Chapter 6 Review Question and Exercises: 6.2, 6.16 (b, c, d, e, f, g, i, j) Chapter 8 Exercises: Specify the queries of exercises 6.16 (b, c, d, e, f, g, i, j) in SQL; 8.16 (b, c, e, f); 8.17 (b, c, d); 8.24(b, d) Chapter 10 Exercises: 10.11, 10.16, 10.19, 10.20, 10.21, 10.26, 10.28, 10.31, 10.33. Chapter 11 Exercises: 11.30. Chapter 17 Exercises: 17.17, 17.22(b, c, d), 17.23 Chapter 18: 18.1.

|  |  |  |
| --- | --- | --- |
| 6.2 | Union compatibility describes two relations who have the same degree and whose attribute domains match for all attributes.  Required for UNION, INTERSECTION, and DIFFERENCE because these operations examine tuples between the two operand relations and keep or discard the tuple based on the operation. If the two tuples had different degrees or had attributes with different domain values, the comparison would not make logical sense and couldn’t be done. | |
| 6.16 and SQL | b | SELECT LNAME, FNAME  FROM EMPLOYEE  WHERE EXISTS  (  SELECT \*  FROM DEPENDENT  WHERE FNAME=DEPENDENT\_NAME AND ESSN=SSN  );  no rows selected |
| c | SELECT LNAME, FNAME  FROM EMPLOYEE  WHERE SUPERSSN IN  (  SELECT SSN  FROM EMPLOYEE  WHERE FNAME='Franklin' AND LNAME='Wong'  );  LNAME FNAME  --------------- ---------------  Smith John  Narayan Ramesh  English Joyce |
| d | SELECT PNAME, SUM(HOURS)  FROM PROJECT, WORKS\_ON  WHERE PNUMBER=PNO  GROUP BY PNAME  ORDER BY SUM(HOURS) ASC  ;  PNAME SUM(HOURS)  -------------------- ----------  Reorganization 25  ProductY 37.5  ProductZ 50  ProductX 52.5  Newbenefits 55  Computerization 55 |
| e | SELECT LNAME, FNAME  FROM EMPLOYEE  WHERE NOT EXISTS (  SELECT PNUMBER  FROM PROJECT  WHERE NOT EXISTS (  SELECT \*  FROM WORKS\_ON  WHERE PNUMBER=PNO AND ESSN=SSN  )  );  no rows selected |
| f | SELECT LNAME, FNAME  FROM EMPLOYEE  WHERE NOT EXISTS (  SELECT \*  FROM WORKS\_ON  WHERE ESSN=SSN  );  no rows selected |
| g | SELECT DNAME, AVG(SALARY)  FROM DEPARTMENT, EMPLOYEE  WHERE DNUMBER=DNO  GROUP BY DNAME  ORDER BY AVG(SALARY) ASC  DNAME AVG(SALARY)  --------------- -----------  Administration 31000  Research 33250  Headquarters 55000 |
| i | SELECT LNAME, FNAME, ADDRESS  FROM EMPLOYEE  WHERE EXISTS (  SELECT \*  FROM WORKS\_ON, PROJECT  WHERE PLOCATION='Houston' AND PNO=PNUMBER AND SSN=ESSN  )  AND NOT EXISTS (  SELECT \*  FROM DEPT\_LOCATIONS  WHERE DNO=DNUMBER AND DLOCATION='Houston'  );  LNAME FNAME ADDRESS  --------------- --------------- ------------------------------  Wallace Jennifer 291 Berry, Bellaire, TX |
| j | SELECT LNAME, FNAME  FROM EMPLOYEE  WHERE EXISTS (  SELECT \*  FROM DEPARTMENT  WHERE SSN=MGRSSN  )  AND NOT EXISTS (  SELECT \*  FROM DEPENDENT  WHERE SSN=ESSN  );  LNAME FNAME  --------------- ---------------  Borg James |
| 8.16 | b | SELECT Course\_name  FROM COURSE  WHERE Course\_number IN  (  SELECT Course\_number  FROM SECTION  WHERE Instructor=’King’ AND (Year=’04’ OR Year=’05’)  ); |
| c | SELECT Course\_number, Semester, Year, COUNT(\*)  FROM SECTION, GRADE\_REPORT  WHERE Instructor='King' AND SECTION.Section\_identifier = GRADE\_REPORT.Section\_identifier  GROUP BY Course\_number, Semester, Year |
| e | SELECT Name, Major  FROM STUDENT  WHERE Student\_number IN  (  SELECT Student\_number  FROM GRADE\_REPORT  WHERE Student\_number NOT IN  (  SELECT Student\_number  FROM GRADE\_REPORT  WHERE Grade<>’A’  )  ); |
| f | SELECT Name, Major  FROM STUDENT  WHERE Student\_number IN  (  SELECT Student\_number  FROM GRADE\_REPORT  WHERE Student\_number NOT IN  (  SELECT Student\_number  FROM GRADE\_REPORT  WHERE Grade=’A’  )  ); |
| 8.24 | b | CREATE VIEW RESEARCH\_EMPLOYEE\_SUMMARY  (EmpLname, EmpFname, MgrLname, MgrFname, Salary )  AS SELECT…  get research employees…  SELECT Lname, Fname  FROM EMPLOYEE  WHERE Dno IN  (  SELECT Dnumber  FROM DEPARTMENT  WHERE Dname=’Research’  );  SELECT Lname, Fname  FROM EMPLOYEE  WHERE |
| d | My girlfriend and I broke up last night. |
| 10.11 |  | A minimal set of functional dependencies   * Has every FD with a single attribute for its right hand side * Has every FD with its left hand side as a minimal set of attributes (we cannot remove any attributes without causing the new FD to be nonequivalent) * And from which we cannot remove any FD without the new FD being nonequivalent   The minimal set of FDs is not always unique. Every set of FDs does have a minimal equivalent set. For example, if the set of FDs is the minimal set, it is equivalent to itself. If the set of FDs isn’t the minimal set, it can be decomposed into the minimal set using the algorithms in the book and the closure of the new set is equivalent to the closer of the old set, and the FD sets are thus equivalent. |
| 10.16 |  | Boyce-Codd Normal form is a normal form “higher” than 3NF. A relation is in BCNF if whenever a nontrivial FD X->A holds in R, then X is a superkey of R. This is different from 3NF because 3NF says the FDs are also allowed to have A as a prime attribute in R. |
| 10.19 |  | F = {A->C, AC->D, E->AD, E->H}  G = {A->CD, E->AH}  F+ = {A, C, E, H, D}  G+ = {A, E, C, H, D}  G+ == F+ => equivalent. |
| 10.20 |  | G = {Ssn->{Edate, Bdate, Address, Dnumber}, Dnumber->{Dname, Dmgr\_ssn}}  {Ssn+} = {Edate, Bdate, Address, Dnumber, Dname, Dmgr\_ssn}  {Dnumber+} = {Dname, Dmgr\_ssn} |
| 10.21 |  | No. The set of FDs is not minimal because:   * Not every FD has a single attribute for its right hand side   The attempted changes are:  Change FDs to canonical form (with 1 attribute on rhs):   1. Ssn->Edate 2. Ssn->Bdate 3. Ssn->Address 4. Ssn->Dnumber 5. Dnumber->Dname 6. Dnumber->Dmgr\_ssn   Remove redundant attributes from LHS:  No removals (as LHS is all single attribute).  Attempt to remove redundant FDs:  Remove i)? => G+ loses attribute.  No removals (as this results in loss of attributes in G+).  Therefore the list above is minimal.  Prove by: G+ = {Ssn+} U {Dnumber+}. |
| 10.26 |  | F = {AB->C, A->DE, B->F, F->GH, D->IJ}  The key is AB since there is no single attribute whose closure contains all attributes in R. AB therefore contains a minimal number of attributes and its closure is all attributes in R: {AB}+= ABCDEFGHIJ.  To obtain 2NF compute:  A+ = A, D, E, I, J  B+ = B, F, G, H.  So for 2NF:  R1(A, B, C) and R2(A, D, E, I, J) and R3(B, F, G, H).  To obtain 3NF compute look at the FDs in F which are valid for R1, R2, and R3 and break down any transitive dependencies.  R11(A, D, E) and R12(D, I, J) and R21(B, F) and R22(F, G, H) and R3(A, B, C). |
| 10.28 | a | A->B does not work because of tuples 1 and 2.  B->C could work.  C->B does not work because of tuples 1 and 3.  B->A does not work because of tuples 1 and 5.  C->A does not work because of tuples 1 and 3. |
| b | Tuple# is a CK.  A is not a CK because of tuples 1 and 5.  B is a CK because it determines {AC}.  C is a CK because it determines {AB}. |
| 10.31 |  | After a natural join the relation looks like:  R(O#, I#, Qty\_ordered, Total\_price, Discount%, Odate, Cust#, Total\_amount)  and the FDs for it are:  {O#, I#}->{Qty\_ordered}  {O#, I#}->{Total\_price}  {O#, I#}->{Discount%}  {O#}->{Odate}  {O#}->{Cust#}  {O#}->{Total\_amount}  This is not in 2NF ecause Odate, Cust#, and Total\_amount only partially depend on the joined key {O#, I#}. Since it isn’t in 2NF then it isn’t in 3NF. |
| 10.33 | a | The relation isn’t in 3NF because it isn’t in 2NF. It isn’t in 2NF because none of the attributes are fully functionally dependent on the key. |
| b | {Author\_name, Book\_title} is the key because its closure is all attributes.  First decompose to 2NF:  Author\_name->Author\_affil is PD on the PK, so it becomes its own relation.  Book\_title->Publisher, Book\_type is PD on the PK, so it becomes its own relation.  Book\_type->List\_price is not PD on the PK so it does not become its own relation.  So we now have:  R0(Book\_title, Author\_name)  R1(Author\_name, Author\_affil)  R2(Book\_title, Publisher, Book\_type, List\_price)  Further decomposition into 3NF is necessary because of the Book\_type->List\_price dependency.  R0(Book\_title, Author\_name)  R1(Author\_name, Author\_affil)  R21(Book\_title, Publisher, Book\_type)  R2(Book\_type, List\_price) |
| 11.30 | a | {M}+ = {MP, C} which is not all attributes and so M is not a CK.  {M, Y}+ = {MP, P, C, Y, M} which IS all attributes and so {M, Y} is a CK.  {M, C}+ = {MP, C, M} which is not all attributes and so {M, C} is not a CK. |
| b | The relation isn’t in 2NF because MP has a partial dependency on only M of the PK. Therefore it is not in 3NF or BCNF. |
| c | R(M, Y, P, MP, C)  R1(M, Y, P)  R2( M, MP, C)  Compute the closures of each FD in F as in part a.  R1 INTERSECT R2 = M and R2 - R1 = {MP, C}.  And so {M}->{MP, C} is part of F+, so this has the lossless property. |