|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 1. **Consider the relation: SJT:(student, subject, teacher) with the set of**   **functional depedencies F: FD1: {student, subject} -> teacher FD2: teacher -> subject**   |  |  |  | | --- | --- | --- | | **Student** | **Subject** | **Teacher** | | **Ramesh** | **Maths** | **Prof. Raghav** | | **Ramesh** | **Physics** | **Prof. Raj** | | **Suresh** | **Maths** | **Prof. Raghav** | | **Suresh** | **Physics** | **Prof. Krishna** | | **Suresh** | **Chemistry** | **Prof. Hemant** |   **a. Identify the candidate key(s) of SJT. (1)**  **{student, teacher}+ --> {student, teacher, subject }**  **{student, subject}+ --> {student, subject, teacher }**  **There are two candidate keys**  **b. Explain the update anomalies before decomposition. (1)**  **SJT suffers from certain update anomalies. If we wish to delete the information that student Suresh is studying Physics, we cannot do so without at the same time losing the information that Prof. Krishna teaches Physics.**  **c. Analyze the two cases of decomposition based on the selection of primary key from the candidate keys. (6)**  **Case1:**  **If {student,teacher} is selected as primary key, then SJT is not in 2NF.**  **SJT: (student,subject,teacher)**  **FD2 is violating 2NF – FD2 is partial FD.**  **Decompose SJT into**  **ST(student,teacher) teacher references TJ(teacher)**  **TJ(teacher,subject)**  **The above decomposition is in 3NF and also in BCNF.**  **Case2:**  **If {student,subject} is selected as primary key, then SJT is in 3NF, but not in BCNF.**  **SJT: (student,subject,teacher)**  **FD2 is violating BCNF – Because in FD2, Teacher is a determinant, but not a candidate**  **key. The two BCNF projections are:**  **ST(student,teacher) teacher references TJ(teacher)**  **TJ(teacher,subject)**  **The above decomposition is in BCNF.**  **d.** After decompostion, is the resulting relations are free from update anomalies and satisfy dependency-preserving property? (2)  **Although the two projections does avoid certain anomalies, they are not independent. An attempt to insert a tuple for Ramesh and Prof. Krishna into ST must be rejected, because Prof.Krishna teaches Physics and Ramesh is already being taught Physics by Prof. Raj : yet the system cannot detect this fact without examining TJ. After decompositon, the FD1 {student, subject} --> teacher cannot be deduced from the FD2 teacher --> subject. Hence the FDs are not preserved.**   |  |  | | --- | --- | | 1. Consider the following relation: R={A,B,C,D,E,F} with the following set of FD’s F: (2+2+6)   FD1: A→CF  FD2: A→D  FD3: C→D  FD4: B→E Construct the minimal cover S for the above set F. | **<CO3>** | | Minimal cover S:  A->C, A->F, A->D, C->D, B->E  A->CF  C->D  B->E  candidate key {AB}+->{ABCDEF}  2NF:  R1: {A,B} where A,B is PK and A references R1(A), B references R2(B)  R2:{A,C,F,D} where A is PK  C->D is transitive FD violating 3NF  R3:{B,E} where B is PK  3NF:  Decompose R2 into R21 and R22  R21: {A,C,F} where A is PK and C references R22(C)  R22: {C,D} where C is PK  Final relations after 3NF: R1, R21,R22 and R3.   1. Consider the following relational schema: (3+3+4)   Emp( **eid : integer** , ename : string, dob : date, salary : real )  Works( **eid : integer , did : integer** , pct\_time : integer )  Dept( **did : integer** , dname : string, budget : real, managerid : integer )   1. Create a view *Senior\_Employee* on Emp that shows eid, name, dob and salary of all employees who were born before 1970’s. Is the view updatable? Justify your statement by writing appropriate SQL statements.   CREATE VIEW Senior\_Employee (empid, sname, age, salary) AS SELECT eid, ename, age, salary FROM Emp WHERE age >= 50;The view is updatable, since the primary key of the relation is included in the view. The view allows INSERT, UPDATE, DELETE through the view.   1. Give an example of view from the above schema, that suffers from   update operations or not updatable. Explain why your example presents the  update problem.  Use any group function to show it not updatable  CREATE VIEW Emp\_Dept AS SELECT did, count(\*)  FROM Emp E, Works W WHERE E.eid=W.eid  GROUP BY did;  The above view Emp\_Dept shows the number of employees  working in department­wise. The view is non­updatable. Because, the view contains  GROUP BY clause and JOIN operations.   1. Write a trigger Chk\_age which fires whenever an employee age is not in   the range of 20 to 56 yrs.  CREATE OR REPLACE TRIGGER Chk\_Age AS BEFORE INSERT OR UPDATE OF dob ON Emp  FOR EACH ROW  DECLARE  age NUMBER;  BEGIN  SELECT (floor(months\_between(:NEW.dob,sysdate)/12)  INTO age FROM DUAL;  IF (age<20) AND (age>56) THEN  raise\_application\_error(­20300, ‘Age OUT OF RANGE’);  END IF;  END;     1. Given R={A,B,C,D} with the following set of functional dependencies, F: (5+5)   A→BCD  BC→AD  D→B  Determine the two different minimal set of FD’s S and T given F using minimal cover.  writing in canonical form:  A->B, A->C, A->D, BC->A, BC->D, D->B  A->BC, BC->D gives A->D which is redundant, hence  A->B, A->C, BC->A, BC->D, D->B  No more redundant FD’s  S:A->BC  BC->AD  D->B  writing in canonical form:  A->B, A->C, A->D, BC->A, BC->D, D->B  BC->A, A->D gives BC->D which is redundant, hence  A->B, A->C, A->D, BC->A, D->B  A->D, D->B gives A->B which is redundant, hence  A->C, A->D, BC->A, D->B  T:  A->CD  BC->A  D->B  b. A car-rental company maintains a database for all vehicles in its current fleet. For all vehicles, it includes the vehicle identification number,license number, manufacturer, model, date of purchase, and color. Special data are included for certain types of vehicles: • Trucks: cargo capacity. • Sports cars: horsepower, renter age requirement. • Vans: number of passengers. • Off-road vehicles: ground clearance, drivetrain (four- or two-wheeldrive). Construct ER/EER diagram for the above specification along with the constraints |  | |  |  |  |
|  |  |  |  |
| PART-C   |  | | --- | | Consider the student registration database comprising of below schema: (3+3+4)  Student (**sno**, name, address, phone)  Course (**cno**, description, hours, pno)  Professor (**pno**, name, office)  Registration ( **sno, cno** , date)  Consider a suitable instance for the above mentioned relations.  Construct DML statements for the queries listed below:   1. Who teaches a course ‘CS6302’ and where is his/her office?   **select p.pno, p.name, p.office**  **from professor p**  **where p.pno in ( select c.pno**  **from course**  **where cno='CS6302');**   1. Who are the professors handling the courses registered by the student ‘S102’?   select \* from professor  where pno in ( select pno from course where cno in  ( select cno from registration where sno in  (select sno from student where sno='s002')  ) );  c. List out the courses not registered by a student ‘S104’. (Use EXISTS) | |  |  |  |
|  |  |  |  |
|  |  |  |  |