* Big O = SELECT operation is denoted by  where σ (sigma) denotes the SELECT operator and the selection condition is a Boolean expression specified on the attributes of relation R.
* Pie = PROJECT Denoted by  where π (pi) represents the PROJECT operation and is the desired sub list of attributes from the relation R
  + Duplicate Elimination – does not output (removes) duplicate tuples
* Union = U
  + Find all cities that are a current city or a HomeTown for some user
  + 
  + Table

    Description automatically generated with low confidence
* Intersection = Icon

  Description automatically generated
  + Find all cities that are a currentCity for someone and a HomeTown for so RegularUser
  + 
  + Table

    Description automatically generated
  + SF is a part of both, appears in result, Same as Dallas.

A picture containing table

Description automatically generated

\*\*1st ping

Correct Mapping Of

|  |  |
| --- | --- |
| Voted relationship | Voted(Name, BillNumber, How) |
| DISTRICT entity | District(DNum, SName) |
| COUNTY entity with its associated  relationships, via foreign keys, to a relation then how many non-  NULL foreign keys will the relation have | 2 |
| map the COUNTY entity with its associated relationships via foreign keys, then which of the following statements will be true about the County relation? | The relation will have two columns associated with the SName of STATE. |
| create one relation to represent the combination of the Voted  and Sponsored relationships? | No |
|  |  |

\*\*Second image

|  |  |
| --- | --- |
| Item relation | ItemID and UnitPrice is a superkey  ItemID is primary |
| violate the referential  integrity constraint | Insert |45|103|2011-05-21|Null| into Order |
|  |  |
|  |  |

Statement about null values - a tuple in the OrderDetail relation cannot have a null value for ItemID

|  |  |
| --- | --- |
| True | False |
| Several attributes in a relation may have the same domain |  |
| From the mathematical definition of a relation, reordering of tuples does not  change the relation state |  |
| The relation state would change more frequently than the relation schema. |  |

OID 44 – Deletions are propagated - only two tuples from OrderDetail must be deleted

|  |  |
| --- | --- |
| Single tuple w/ value of 5: | UnitPrice(ItemID=1(Item))  QOnHand(ItemID=4(Item))  QOrdered(ItemID>2 and OID >43(OrderDetail)) |
| How many -> ItemID(OrderDetail) | 4 |
| How many - > Customer\*Order\*OrederDetail\*Item | 9 |
| OID(Order)XItemID(Item))-(OID,ItemID(OrederDetail) | 11 |
| Right outer join between OrderDetail(left) and Item(Right) | 10 |
| customers' Id and Phone number for  customers who dont have any orders | (CID(Customer)-(CID(Order)))\*(CID,Phone(Customer)) |

Same Relation Produced:

|  |  |
| --- | --- |
| True | False |
| OrderDetail.OID=Order.OID(OrderDetail X Order) | OrderDetail OrderDetail.OID=Order.OID Order |
|  | OrderDetail \* Order |
|  |  |

Returned By Query

|  |  |
| --- | --- |
| CID(Customer) – CID(Order) | (1)a relation with one column and two tuples whose values are 102 and 106  (2)a relation with IDs of customers who did not place any orders |
| Cname(Qordered>3(Order\*OrderDetail\*Customer) | Brown, Smith |
| Customer-((CID(Date=’2011-06-02’(Order)))\*Customer) | returns the CID, Cname, Phone and City of customers who did not place an  order on 2011-06-02 |
| OID((OID(Order)XItemID(Item))-(OID,Item(OrderDetail) | returns the OID of orders that dont include all items from the Item relation |
| OID = 44 | (1)OID(CID=105 Order\*OrederDetail) (2)OID(OrderDetail)-OID((CID<105 Order) (3)OID(QOredered>4 and ItemID=3(OrderDetail)) |
| {CID, c.Cname|Customer(c) AND (Ed)(Order(d) and d.Date=’2011-06-02’ AND c.CID=d.CID)} | (1)Is writted using tuple calculus (2)105 | Brown |
| {d.OID, d.ItemID|OrderDetail(d) AND (Et)(Item(t) and d.ItemID=t.ItemID AND d.QOrdered=t.QOnHand)} | finds the OID, ItemID pairs where item is ordered for a quantity that  equals the quantity on hand. |
| Left outer join – customers and order, Customers is left operand and Orders is right operand. | (1)include customer Doe and Green  (2)6 tuples in result |

\*\*END\*\*

E 1 Diagram (Same as previous)

|  |  |
| --- | --- |
| attributes of the relation HANGAR resulting from  mapping the entity type HANGAR | (1)Number, (2)Location, (3)Capacity |
| relation results from mapping the entity type  PERSON? | Person-nicknames  Person |
| attributes of the relation PERSON resulting from  mapping the entity type PERSON | (1)SSN, (2)Phone, (3) Address, (4)Owner-ID |
| attributes of the relation SERVICE resulting when  mapping the weak entity type SERVICE? | Reg#, Date, Workcode, Hours |
| key of the relation SERVICE resulting when  mapping the weak entity type SERVICE | Reg#, Date, Workcode |
| key of the relation EMPLOYEE resulting from  mapping the entity type EMPLOYEE? | Ssn |
| attributes of the relation WORKS-ON resulting  from mapping the many-many relationship type WORKS-ON | Model,SSN |
| attributes of the relation AIRPLANE  resulting from mapping the entity AIRPLANE | (1)Model, (2) Reg#, (3)Number |
| foreign key attributes of the relation AIRPLANE  resulting from mapping the entity AIRPLANE | (1)Model, (2)NUmber |
| Min and max number of attributes of a  superkey in the relation AIRPLANE? | 1 and 3 |
| many-many relationship type OWNS, which  relation will the attribute Pdate belong to | OWNS |

E2 Diagram

Relation

Must have at least one key

Contains no duplicate tuples

Must have at least one superkey

INTERSECTION = relational algebra operator requires type compatibility

|  |  |
| --- | --- |
| City,Type(Membership) | 5 |
| Club\*Meeting | 5 |
| Club X Member |  |
| Left outer natural join between Club and Meeting (Club is left) | 7 |
| {m.Mid|Membership(m)} | 6 |
| {c|Club(c) AND c.Fee>25 | 2 |
| Returns Club CID for clubs that held no meeting | Cid(Club\_-Cid(Meeting) |
| Name(Member\*dType=’office’ and CID>13(Membership)) | Don Black |
| Name(Name,City(City=’Atlanta’(Name,City,Fee(Club))) &&  Name(City=’Atlanta’(Club)) | True |
| Returns the Mids of persons who are members of all Clubs in Atlanta | (Cid,Mid(Membership)) % Cid(City=’Atlanta(Club)) |
| tuple calculus query returns the Cid and Name for clubs that did not have any meetings | {c.Cid,c.Name | Club (c) And (Am)(NOT(Meeting(m) OR NOT (m.cid=c.cid)))} |
| {c.Name,m.Name | Club(c) AND Member(m) AND (Eb)(Membership(b) AND b.Mid=m.Mid AND b.Cid=c.Cid AND b.Type=’officer’)} | Chess,Don Black – Chess, David Gray – Auto, David Gray – Movie, Don Black |

\*\*

|  |  |
| --- | --- |
| Mapping the SECTION entity type to a relation | Ssn, C# |
| Mapping MINOR to the relational model | Including DEPARTMENT Dname in the STUDENT relation |
| Mapping PERSON, FACULTY, STUDENT to relations – reduce NULL values | 3 relations – PERSON,FACULTY, STUDENT |
| Mapping CHAIRS to the relational mod | Either would work |
| Mapping TRANSCRIPT to the r mod | Separate relation for TRANSCRIPT |
| What happens multi-value Degree attribute when mapping to r mod | Separate relation created SSN, College, Degree, Year |
|  |  |
|  |  |

|  |  |
| --- | --- |
| ItemID,Qty)OrderItem) | 6 |
| Degree of rel - UnitPrice>15(Item\*OrderItem) | 4 |
| Cardinality - UnitPrice>15(Item\*OrderItem) | 1 |
| ItemID(Item)  ItemID(OrderItem) | 4 |
| ItemID(Item) U ItemID(OrderItem) | 5 |
| Left outer join, item is left and OrderItem is right. | 8 |
| {s.ItemId, s.UnitPrice, t.orderID, t.Qty | Item(s) AND OrderItem(t) AND s.ItemID=t.ItemID AND s.UnitPrice>15 | 1 |
| ItemID(Item)-ItemID(OrderItem) | Returns itemId of items that do not appear on any order |
| CustID(Oreder\*qty>2(OrderItem)) | 11 and 12 |
| CustID=11(UnitPrice<10(Qty>2(Order\*Item\*OrderItem))) | Both |
| Only consider relational db schema, which is true | ItemID(OrderItem) will never return more tuples than ItemId(Item) |
| Jack Black | Atlanta | Cname(CustID=13(Customer)) x city(CustID=11(Customer)) |
| |R| represents cardinality, which is true | |OrderItem\*OrderItem|=|OrderItem U OrederItem| |
| Projection of a list of attributes from a relation R return same number of tuples as relation R | When the list of attributes is a superkey for R |
| s.CustID, s.Cname, S.City, t.ItemIDmt.UnitPrice | Customer(s) AND Item(t) | Customer X Item |
| {t.City|Customer(t) AND (Es) (Order(s) AND t.CustID=s.CustID)} | Atlanta | Marietta |

SQL

|  |  |
| --- | --- |
| DROP TABLE User Cascade | (1)two foreign keys constraints are removed from RegularUser and AdminUser  (2)Table def is removed from catalog  (3)6 Tuples are updated |
| DELETE FROM User | (1)6 Tuples are deleted  (2)Deletion is rejected due to foreign key costraints  (3)the 2 foreign key constraints are removed from RegularUser and AdminUser |
| INSERT INTO Users VALUES (‘leo1@gt’, ‘abc’, ‘Mark’ | The tuple is inserted |
| INSERT INTO User VALUES (‘leo@gt’, ‘stu’, ‘Leo’); | rejected because of redundant email |
| INSERT INTO User VALUES (‘john@gt’, ‘john’, ‘John’) | The tuple is inserted |
| Attributes - SELECT \* FROM RegularUser | 5 |
| Tuples - SELECT DISTINCT CurrentCity FROM RegularUser; | 2 |
| Tuples - SELECT Sex FROM RegularUser | 4 |
| Tuples - SELECT CurrentCity FROM RegularUser UNION SELECT Hometown FROM RegularUser | 2 |
| Tuples - SELECT CurrentCity FROM RegularUser UNION ALL SELECT Hometown FROM RegularUser; | 8 |
| Attributes - SELECT \* FROM Employer, Jobs | 4 |
| Tuples - SELECT UserName FROM User WHERE Email LIKE ‘%y@gt’; | 2 |
| Attributes - SELECT \* FROM Jobs NATURAL JOIN Employer | 3 |
| Tuples - SELECT DISTINCT Email FROM Jobs NATURAL LEFT OUTER JOIN Employer; | 3 |
| Tuples - SELECT \* FROM RegularUser WHERE CurrentCity=’Atlanta’ AND Sex=’F’; | 2 |
| SELECT \* FROM USER NATURAL JOIN USER | 6 |
| SELECT Email FROM Jobs ORDER BY Email | ed@gt, ed@gt, mary@gt, mary@gt, mary@gt, rocky@gt |
| SELECT EmpName, COUNT (Email) FROM Jobs GROUP BY EmpName HAVING COUNT (\*) >= 2 ORDER BY EmpName | GT 3, UFL 2 |
| SELECT DISTINCT Email FROM Jobs WHERE JobTitle IN (Programmer, Professor, Researcher) | ed@gt, rocky@gt |
| SELECT Email FROM Jobs X WHERE NOT EXISTS ( (SELECT EmpName FROM Employer) EXCEPT (SELECT EmpName FROM Jobs Y WHERE X.Email=Y.Email)); | mary@gt, mary@gt, mary@gt |
| SELECT DISTINCT Email FROM Jobs X WHERE EXISTS ( (SELECT EmpName FROM Employer) EXCEPT (SELECT EmpName FROM Jobs Y WHERE X.Email=Y.Email)); | ed@gt, rocky@gt |
| Which query returns hightes number of tuples | (1) SELECT \* FROM USER UNION SELECT \* FROM USER  (2) SELECT \* FROM USER NATURAL JOIN USER  (3) SELECT \* FROM USER INTERSECT SELECT \* FROM USER |
| CREATE VIEW GTUsers AS SELECT Email, UserName FROM User WHERE Email LIKE ‘%@gt | The definition of the view is inserted |
| SELECT UserName FROM GTUsers | The SQL query is modified using the View definition, then the modified query is executed |
|  | |
| Rows - select startdate from reservation intersect select enddate from reservation | 2 |
| Rows-select rid from reservetools | 7 |
| Columns- select \* from reservation, reservetools, customer where customer.cemail = reservation.cemail and reservation.rid = reservetools.rid; | 9 |
| Rows- select \* from reservation, reservetools, customer where customer.cemail=reservation.cemail and reservation.rid=reservetools.rid | 7 |
| Rows - select startdate from reservation union all select enddate from reservation | 10 |
| Columns - select \* from reservation natural join (reservetools natural join customer); | 7 |
| Rows - select \* from reservation natural join (reservetools natural join customer); | 7 |
| Rows - select \* from (reservation natural join reservetools) natural join customer | 7 |
| Rows - select distinct r.cemail from reservation r, reservation s where r.cemail = s.cemail and (r.startdate - s.enddate) = 1; | 1 |
| select cname from (customer natural join reservation) 5/18/2020 Exam 3: DB Sys Concepts& Design - CS-6400-OAN https://gatech.instructure.com/courses/111492/quizzes/94889 9/21 where city = 'Marietta' and enddate > '13-AUG2017' | John Smith |
| select toolid from tools t where not exists ( select \* from reservetools where toolid = t.toolid | The toolid for tools that have not been reserved |
| Rows - select \* from customer, reservation | 20 |
| Possible null values - select \* from (tools natural left outer join reservetools) | rid |
| select cname from customer order by cname asc | Jane Doe |
| select cemail, rid from (reservation natural join reservetools) group by cemail,rid having count(\*) > 1 | jb@hotmail, 2  sb@gmail, 1 |
| select cname from customer where cemail in (select cemail from reservation where rid in (select rid from reservetools where toolid in (select toolid from tools where originalprice > 2000))) | Jane doe, john smith |
| select rid from reservation where cemail like '%@\_mail%' | 4, 1 |
| select rid from reservetools where toolid in (101, 105) | 1, 3, 4, 5 |
| Which of the following sets of Create Table statements would be appropriate for the relational schema | Create Table Customer (CEmail varchar(30), Cname varchar(30), City varchar(30), Primary key (CEmail));  Create Table Reservation (RID int, StartDate date, EndDate date, CEmail varchar(30), Primary key (RID), Foreign key (CEmail) references Customer (CEmail));  Create Table Tools (TooLID int, RentalPrice int, OriginalPrice int, Deposit int, Primary Key (ToolID));  Create Table ReserveTools (RID int, TooLID int, Primary Key (RID,ToolID), Foreign Key (RID) references Reservation (RID), Foreign Key ToolID references Tools (ToolID)) |
| Required in a syntactically correct SQL SELECT | Select and From |
| none of the attributes in the relational schema are set as NOT NULL – Which statement | insert into tools (toolid) values (108) |
| update Tools set deposit = deposit \* 0.5 where originalprice > 2000 and toolid not in (select toolid from reservetools); | toolId, rentalPrice, OriginalPrice, Deposit  101, 50, 2000, 100  102, 30, 1000, 60  103, 30, 800, 60  104, 40, 1500, 80  105, 50, 2200, 100  106, 30, 800, 60  107, 70, 3000, 70 |
| Select toolid from tools where deposit > 65 and deposit < 95 | 104 and 107 |
| create view rentalcost(rid, cost) as select r.rid, (1 + enddate - startdate) \* rentalprice from reservation r, reservetools s, tools t where r.rid = s.rid and s.toolid = t.toolid ;  What would the following query return? select rid, sum(cost) from rentalcost group by rid order by sum(cost) desc; | RID, SUM(Cost)  1, 400  2, 280  4, 200  5, 150  3, 50 |