NEPAL COLLEGE OF INFORMATION TECHNOLOGY

Balkumari, Lalitpur

Affiliated to Pokhara University

ASSIGNMENT FOR DATABASE MANAGEMENT SYSTEM

ASSIGNMENT 3

Submitted by:
Name: Arpan Adhikari
Roll No.: 231309
BE-CE (3rd SEM)

Submitted to:
Er. Simanta Kasaju

Name : Arpan Adhikari
Roll No. 231309
BECE DAY

DBMS Assignment 3

1) What is key? Explain all kinds of keys present in DBMS with suitable examples.

→ Key is an attribute or set of attributes that uniquely identifies a tuple (record) in a table.

There are six kinds of keys present in DBMS. They are:

a) Super Key
It is a set of an attribute which can uniquely identify a tuple. It is a super set of a candidate key.
E.g. : $A \rightarrow B$ & $B \rightarrow C$, SK = A

b) Candidate Key
It is an attribute or set of an attribute which can uniquely identify a tuple. It is the proper subset of a super key. Every table must have at least a single candidate key.
E.g. $A \rightarrow B$ & $B \rightarrow C$, CK = A

c) Primary Key

It is the 1st key which is used to identify one & only one instance of an entity uniquely. It should be unique & not null. e.g. ID, ISBN no. etc.

d) Alternate Key

Those attributes that are left out after selection of P.K. are called alternate key. e.g. $A \rightarrow B, B \rightarrow C, C \rightarrow A$

PK = A, Alternate key = B, C

e) composite Key

If a key can be formed by combining two or more attributes, we call them composite key. e.g. ABC, AB, DE etc.

f) Foreign Key

If a PK in one table (parent) is referenced to another table (child), then we call them as foreign key.
e.g. S_id is PK in student table & Foreign key in Teacher table.

Q2) Explain the fundamental operations of relational algebra

→ Relational algebra is a formal query language used in databases that operates on relations (tables) &

provides a foundation for SQL. It consists of a set of fundamental operations to retrieve & manipulate data. They are:

a) Selection (σ)
- It selects tuples that satisfy a given condition.
Symbol : $\sigma_{condition}(R)$

b) Projection (π)
- It retrieves specific columns (attributes) from a relation, eliminating duplicate rows
Symbol : $\pi_{attributes}(R)$

c) Union ($\cup$)
- It combines two relations with the same attributes removing duplicates.
Symbol : $R_1 \cup R_2$

d) Set Difference ($-$)
- It retrieves tuples in one relation but not in another.
Symbol : $R_1 - R_2$

e) Cartesian Product ($\times$)
- It combines each tuple of one relation with every tuple

of another relation.
Symbol : $R_1 \times R_2$

f) Intersection ($\cap$)
- It retrieves tuples present in both relations.
Symbol : $R_1 \cap R_2$

g) Join ($\bowtie$)
- It combines tuples from two relations based on a related attributes. The most common type is Natural Join.
Symbol : $R_1 \bowtie_{condition} R_2$

Q3) what are the differences between primary, candidate & foreign key ?
→ They are :

Aspect	Primary key	Foreign key	Candidate key
Definition	A unique identifier for each record in a table.	A key in one table that reference as the primary key of another table.	A set of attributes that can uniquely identify a record.
Uniqueness	Always unique for each record.	May or may not be unique in the referencing table.	All candidate keys are unique.
Null values	Cannot contain NULL.	Can contain NULL if not a part of the relationship.	Can contain NULL if not chosen as the primary key.
Purpose	Uniquely identifies records within tables.	Establishes relationships beth tables.	Serves as a potential primary key.
Number Allowed	Only one primary key per table.	Can have multiple foreign keys referencing diff. tables.	Can have multiple candidate keys per table.
Example	Student ID in the Students table	ClassID in the students table referencing classID in classes table	Student ID, Email in the Students table

Q4) What is union compatibility (type compatibility)? Explain with an example.

→ Union compatibility is a property required for performing set operations on two relation (i.e. union (∪), intersection (∩) & set difference (−).

Two relations are union compatible if they meet following ~~tod~~ conditions:

- same number of attributes : Both relations must have the same number of columns.
- same attribute domains : corresponding columns in both relations must have the same datatype or domain.

Union Compatible Relations

R_1 : students - A

ID	Name	Age
1	Ram	20
2	Hari	23

R_2 : students - B

ID	Name	Age
3	Sita	24
4	Rita	21

No. of attributes : Both have 3 columns.

Domains : ID, Name, Age attributes in both tables have the same data types.

Union operation : students-A ∪ students-B

ID	Name	Age
1	Ram	20
2	Hari	23
3	Sita	24
4	Rita	27

Non-Union compatible relations :

R_1 : students-A

ID	Name	Age
1	Hari	22
2	Sita	23

R_2 : students-B

ID	Name	City
3	Ram	Dang
4	Gita	Ktm

Mismatch in Attribute names : Though the number of attributes is the same, the 3rd attribute is Age in first table, city in second table.

Domain Mismatch : Age is numeric in R_1 but city is a string in R_2.

So, relations are not union compatible.

Q5) Differentiate between Cartesian Product & Join.

→

Cartesian Product (×)	Join (⋈)
1. Combines every tuple of one relations with every tupe of another relation.	Combines tuples from two relations based on a specified condition.
2. No condition is required, all possible combinations are formed.	Requires a condition (e.g. matching attribute or other criteria)
3. Includes all attributes from both relations.	3. Includes attributes from both relations but can eliminate redundant one.
4. Forms the basis for other operations like joins.	4. Retrieves meaningful data by combining related tuples.
5. Computationally expensive for large relations.	5. More efficient due to filtering based on conditions.
6. e.g.: $R_1 = \{(A,1),(B,2)\}$ $R_2 = \{(X,3),(Y,4)\}$ Cartesian Product: $R_1 \times R_2 = \{(A,1,X,3),(A,1,Y,4)$ $(B,2,X,3),(B,2,Y,4)\}$	6. e.g.: $R_1 = \{(A,1),(B,2)\}$ $R_2 = \{(A,X),(B,Y)\}$ Join : $\{(A,1,X),(B,2,Y)\}$

Q6) What are procedural & non-procedural languages? Explain their differences & also mention the advantages of RA.

	Procedural Languages	Non-Procedural Languages
1.	Specifies how to perform a task step-by-step.	1. Specifies what result is reqd, not the steps.
2.	Focuses on the procedure (sequence of actions).	2. Focuses on the result or o/p.
3.	Requires explicit control flow (i.e. loops, conditions).	3. Abstracts control flow, focuses on declarative logic.
4.	Requires more effort to implement & debug.	4. Easier to use & more abstract.
5.	More flexible for complex tasks & algorithms.	5. Best suited for database queries & specific tasks.
6.	E.g.: C, C++, Java	6. e.g. SQL, HTML, Prolog

Advantages of RA:

* Foundation of Query languages: Basis of SQL & other query languages.
* Declarative Nature: Focuses on what to retrieve, not how.
* Mathematical Precision: Clear, well-defined operations ensure consistent results.
* Query optimization: Helps in optimizing queries for better performance.
* Data Independence: Abstracts data storage, unaffected by changes in structure.

* Standardize Approach : Provides consistency across relational databases.

Q7) Differentiate between Inner & Outer Join.

→

Inner Join	Outer Join
1. combines rows from two relations where is condition is met.	1. Combines rows from two relations including unmatched rows from one or both tables.
2. Only returns rows with matching values in both relations.	2. Returns matching rows & unmatched rows.
3. Typically smaller as it only includes matching rows.	3. Larger as it includes unmatched rows along with the matched ones.
4. It is only one type i.e. INNER join.	4. It is of three type i.e. left-outer join, right outer join & full outer join.
5. It excludes rows that do not match in both tables.	5. It includes all rows from one or both tables, even if there is no match.
6. Used when you only need data that exists in both tables.	6. Used when you need all data from one or both tables, regardless of matches.

Q8) What are the difference between Entity & Domain constraints?

	Entity constraints	Domain Constraints
1.	It ensure the uniqueness & identify of rows in a table.	1. It ensures the validity & consistency of values in column
2.	It focuses on maintaining the uniqueness & existence of entities (rows).	2. It focuses on ensuring data falls within a specific range, type or set of permissible values.
3.	Applies to rows & their relationships within table.	3. Applies to the values within a specific column.
4.	Used to uniquely identify records & enforce the integrity of the entity.	4. Used to validate individual column values to meet application specific rules or datatype restrictions.
5.	A table with duplicate primary key values violates entity constraints.	5. A column accepting a -ve value where only positive values are allowed violates domain constraints.

(Q9) What are the advantages of constraints in DBMS? Also, mention the difference between key & referential constraints.

→

Constraints in a database provide rules to ensures data integrity, consistency & validity. Its advantages are:

* Data Integrity : Ensures data in the database is accurate, valid & consistent.
* Automated Validation : Reduces the need for application level checks by enforcing rules at the database level.
* Consistency across applications : Enforces uniform rules across all applications interacting with the database.
* Error Prevention : Prevents invalid data from being entered into the databases i.e. negative salaries, duplicate IDs etc.
* Simplified Maintenance : Makes database maintenance easier by centralizing rules instead of spreading them across application code.
* Performance Optimization : Some constraints, like primary keys & indexes can improve query performance.
* Relational Integrity : Ensures relationships between tables are valid & maintained (i.e. foreign key constraints).

Relational Algebra →

RA answers:

1. a) → $Emp \leftarrow Emp \cup (100, 'Ram', 'Balaju', 1000, 5)$

b) $\sigma_{salary > 10000 \wedge Dname = "computer"} (Emp \bowtie Depart)$

c) $Emp \leftarrow \Pi_{Eid, Ename, Address, Salary * 1.1, Dptid} (Emp)$

d) $Emp \leftarrow Emp - \sigma_{Dptid = "ELX"} (Emp)$

2. a) $\Pi_{sname} (\sigma_{boatid = 105} (((Sailor \bowtie reserves) \bowtie boat)))$

b) $\Pi_{sname} (\sigma_{color = "red"} (((Sailor \bowtie reserves) \bowtie boat)))$

c) $\Pi_{sname} (\sigma_{color = "red" \vee color = "green"} (((Sailor \bowtie reserves) \bowtie boat)))$

d) $\Pi_{sname, age} (\sigma_{color = "red"} (((Sailor \bowtie reserves) \bowtie boat)))$

3. a) $\Pi_{a\_name, city} (((Book \bowtie Instock) \bowtie Branch))$

b)

c) $Book \leftarrow Book - \sigma_{a\_name = "xyz"} (Book)$

d) $\Pi_{title, subject} \; subject\,\mathcal{G}_{subject, count(isbn)} (Book \bowtie Topic)$

4.(i) $\Pi_{cust\_name}(\sigma_{branchname="Koteshwor"}(((customer \bowtie Borrower) \bowtie Loan)))$

(ii) $\mathcal{G}_{max(balance)}(Account)$

(iii) $\Pi_{cust\_name, account\_number, cust\_street, cust\_city}((Account \bowtie Depositor) \bowtie customer)$

5.(i) $\Pi_{ename}(\sigma_{company\_name="NMB"}(works))$

(ii) $\Pi_{Ename}(Employee \bowtie_{employee.city = company.city} company)$

(iii) $\Pi_{Ename, Employee.city}(\sigma_{salary > 30000 \wedge city="ktm"}(Employee \bowtie works))$

6.(i) Π

(ii) $\Pi_{name}(\sigma_{budget > 50000}((Employee \bowtie works\_on) \bowtie Project))$

(iii) $_{dname}\mathcal{G}_{count(pid), dname}(Department \bowtie Project)$

7) No table given

8) a)

b) $\Pi_{name}(\sigma_{title="Network\_monitoring" \wedge hours > 15}((Employee \bowtie works\_on) \bowtie Project))$

c) $Project \leftarrow Project - \Pi_{name="Electrical"}(Department \bowtie Project)$

d) ${}_{name}\mathcal{G}_{count(pid),\, name}(Project \bowtie Department)$

9. a) $\Pi_{title,\, name}(Department \bowtie Project)$

b)

c) ~~$Works\_on \leftarrow Works\_on$~~

c) $Works\_on \leftarrow \Pi_{eid,pid,(hours*5)}(\sigma_{title="voter\_registration"}(works\_on \bowtie Project))$

d) ${}_{title}\mathcal{G}_{count(eid),\, title}((Employee \bowtie works\_on) \bowtie Project)$

10. a) $\Pi_{ssn, name}(\sigma_{category = "telephone"}(Product \bowtie manages))$

b) $\Pi_{pid, name}(\sigma_{price > 500 \wedge country = "china"}(Product \bowtie company))$

c) $Product \leftarrow \Pi_{pid, name, price * 1.10, category, cid}(\sigma_{category = "Television"}(Product))$

d)

11. a) $\Pi_{year}(\sigma_{Title = "DBMS" \wedge price = 2000}(Book \bowtie Written\_by))$

b) $\Pi_{Name}(\sigma_{Title = "OS" \wedge Pages > 2000}((Book \bowtie Written\_by) \bowtie Author))$

d) $\Pi_{Aid, Title}(\sigma_{year = 1994}(Book \bowtie Written\_by))$

c) $\Pi_{ISBN, title, Aid, year}(\sigma_{Name = "simanta"}((Book \bowtie Written\_by) \bowtie Author))$

e) $\Pi_{category, pages, title, price}(\sigma_{ISBN = 123123}(Book))$

f) $\Pi_{title}(\sigma_{year = 1994 \wedge Name = "simanta"}((Book \bowtie Written\_by) \bowtie Author))$

12. i) $\Pi_{emp\text{-}name, city}(\sigma_{company\text{-}name = \text{"Nabil Bank"}}(WorksIn \bowtie Employee))$

ii) $\Pi_{emp\text{-}name, street, city}(\sigma_{company\_name = \text{"Nabil Bank"} \wedge salary > 24000}(WorksIn \bowtie company))$

iii) $Employee \leftarrow \Pi_{emp\text{-}name, street, city = \text{"Kathmandu"}}(\sigma_{name = \text{"Kiran"}}(Employee))$

iv) $WorksIn \leftarrow \Pi_{emp\text{-}name, company\_name, (salary * 1.1)}(\sigma_{company\_name = \text{"Nabil Bank"}}(WorksIn))$

13. a) $\mathcal{G}_{sum(salary)}(works)$

b) $Employee \leftarrow \Pi_{person\_name, street, city = \text{"Kathmandu"}}(\sigma_{person\_name = \text{"Ram"}}(Employee))$

c) $\Pi_{person\_name, street, city}(\sigma_{bank\_name = \text{"Nepal world Bank}}$

corporation" ∧ salary > 1000 (works ⋈ Employee))

d) works ← works − $\sigma_{bank\_name = \text{"small bank corporation"}}$ (works)

f) $\Pi_{person\_name, city}$ ($\sigma_{bank\_name = \text{"Nepal Rastra Bank"}}$ (Employee ⋈ Bank)

e) Π_{person_name} ($\sigma_{bank\_name = \text{"Nepal Rastra Bank"} \wedge salary > 10000}$ (works))

g) $\Pi_{person\_name, street, city}$ ($\sigma_{bank\_name = \text{"Nepal Rastra Bank corporation"} \wedge salary > 10000}$ (Works ⋈ Employee))

h) Works ← Works − $\sigma_{bank\_name = \text{"Nepal Rastra Bank"}}$ (works)

i) Π_{person_name} (Works ⋈ $_{works.salary > manages.salary}$ Manages)

j) Π_{person_name} (Employee ⋈ $_{Employee.city = Bank.city \wedge Employee.street = Bank.street}$ Bank)

20. a) $\Pi_{sid, sname, rating, age}(\sigma_{bid=103}(Sailors \bowtie Boats))$

b) $Boat \leftarrow \Pi_{bid, bname, color="Green"}(\sigma_{bid=104}(Boats))$

c) $\Pi_{sname}(\sigma_{color="red" \vee color="Green"}(Sailor \bowtie Boats))$

d) $\Pi_{sname}(\sigma_{bid=103 \wedge day=5}(Sailors \bowtie Reserves))$

e) $\Pi_{sname} - \Pi_{sname}(\sigma_{sname="Ram"}(Sailor))$

f) $\Pi_{bname}(Boats)$

21. a) $\Pi_{person\text{-}name}(\sigma_{city="Butwal" \wedge salary<50000}(Employee \bowtie Works))$

b) $\Pi_{person\text{-}name}(\sigma_{company\text{-}name="Nabil\ Bank\ Limited"}(Works))$

c) $\Pi_{person\text{-}name, city}(\sigma_{company\text{-}name="Global\ Bank"}(Employee \bowtie Works))$

d) Works $\leftarrow \Pi_{\text{person\_name, company\_name, salary} * 1.1}$ (Works)