified with product_id, name, price, and quantity. During festive sales, price of the products and quantity will be changed frequently. Differentiate between intension and extension using this scenario. [3 Marks] b. A large commercial bank provides real-time banking services such as fund transfers, balance inquiries, loan approvals, and transaction auditing through multiple channels including branch offices, ATMs, mobile applications and internet banking portals. Behind the scenes, a centralized database system coordinates the activities of various users, application programs, DBMS software, and hardware resources. Discuss the required components of database system environment with a neat diagram. Highlights the role of various users in the given scenario. [7 Marks] **Intension (Schema):** • Intension refers to the **logical structure** of the database. • It defines **attributes, data types, and constraints**. • It does **not change frequently**. **Example (E-commerce):** PRODUCT(Product_ID, Name, Price, Quantity) This structure remains constant even during festive sales. **Extension (Instance):** • Extension refers to the **actual data stored** at a particular time. • It **changes frequently** as product price and quantity change during sales.	10	CO1	K2

Example:

Product_ID	Name	Price	Quantity
P100	Pen	10	50
P101	Pencil	5	75

✔ During festive sales, only **extension changes**, not the intension.

- b.

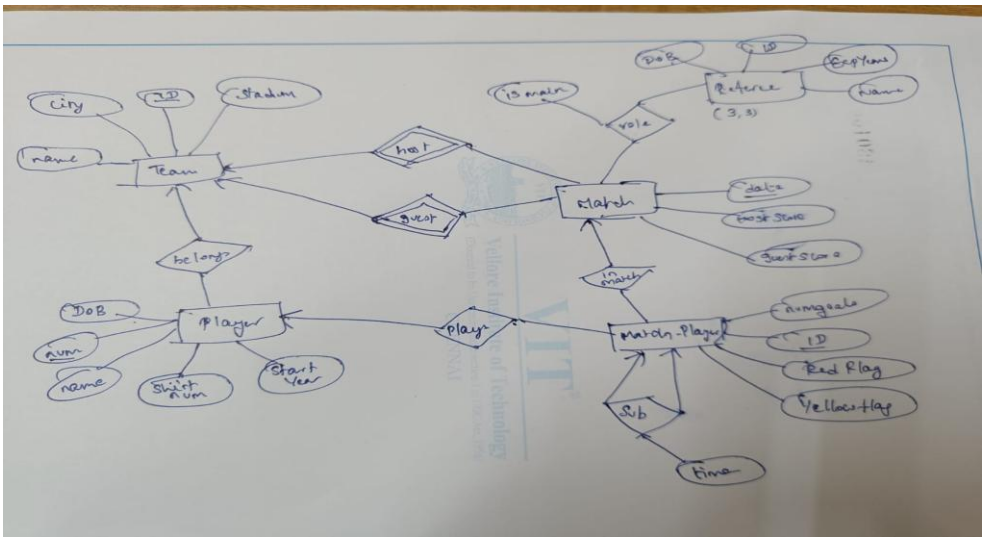


Components of Database System Environment

1. **Hardware**
   - Servers, storage devices, network infrastructure.
   - Supports real-time banking operations.
2. **Software**
   - DBMS (e.g., Oracle/MySQL)
   - Banking applications (ATM, Mobile App, Web Portal)
   - Operating System
3. **Data**
   - Customer records, account details, transaction logs, audit trails.
4. **Procedures**
   - Rules for transactions, security policies, backup & recovery.
5. **Database Users**
   - **End Users (Customers):**
     - Perform fund transfer, balance enquiry via ATM/mobile/web.
   - **Application Programmers:**
     - Develop banking software and write SQL queries.

	○ **Bank Staff:** ▪ Approve loans, verify transactions, audit accounts. ○ **DBA (Database Administrator):** ▪ Manages schema, security, backup, tuning, and recovery. 6. **DBMS Components** ○ DDL Compiler – processes schema definitions. ○ DML Compiler – processes queries. ○ Query Optimizer – improves query performance. ○ Runtime Database Processor – executes transactions. ○ Concurrency Control & Recovery Manager – ensures ACID properties. ✔ **Centralized DBMS ensures data consistency, security, concurrency, and reliability.**			
2	An online Movie Ticket Booking System manages movie shows and customer bookings as follows, • Every customer is registered in this system by using customer ID, customer name, city, and mobile number. • The cinema theatre consists of multiple screens which are identified by a movie ID. Every movie has a movie title, genre (Action, Comedy, Drama), and a ticket price. • Whenever a customer books a ticket for a movie, a booking record is created. Each booking contains booking ID, customer ID, movie ID, booking date, and booking status (Booked or Cancelled). Write SQL query for the following, a. Create all the tables with appropriate data types and constraints**.** [4 Marks] b. Display the customer name and movie title for all bookings with booked status. [2 Marks] c. List the customer name, movie title, and booking date for all bookings. Sort the result by booking date in ascending order. [2 Marks] d. Find the customer name and city of customers who have booked a Comedy movie for each city. [2 Marks] a) CREATE TABLE Customer (CustomerID INT PRIMARY KEY, CustomerName VARCHAR(50) NOT NULL, City VARCHAR(30), MobileNo VARCHAR(15) UNIQUE); CREATE TABLE Movie (MovieID INT PRIMARY KEY, MovieTitle VARCHAR(50) NOT NULL, Genre VARCHAR(20) CHECK (Genre IN ('Action', 'Comedy', 'Drama')), TicketPrice DECIMAL(7,2) NOT NULL); CREATE TABLE Booking (BookingID INT PRIMARY KEY, CustomerID INT, MovieID INT, BookingDate DATE NOT NULL,	10	CO1	K3

	Status VARCHAR(15) CHECK (Status IN ('Booked', 'Cancelled')), FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID), FOREIGN KEY (MovieID) REFERENCES Movie(MovieID)); b) SELECT (SELECT customer_name FROM Customer c WHERE c.customer_id = b.customer_id) AS customer_name, (SELECT movie_title FROM Movie m WHERE m.movie_id = b.movie_id) AS movie_title FROM Booking b WHERE b.booking_status = 'Booked'; c) SELECT (SELECT customer_name FROM Customer c WHERE c.customer_id = b.customer_id) AS customer_name, (SELECT movie_title FROM Movie m WHERE m.movie_id = b.movie_id) AS movie_title, b.booking_date FROM Booking b ORDER BY b.booking_date ASC; d) SELECT customer_name, city FROM Customer WHERE customer_id IN (SELECT customer_id FROM Booking WHERE booking_status = 'Booked' AND movie_id IN (SELECT movie_id FROM Movie WHERE genre = 'Comedy'));			
3	For a football game, an application has to be developed for the games they play, and the players in each team. In the design, we want to capture the following: • We have a set of teams, each team has an ID (unique identifier), name, main stadium, and to which city this team belongs. • Each team has many players, and each player belongs to one team. Each player has a number (unique identifier), name, DoB, start year, and shirt number that he uses. • Teams play matches, in each match there is a host team and a guest team. The match takes place in the stadium of the host team. • For each match we need to keep track of the following:	10	CO2	K3

	➢ The date on which the game is played ➢ The final result of the match ➢ The players participated in the match. For each player, how many goals he scored, whether or not he took yellow card, and whether or not he took red card. ➢ During the match, one player may substitute another player. We want to capture this substitution and the time at which it took place. • Each match has exactly three referees. For each referee we have an ID (unique identifier), name, DoB, years of experience. One referee is the main referee and the other two are assistant referee. Design an ER diagram to capture the above requirements.  			
4	A College Bus Management System (CBMS) is developed to efficiently manage transportation services for students and staff commuting between their residences and the college campus. The database is designed to support multiple academic years, ensuring data integrity, historical accuracy, and consistent enforcement of operational policies. The following business rules apply: • Each Bus is uniquely identified and must be assigned to exactly one Route at any given time. • A Route may have one or more Buses operating on it. • A Bus cannot operate without being associated with a valid Route. • Each Route consists of multiple Bus Stops arranged in a defined sequence. • A Bus Stop may be shared by multiple Routes, establishing a many-to-many relationship between Routes and Bus Stops. • A Driver may operate multiple Buses over different time periods. • Each Trip represents a single bus journey and must be operated by exactly one Bus and one Driver. • A Trip must always be associated with a valid Route. • A Trip record cannot exist unless the associated Bus, Driver, and Route already exist in the system. • Trips serve as historical records and must remain stored even after operational changes. • When a Bus is removed from service, it is marked as inactive rather than deleted. • All past Trip records associated with the bus must be retained for auditing and reporting purposes. • No new Trips may be assigned to an inactive Bus.	10	CO2	K3

- Students and Staff Members are registered as system users with unique identifiers.
- A Bus Pass may be issued to either a Student or a Staff Member for exactly one Route and one Academic Year.
- A Bus Pass cannot exist unless the corresponding user and Route exist.
- If a Student or Staff Member is deleted from the system, all Bus Passes associated with that user must be automatically deleted.
- This ensures no orphaned pass records remain in the database.
- All pass allocations and trip records must be associated with an Academic Year, enabling historical tracking and multi-year data consistency.

a. Translate the given business requirements into a relational database design by developing an appropriate relational schema along with necessary integrity constraints. Explain the role of these constraints in ensuring data consistency and correctness. (6 Marks)
b. Analyze how the proposed design avoids common database anomalies and maintains referential integrity in critical operational cases. (4 Marks)

Answers:

(a) Relational Schema + Constraints (6 Marks)

✔ **Schema refined & correct** (your original was good, explanation improved):

- ROUTE(Route_ID PK, Route_Name)
- BUS(Bus_ID PK, Reg_No UNIQUE, Capacity, Status, Route_ID FK)
- BUS_STOP(Stop_ID PK, Stop_Name)
- ROUTE_STOP(Route_ID FK, Stop_ID FK, Stop_Sequence)
- DRIVER(Driver_ID PK, Driver_Name, License_No UNIQUE)
- TRIP(Trip_ID PK, Trip_Date, Academic_Year, Bus_ID FK, Driver_ID FK, Route_ID FK)
- STUDENT(Student_ID PK, Student_Name)
- STAFF(Staff_ID PK, Staff_Name)
- BUS_PASS(Pass_ID PK, Academic_Year, Route_ID FK, Student_ID FK, Staff_ID FK)

Constraints ensure:

- Entity integrity (PK)
- Referential integrity (FK)
- Domain validity (CHECK)
- No orphan records (ON DELETE CASCADE)

b) Prevention of Anomalies and Referential Integrity (4 Marks)

1. **Insertion Anomalies**
   - A Trip cannot be inserted unless the referenced Bus, Driver, and Route exist.
   - A Bus Pass cannot be inserted without a valid user and route.
2. **Deletion Anomalies**
   - Deleting a Student or Staff member automatically deletes related Bus Passes, preventing dangling tuples.
   - Buses are not deleted when removed from service, preserving historical Trip data.
3. **Update Anomalies**
   - Route, bus, and user details are stored only once, ensuring consistent updates without redundancy.
4. **Referential Integrity in Critical Scenarios**
   - Inactive buses cannot be assigned new trips due to domain constraints.

	○ Past trips remain unaffected by operational changes, ensuring historical accuracy.			
5	Consider the schema R(A,B,C,D,E,F,G) with the following set of functional dependencies: F={A→BC, B→D, CD→E, A→D, E→F, AF→G, C→D, D→E} a. Find the closure of all the attributes given in the relation. [3 Marks] b. Find the canonical (minimal) cover for the given set of functional dependencies step by step. [7 Marks] **(a) Closure of attributes [3 Marks]** We compute the **closure of A**, since it is the most informative and helps identify keys. **A^+ (Closure of A)** Start with: $A^+ = \{A\}$ 1. **A → BC** ⇒ add B, C $A^+ = \{A, B, C\}$ 2. **B → D** ⇒ add D $A^+ = \{A, B, C, D\}$ 3. **D → E** ⇒ add E $A^+ = \{A, B, C, D, E\}$ 4. **E → F** ⇒ add F $A^+ = \{A, B, C, D, E, F\}$ 5. **AF → G** ⇒ add G (A and F present) $A^+ = \{A, B, C, D, E, F, G\}$ ✅ **Result** $$A^+ = \{A, B, C, D, E, F, G\}$$ 👉 Hence, **A is a candidate key**, and the closure of attributes in the relation is the **entire schema**.	10	CO2	K4
	(b) Canonical (Minimal) Cover – Step by Step [7 Marks] **Step 1: Make RHS single attribute** Split multi-attribute RHS dependencies: • A → BC ⇒ A → B, A → C • CD → E (already single RHS) Rewritten set: A → B A → C			

B → D

CD → E

A → D

E → F

AF → G

C → D

D → E

Step 2: Remove extraneous attributes from LHS

Check CD → E

- From **C → D**, we already get D from C
- So **C alone determines D**
- Then **D → E**

Thus, **CD → E is reducible to D → E**, which already exists.

✅ **Remove CD → E**

Check AF → G

- From **A → B, C, D, E, F**, we already get **F**
- Hence **A alone determines F**
- Therefore **A → G**

✅ **Replace AF → G with A → G**

Step 3: Remove redundant dependencies

Current set:

A → B

A → C

B → D

A → D

E → F

A → G

C → D

D → E

Check A → D

- A → B
- B → D
  So **A → D is redundant**

✅ **Remove A → D**

	(a) Closure $$A^+ = \{A, B, C, D, E, F, G\}$$ **(b) Canonical Cover** $$\{A \to B,\ A \to C,\ B \to D,\ C \to D,\ D \to E,\ E \to F,\ A \to G\}$$			

**********All the best **********

Faculty Signature