Relational Database Operations(1)

Can be categorized into two groups:

- Updates
  - Insert
  - Modify
  - Delete

  All update operations must satisfy all constraints (entity integrity, referential integrity and enterprise constraints)

Relational Database Operations(2)

- Retrievals
  - Relational Algebra operations are used to specify retrievals
- Relational algebra operations include:
  - SELECT operation
  - PROJECT operation
  - Set theoretic operations
  - JOIN Operation

INSERT Operation(1)

- Used to insert a new tuple or tuples in a relation
- Provides a list of attribute values for a new tuple
- Insert can violate any of four types of constraints

INSERT Operation(2)

- Domain constraint can be violated
  - value of some attribute does not appear in its domain
- Key constraint can be violated
  - Key value of the tuple already exists in another tuple as its key value
- Entity integrity constraint can be violated
  - Key value of tuple is null
- Referential integrity constraint can be violated
  - Some foreign key value of tuple does not exist in the references relation

DELETE Operation

- Used to delete a tuple or tuples from a relation
- It can violate only referential integrity
  - If tuple being deleted is referenced by the foreign keys from other tuples in the database

MODIFY/UPDATE Operation

- Used to change the values of one or more attributes in a tuple or tuples of a relation
- Like INSERT operation all four constraints can be violated by UPDATE operation

Relational Algebra Operations(1)

- Enable user to specify basic retrieval requests
- Result of a retrieval is a new relation

Relational Algebra Operations(2)

- SELECT operation
- PROJECT operation
- Set theoretic operations
  - UNION, DIFFERENCE, INTERSECTION and CARTESION PRODUCT
- JOIN Operation
  - EQUI JOIN, NATURAL JOIN, INNER JOIN, OUTER JOIN, SELF JOIN

SELECT Operation(1)

- Used to select a subset of tuples from a relation that satisfy a *selection condition*
- In other words SELECT operation can be considered as a filter that keeps only those tuples which satisfy a qualifying condition

$$\sigma_{<\ select\ condition\ >}(\text{Relation})$$

- Here **sigma** denote select operator
- **Select condition** is the boolean expression specified on the attributes or relation
- **Relation** is either itself a relation or another select/project operation which results a relation

SELECT Operation(2)

- The resulting relation has the same attributes as the original relation.

Example Database

S

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris

SP

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S2	P1	300
S2	P2	400
S3	P2	200

P

P#	PNAME	COLOUR	WEIGHT	CITY
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London

SELECT Operation(3)

$$\sigma_{city = paris}(S)$$

S#	SNAME	STATUS	CITY
S2	Jones	10	Paris
S3	Blake	30	Paris

$$\sigma_{weight < 17}(P)$$

P#	PNAME	COLOUR	WEIGHT	CITY
P1	Nut	Red	12	London
P4	Screw	Red	14	London

SELECT Operation(4)

$$\sigma_{S\# = s1 \text{ and } p\# = p1}(SP)$$

S#	P#	QTY
S1	P1	300

PROJECT Operation(1)

- PROJECT operations selects certain columns from a relation and discards other columns and hence constructs a vertical subset of a relation
- When we are interested in only certain attributes of a relation we use PROJECT operation

PROJECT Operation(2)

$$\pi_{<\text{attribute list}>}(\text{Relation})$$

- Here **pi** denote project operator
- **Attribute list** is list of attributes to be projected
- **Relation** is either itself a relation or another select/project operation which results a relation

PROJECT Operation(2)

$\pi_{city}(S)$

CITY
London
Paris

$\pi_{sname,\ status}(S)$

SNAME	STATUS
Smith	20
Jones	10
Blake	30

Sequences of operations

- Many operations can be performed in sequence in one expression. For example you want part names where part weight is less than 17.

$$\pi_{pname}(\sigma_{weight > 17}(P))$$

- Both operations can be written seperately

$$Temp \leftarrow \sigma_{weight > 17}(P)$$
$$Result \leftarrow \pi_{pname}(Temp)$$

Set Theoretic Operations

- UNION, DIFFERENCE, INTERSECTION binary operations - they take two relations
- The two relations must be **union-compatible** i.e same degree and matching domains (ith column of first relation and ith column of second relation have same domain)

UNION Operation

- Denoted by RUS
- Results in a relation that includes all tuples that are either in R or S or in both R and S
- It is commutative i.e. RUS=SUR

INTERSECTION Operation

- Denoted by R∩S
- Results in a relation that includes all tuples that are both in R and S
- It is commutative i.e. R∩S =S∩R

DIFFERENCE Operation

- Denoted by R-S
- R-S is a relation that includes all tuples that are in R but not in S
- It is not commutative

CARTESION PRODUCT

- Cartesion product of R and S is denoted by RxS
- Also known as Cross Product or Cross join
- In R x S each row of S is paired with each row of R
- If there are m rows in R and n Rows then there will be m*n rows in RxS
- If there are *a* attributes in R and *b* attributes in S then there are *a+b* attributes in RxS

CARTESION PRODUCT(1)

- What will be the result of cartesion product of S and SP table?

S

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris

SP

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S2	P1	300
S2	P2	400
S3	P2	200

P

P#	PNAME	COLOUR	WEIGHT	CITY
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London

JOIN Operation

- JOIN operation used to combine *related tuples* from two relations into a single tuple
- Very important operation: allows to process relationships among relations

JOIN Operation Example(1)

- Consider employee and department relations of an organization

Employee

EID	ENAME
1000	Naveed
1001	Anees
1002	Khurram
1003	Asim
1004	Mohsin

Department

DID	DNAME	MGRID
101	Accounts	1000
102	Development	1001
103	Research	1001
104	Management	1000
105	Academics	1004

JOIN Operation Example(2)

- Now suppose we want to retrieve the name of manager of each department
- We need to combine each department tuple with the employee tuple whose EID matches with the MGRID in department tuple

JOIN Operation Example(3)

- We can do it in two ways
- We can simply use cartesion product first and then project required attributes

$$EMP\_DEPT \leftarrow EMPLOYEE \times DEPARTMENT$$

$$RESULT \leftarrow \pi_{DID, DNAME, ENAME} (\sigma_{EID = MGRID} EMP\_DEPT)$$

JOIN Operation Example(4)

- We can use JOIN operation (More common way)

$$DEP\_MGR \leftarrow DEPARTMENT \bowtie_{MGRID=EID} EMPLOYEE$$

$$RESULT \leftarrow \pi_{DID,\ DNAME,\ ENAME} (DEP\_MGR)$$

JOIN Operation Example(5)

- Result of both methods will be:

Result

DID	DNAME	MGRNAME
101	Accounts	Naveed
102	Development	Anees
103	Research	Anees
104	Management	Naveed
105	Academics	Mohsin

JOIN and Caresion Product

- Each tuple of result of JOIN is combination of one tuple from both relations
- Then what is difference between JOIN and Cartesion Product?
- Cartesion product is combniation of each row of first relation with each row of second relation
- In join only those combinations are included which satisfy the join condition

JOIN Condition and Theta Join

- General join conditions is of the form:

  <condition> and <condition> …..<condition>

  Where each condition is of the form Ai Θ Aj

  Ai and Aj are attributes of first and second relation respectively

  Ai and Aj have same domain

  And Θ may be any comparison operator (<,>,=,<=,>=,<>)

EQUI JOIN

- Most common join involves join condition with equality comparisons only
- Such join is called EQUI JOIN

Join Cont.…