Page :
Date : / /

# Assignment 2

Q.1) Explain the following relational algebra operation:-

a) Cartesian Product:- ① The Cartesian-Product operation, denoted by a cross (x), allows us to combine information from any two relations. We write the Cartesian product of relations $\gamma_1$ and $\gamma_2$ as $\gamma_1 \times \gamma_2$.

② A relation is a subset of a Cartesian product of a set of domains.

e.g:- If $C = \{\gamma_1, \gamma_2, \gamma_3, \ldots\}$ i.e $\gamma_1 \times \gamma_2 \times \gamma_3 \times \ldots$

$\therefore \gamma_1, \gamma_2, \gamma_3, \ldots \subseteq \gamma_1 \times \gamma_2 \times \gamma_3 \times \gamma_4 \times \ldots$

③ Since the same attribute name may appear in both $\gamma_1$ and $\gamma_2$, to distinguish between these attributes we devise a naming schema.

④ Let, us see by example given below:-

let, $\gamma_1$ = borrower (customer_name, loan_number)

$\gamma_2$ = loan (loan_number, branch_name, amount)

let, $\gamma = \gamma_1 \times \gamma_2$

then, $\gamma$ = borrower × loan (customer_name, borrower.loan_number, loan.loan_number, branch_name, amount)

∴ loan_number in both ($\gamma_1$ & $\gamma_2$) we use naming as relation_name.attribute_name to distinguish between them.

| Customer_name | loan_number |
|---|---|
| Adams | L-16 |
| Curry | L-93 |

The borrower relation ($r_1$)

| loan_number | branch_name | amount |
|---|---|---|
| L-11 | Round Hill | 900 |
| L-14 | Downtown | 1500 |

The loan relation ($r_2$)

| Customer_name | borrower. loan_number | loan. loan_number | branch_name | amount |
|---|---|---|---|---|
| Adams | L-16 | L-11 | Round Hill | 900 |
| Curry | L-93 | L-11 | Round Hill | 900 |
| Adams | L-16 | L-14 | Downtown | 1500 |
| Curry | L-93 | L-14 | Downtown | 1500 |

borrower × loan ($r_1 \times r_2$)

b) Natural-Join:- ① The natural join is a binary operation that allows us to combine certain selections and a Cartesian product into one operation.

② The natural-join operation forms a Cartesian product of its two arguments, performs a selection forcing equality on those attributes that appear in both relation schemas, and finally removes duplicate attributes.

③ Let, us see how its work by above example:-

$r_1 \bowtie r_2 = \sigma_{\text{borrower.loan\_number} = \text{loan.loan\_number}} (\text{borrower} \times \text{loan})$

↑ natural Join

Page :
Date : / /

④ We select those tuples in borrower x loan in which borrower.loan_number = loan.loan_number.

C)» Generalize projection:- ① The generalized-projection operation extends the projection operation by allowing arithmetic functions to be used in the project list. The generalized-projection operation has the form:- $\Pi_{F1,F2,F3,\ldots,Fn}(E)$

② Where E is any relational-algebra expression, and each $F_1, F_2, F_3, \ldots, F_n$ is an arithemetic expression involving constants & attributes in the schema of E. As a special case, the arithmetic expression may be simply an attribute or constant.

③ e.g:- Let, us take a relation with credit_info having attributes (customer_name, limit, credit_balance) using generalized projection:-

$\Pi_{customer\_name, (limit - credit\_balance)}(credit\_info)$

we will get tuple of customer_name and amount of limit-credit_balance we can give it a name using 'as'.

| customer_name | limit | credit-balance | |
|---|---|---|---|
| Curry | 2000 | 1750 | |
| Hayes | 1500 | 1500 | ← credit_info |
| Jones | 6000 | 700 | relation. |
| Smith | 2000 | 400 | |

| Customer_name | limit - credit_balance | |
|---|---|---|
| Curry | 250 | ← generalized projection. |
| Jones | 5300 | |
| Hayes | ~~530~~ | |
| Smith | 1600 | |

Page:
Date: / /

d)) Set operation:-

(i) Union operation:- If we want to show all the elements of two different relations in one table (can be more than two) having same number of attributes and corresponding attribute have same domain we use union operation.

e.g:- $\Pi_{customer\_name}(borrower) \cup \Pi_{customer\_name}(depositer)$
↑ (Union operator)

(2) It will give name of customers appear in either or both relations and will remove duplicate tuples.

(3) If we don't want to remove duplicate tuples we use Union all.

(ii) Set-Difference operation:- The set-difference operation, denoted by −, allows us to find tuples that are in one relation but are not in another.

(2) The expression r-s yields relation containing those tuples in r but not in s.

e.g:- $\Pi_{customer\_name}(borrower) - \Pi_{customer\_name}(depositer)$
↑ Difference operator.

(3) It will give name of customers which appears only in borrower and remove all depositor name from borrower name.

(4) It follows same constraint as Union operation i.e, relations should be of same arity, and that the domains of the $i^{th}$ attribute of r and the $i^{th}$ attribute of s be the same.

(iii) Set Intersection operation:- (1) Intersection operation will yield tuples similar in both relations.

(2) e.g. $\Pi_{customer\_name}(borrower) \cap \Pi_{customer\_name}(depositor)$
$\cap$ Intersection operator

(3) It will give customer name which appears in both borrower & depositor

(4) following equality holds:- $r \cap s = r - (r - s)$

(4) It follows same constraints as Union & Difference operations.

Q.2) Define and explain normalization with example.

(1) Normalization is a process of designing a consistent database by minimizing redundancy and ensuring data integrity through decomposition which is lossless.

(2) Normalization is step by step decomposition of complex records into simple records.

(3) Goals of database Normalization:-
(i) Ensures the integrity.
(ii) Prevents redundancy in data.
(iii) To avoid data anomaly.
<<a>> Update anomaly <<b>> Insertion anomaly
<<c>> Deletion anomaly.

(4) Normalization process will make, use of some type of keys to remove the redundancies present in data of relational tables:

5) Types of Normal Forms:-
(i) First Normal Form
(ii) Second Normal Form
(iii) Third Normal Form
(iv) ~~BCNF~~ BCNF Boyce-Codde Normal Form.

6) (i) First Normal form (1NF):- A database is said to be in 1NF if no table have multivalued attribute i.e each attribute of table have Single domain in 1NF.
(2) If a relation is not in 1NF then it can be translated to ~~one~~ 1NF by decomposing relation in two parts :- relation 1 will contain all Single domain attributes of original relation & relation 2 will have attributes of multivalued domain of original relation & both relation should have candidate key to uniquely identify each tuple & rejoin them in original relation.

7) Second Normal Form (2NF):- If database have Relation set $R = \{R_1, R_2, R_3, --\}$ & Functional dependency on R say set $F = \{F_1, F_2, --\}$ then each element of F should show fully functional dependency for database to be in a 2NF.

8) Third Normal Form:- A relation is in 3.N.F if all non-prime attributes are:-
i) Fully functionally dependent on primary key.
ii) Non-transitive dependent on every key.

Page :
Date : / /

9)) Boyce-Codd Normal Form (B.C.N.F) :- ① A relation R is said to be in a B.C.N.F if determinant of functional dependency contained candidate key i.e If $X \to Y$ then each attributes of X should contained in a candidate key.
② If a relation is in B.C.N.F then it is in 3NF also.

e.g:- Relation : R(A, B, C, D, E, F, G, H, I, J)
F.D $F = \{AB \to C, C \to EF, AD \to GH, G \to I, H \to J\}$
Convert to highest Normal form.

a) 1NF :- Assuming all attributes ~~a~~ are atomic domains ∴ R is in 1NF

b) 2NF :-

$\{ABD\}^+ = \{A, B, C, D, E, F, G, H\}$
ABD is a candidate Key of R but no ~~fun~~ attribute in relation R is fully functional dependent on above candidate key.
∴ R is not in 2NF ∴ decomposing it in 2NF.
∴ $R_1(A, B, C, E, F)$ ; $AB \to C$ ; $C \to EF$
$R_2(A, D, G, H, I, J)$ ; $AD \to GH$ ; $G \to I$ ; $H \to J$

c) 3NF:- Relation $R_1$ is not in 3·N·F
∴ $AB \rightarrow C$ & $C \rightarrow EF$ ∴ $AB \rightarrow EF$
Relational $R_2$ is not in 3·N·F
∴ $AD \rightarrow GH$ & $G \rightarrow I$ & $H \rightarrow J$
∴ $AD \rightarrow I$ ; $AD \rightarrow J$

Decomposing $R_1$ & $R_2$ in 3·N·F

| | | |
|---|---|---|
| $R_1$ | (A, B, C) | $AB \rightarrow C$ |
| $R_2$ | (C, E, F) | $C \rightarrow EF$ |
| $R_3$ | (A, D, G, H) | $AD \rightarrow GH$ |
| $R_4$ | (G, I) | $G \rightarrow I$ |
| $R_5$ | (H, J) | $H \rightarrow J$ |

we can combine $R_4$ & $R_5$ above.

d) B·C·N·F:-

| Relation | F·D | Determinant | Key |
|---|---|---|---|
| $R_1$ (A, B, C) | $AB \rightarrow C$ | AB | A, B |
| $R_2$ (C, E, F) | $C \rightarrow EF$ | C | C |
| $R_3$ (A, D, G, H) | $AD \rightarrow GH$ | AD | A, D |
| $R_4$ (G, I) | $G \rightarrow I$ | G | G |
| $R_5$ (H, J) | $H \rightarrow J$ | H | H |

Every determinant in each relation is a primary.
∴ All Relations above is in BCNF.

Page :
Date : / /

Q.3) Explain types of functional dependency.

Ans) ① Functional dependencies are restrictions imposed between two set of attributes in a relation from a database.

② In a relation R with attributes X and Y represented as R(X, Y), where Y is functional dependent on other column X or X functionally determines Y

$X \rightarrow Y$ i.e $Y = f(X)$.

X is called Determinant & Y is called determine.

## Types of functional dependencies:-

1) Fully functional dependcy:- A functional dependency is a fully functional dependency if removal of any attribute from determinant invalidate the dependency.

| emp_id | ename | Salary | Proj_id | Hours | allowance |
|---|---|---|---|---|---|
| 10 | Suresh | 50000 | E59 | 44 | 40000 |
| 12 | Mahesh | 25000 | B26 | 31 | 30000 |
| 15 | Adam | 26000 | E78 | 23 | 20000 |
| 18 | Jamal | 50000 | A59 | 12 | 15000 |

employee relation

If we take following F.D:-

emp_id, Proj_id → Hours, allowance

the removal of emp_id or proj_id will invalidate our F.D i.e we won't be able to determine Hours and allowance

∴ It is fully functional dependency.

Page :
Date : / /

2) Partial Functional dependency :-

(i) F.D is said to be Partial F.D if $A \to B$ & if remove $x \in A$ where $x$ is an attribute in A, doesn't invalidate F.D i.e F.D will hold.

(ii) In Partial F.D only part of determinant is needed for determine

If $A \to B$ where $A = \{x, y\}$

$x, y \to B$

If $y \to B$ holds then

$A \to B$ is Partial F.D.

(iii) e.g :- emp_id, proj_id → ename, salary

If we remove proj_id from above functional dependency we can still determine employee name & salary using emp_id i.e only emp_id is necessary determinant in above F.D ∴ it is a partial Functional dependency.

3) Transitive dependency :- If $X \to Y$ & $Y \to Z$ then $X \to Z$ is a transitive functional dependency

e.g :-

| emp_id | name | Salary | dep_id | d_name |
|---|---|---|---|---|
| 10 | Adam | 55,000 | ME05 | Mechanical |
| 11 | James | 80,000 | CS 10 | Computer |
| 14 | Jamal | 40,000 | CV09 | Civil |
| 15 | Qazi | 75,000 | IT69 | I.T |

In above relation we see that emp_id → dep_id

and dep_id → d_name so we can write emp_id → d_name i.e emp_id is i.e dname is transitively dependent on emp_id.

4) Trivial functional dependency:-

If $X \to\to Y$ & $Y \subseteq X$ then the functional dependency is said to be Trivial functional dependency.

If $X \to\to Y$ & $Y \not\subseteq X$ then the F.D is said to be non-Trivial functional dependency.

e.g:- emp_id, ename → ename ← Trivial F.D
emp_id, proj_id → allowance ← Non-Trivial F.D

5) Multivalued dependency:-

Multivalues dependency defined by $X \to\to Y$ is said to be hold for $R(X, Y, Z)$ if for a given set of values of X there is a set of associated values of attribute Y. Y values depends only on X values and have no dependence on the set of attribute Z.

e.g:-

| emp_id | ename | Car |
|---|---|---|
| 10 | Mahesh | Speedwagon |
| 12 | Suresh | Suzuki 69 |
| 15 | Ganesh | Hyundai 110 |
| 10 | Mahesh | BMW 420 |

emp_id →→ Car.

Page :
Date : / /

Q.4)) Explain lock based Protocol.

Ans) ① Database System equipped with lock-based Protocols use a mechanism by which any transaction cannot read or write data unit it acquires on appropiate lock on it.

② Locks are of two kinds:-

(i) Binary Locks:- A lock on a data item can be in two states; it is either ~~or~~ locked or unlocked.

(ii) Shared / Exclusive Locks:- This type of locking mechanism differentiate the locks based on their uses. If a lock is acquired on a data item to perform a write operation, it is an exclusive lock. Allowing more than one transaction to write on the same data item would lead the database into inconsistent state. Read locks are Shared lock because no data value is being exchanged.

③ There are four types of lock Protocols available:-

(i) Simplistic lock Protocol:- 1) Simplistic lock-based Protocols allow transactions to obtain a lock on every object before a 'write' operation is performed.

2) Transaction may unlock the data item after completing the 'write' operation.

(ii) Pre-claiming lock Protocol:- 1) It evaluate their operations and create a list of data items on which they need locks.

2) Before initiating an execution, the transaction requests the system for all the locks it needs before hand.

3)(ii) 3) If all the locks are granted the transaction executes and released all the locks when all its operations are over. If all the locks are not granted the transactions roll back and waits until all the locks are granted.

3) (iii) Two-phase locking 2PL:- 1) This locking protocol divides the execution phase of trasaction into three parts.
2) In the first part, when the transaction start executing, it seeks permission for the locks it requires
3) The second part, is where the transaction acquire all the locks. As soon as transaction releases its first lock.
4) In the third part, the transaction cannot demand any new locks; it only releases the acquired locks.
5) Two-phase locking has two phases, one is growing, where all the locks are being acquired by the transaction; & the second phase is shrinking, where the locks held by the transaction are being released.
6) To claim an exclusive (write) lock, a transaction must find acquire a shared [read] lock & then upgrade it to an exclusive lock.

(iv) Strict Two-Phase locking:- (1) The first phase of Strict-2PL is same as 2PL. After acquiring all the locks in the first phase, the transaction continues to execute normally.
(2) But in contrast to 2PL, Strict-2PL doesn't releases a lock after using it. It holds all the lock until commit point & releases all the lock at a time.
(3) Strict-2PL doesn't have cascading abort as 2PL does.

Page :
Date : / /

5) Explain log-based recovery.

Ans) ① The log is a sequence of records. Log of each transaction is maintained in some stable storage so that if any failure occurs then it can be recovered from there.

② If any operation is performed on the database, then it will be recorded in the log.

③ But the process of storing the logs should be done before the actual transaction is applied in the database.

④ let's assume there is a transaction to modify the city of a student. The following logs are written for this transaction :-

(i) When the transaction is initiated, then it writes 'start' log

⟨Tn, Start⟩

(ii) When the transaction modifies the city from 'Noida' to 'Bangloxe', then another log is written to the file.

⟨Tn, city, 'Noida', 'Bangloxe'⟩

(iii) When the transaction is finished, then it writes another log to indicate end of transaction

⟨Tn, Commit⟩.

⑤ There are two approaches to modify the database :-

(i) Defined database modification:

- It occurs if the transaction does not modify the database until it has committed.
- In this method, all the logs are created and stored in the stable storage, and the database is updated when a transaction commits.

(ii) Immediate database modification:-
- It occurs if database modification occurs while the transaction is still active.
- In this database is modified immediately after every operation. It follows an actual database modification.

(6) Recovery using log records:-
(i) When the system is crashed, then the system consults the log to find which transactions need to be undone and which needs to be redone.
(ii) If the log contains the record $\langle T_n, Start \rangle$ & $\langle T_n, commit \rangle$ then transaction needs to be redone.
(iii) If log contains record $\langle T_n, Start \rangle$ but doesn't contain the record either $\langle T_n, commit \rangle$ or $\langle T_n, abort \rangle$, then the transaction $T_n$ needs to be undone.