CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE &ENGINNERING

Bachelor of Engineering (Computer Science & Engineering)

Database Management System and CST-227

Prepared By: Ms. Heena Arora

Database Management System

Course Outcome

CO Number	Title	Level
CO1	Learn the fundamentals of database systems design and draw ER diagram for the real world problems.	Understand
CO2	Design and query database using SQL.	Apply
CO3	Analyze and apply concepts of normalization to relational database design	Understand
CO4	Learn the concept of transaction, concurrency and recovery.	Remember

Introduction

- **Relational algebra**
  - Basic **set of operations** for the **relational** model
  - More operational **(procedural),** very useful for **representing execution plans**.

- **Relational calculus**
  - **Higher-level declarative** language for specifying **relational queries**
  - Lets users describe **what they want, rather than how to compute** it: **Non-operational**, declarative.

Relational Algebra

- Relational algebra operations work on **one or more relations to define another relation** without changing the original relations.
- **Both operands and results are relations**, so **output from one** operation can become **input to another** operation.
- Allows **expressions to be nested**, just as in arithmetic. This property is called **closure**.
- Five **basic operations** in relational algebra: **Selection, Projection, Cartesian product, Union and Set Difference**.
- These perform most of the **data retrieval operations** needed.
- Also have **Join, Intersection, and Division operations**, which can be expressed in terms of 5 basic operations.

Operations

- Basic operations:
  - *Selection* ( ) Selects a subset of rows from relation.
  - *Projection* ( ) Deletes unwanted columns from relation.
  - *Cross-product* ( ) Allows us to combine two relations.
  - *Set-difference* ( ) Tuples in reln. 1, but not in reln. 2.
  - *Union* ( ) Tuples in reln. 1 and in reln. 2.
- Additional operations:
  - Intersection, join, division

Selection

$\sigma_{predicate}$ **(R)**

- Works on a single relation R and defines a relation that contains only those tuples (rows) of R that satisfy the specified condition (predicate).
- **Selects rows that satisfy selection condition**.
- **No duplicates** in result!
- **Schema of result identical to schema of input relation**.
- Selection is distributive over binary operators
- Selection is **commutative**

Example of Selection

- List all staff with a salary greater than £10,000.

$\sigma_{salary > 10000}$ (Staff)

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24- Mar-58	18000	B003
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003

Figure: 1.1:Example of Selection

Projection

- $\Pi_{col1, \ldots, coln}(R)$
  - Works on a single relation R and defines a relation that contains a vertical subset of R, extracting the values of specified attributes and eliminating duplicates.
  - Deletes attributes that are not in projection list.
  - **Schema of result contains exactly the fields in the projection list, with the same names that they had in the input relation**.
  - **Projection operator has to eliminate duplicates!**

Note: real systems typically don't do duplicate elimination unless the user explicitly asks for it (by DISTINCT). Why not?

Example of Projection

- Produce a list of salaries for all staff, showing only staffNo, fName, lName, and salary details.

$$\Pi_{staffNo,\ fName,\ lName,\ salary}(Staff)$$

staffNo	fName	lName	salary
SL21	John	White	30000
SG37	Ann	Beech	12000
SG14	David	Ford	18000
SA9	Mary	Howe	9000
SG5	Susan	Brand	24000
SL41	Julie	Lee	9000

Figure:1.2 : Example of Projection

Union

- R ∪ S
  - Union of two relations R and S defines a relation that contains all the **tuples of R, or S, or both R and S**, **duplicate tuples being eliminated**.
  - R and S must be **union-compatible.**

- If R and S have *I* and *J* tuples, respectively, union is obtained by concatenating them into one relation with a maximum of (*I* + *J*) tuples.

Example of Union

- List all cities where there is either a branch office or a property for rent.

$$\Pi_{city}(\text{Branch}) \cup \Pi_{city}(\text{PropertyForRent})$$

city
London
Aberdeen
Glasgow
Bristol

Figure:1.3:Example of Union

Set Difference

- R – S
  - Defines a relation consisting of the **tuples that are in relation R, but not in S.**
  - R and S must be **union-compatible.**

Example of Set Difference

- List all cities where there is a branch office but no properties for rent.

$$\Pi_{city}(Branch) - \Pi_{city}(PropertyForRent)$$

city
Bristol

Figure:1.4:Example of Set Difference

Intersection

- $R \cap S$
  - Defines a relation consisting of the set of all tuples that are **in both R and S.**
  - R and S must be **union-compatible.**

- Expressed using basic operations:

  $\mathbf{R \cap S = R - (R - S)}$

Example of Intersection

- List all cities where there is both a branch office and at least one property for rent.

$$\Pi_{city}(Branch) \cap \Pi_{city}(PropertyForRent)$$

city
Aberdeen
London
Glasgow

Figure1.5:Example of Intersection

Cartesian Product

- R X S
  - Defines a relation that is the **concatenation of every tuple of relation "R with every tuple of relation S"**

Example of Cartesian Product

- List the names and comments of all clients who have viewed a property for rent.

  $(\Pi_{clientNo, fName, lName}(Client)) \times (\Pi_{clientNo, propertyNo, comment}(Viewing))$

client.clientNo	fName	lName	Viewing.clientNo	propertyNo	comment
CR76	John	Kay	CR56	PA14	too small
CR76	John	Kay	CR76	PG4	too remote
CR76	John	Kay	CR56	PG4	
CR76	John	Kay	CR62	PA14	no dining room
CR76	John	Kay	CR56	PG36	
CR56	Aline	Stewart	CR56	PA14	too small
CR56	Aline	Stewart	CR76	PG4	too remote
CR56	Aline	Stewart	CR56	PG4	
CR56	Aline	Stewart	CR62	PA14	no dining room
CR56	Aline	Stewart	CR56	PG36	
CR74	Mike	Ritchie	CR56	PA14	too small
CR74	Mike	Ritchie	CR76	PG4	too remote
CR74	Mike	Ritchie	CR56	PG4	
CR74	Mike	Ritchie	CR62	PA14	no dining room
CR74	Mike	Ritchie	CR56	PG36	
CR62	Mary	Tregear	CR56	PA14	too small
CR62	Mary	Tregear	CR76	PG4	too remote
CR62	Mary	Tregear	CR56	PG4	
CR62	Mary	Tregear	CR62	PA14	no dining room
CR62	Mary	Tregear	CR56	PG36	

Figure: 1.6: Examples of Cartesian Product

Join Operations

- JOINs can be used to **combine tables**
- Join is a **derivative of Cartesian** product.
- Equivalent to performing a Selection, using join predicate as selection formula, over Cartesian product of the two operand relations.
- One of the **most difficult operations** to implement efficiently in an RDBMS and **one reason why RDBMSs have intrinsic performance problems.**

Different Types of Joins

- Various forms of join operation
  - JOIN (Inner Join) : Return rows when there is **at least one match** in both tables
  - LEFT JOIN(Left Outer Join): **Return all rows from the left table**, even if there are no matches in the right table
  - RIGHT JOIN(Right Outer Join): Return all rows from the **right table**, even if there are no matches in the left table
  - FULL JOIN: Return rows **when there is a match in one of the tables**

SQL Inner Join Syntax

```
SELECT column_name(s) FROM table_name1 INNER JOIN table_name2  ON
table_name1.column_name=table_name2.column_name
```

INNER JOIN is the same as JOIN

- **The word "INNER" is optional**

Example: Inner Join

We use the following SELECT statement:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
INNER JOIN Orders
ON Persons.P_Id=Orders.P_Id
ORDER BY Persons.LastName
```

We use the following SELECT statement:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
INNER JOIN Orders
ON Persons.P_Id=Orders.P_Id
ORDER BY Persons.LastName
```

The result-set will look like this:

LastName	FirstName	OrderNo
Hansen	Ola	22456
Hansen	Ola	24562
Pettersen	Kari	77895
Pettersen	Kari	44678

The INNER JOIN keyword return rows when there is at least one match in both tables. If there are rows in "Persons" that do not have matches in "Orders", those rows will NOT be listed.

Figure: 1.7:Example of Inner join

Left Join

- A LEFT OUTER JOIN is one of the JOIN **operations** that allow you to specify a join clause. It preserves the unmatched rows from the first (left) table, joining them with a NULL row in the shape of the second (right) table.

SELECT column_name(s) FROM table_name1 Left Join table_name2 ON

table_name1.column_name=table_name2.column_name

Example: Left Join

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

The "Orders" table:

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	15

We use the following SELECT statement:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
LEFT JOIN Orders
ON Persons.P_Id=Orders.P_Id
ORDER BY Persons.LastName
```

The result-set will look like this:

LastName	FirstName	OrderNo
Hansen	Ola	22456
Hansen	Ola	24562
Pettersen	Kari	77895
Pettersen	Kari	44678
Svendson	Tove	

The LEFT JOIN keyword returns all the rows from the left table (Persons), even if there are no matches in the right table (Orders).

Figure:1.8:Example of Left Join

Right Join

- The **RIGHT JOIN** keyword returns all records from the **right** table (table2), and the matched records from the left table (table1). The result is NULL from the left side, when there is no match.

SELECT column_name(s) FROM table_name1 Right Join table_name2 ON table_name1.column_name=table_name2.column_name

Example: Right Join

We use the following SELECT statement:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
RIGHT JOIN Orders
ON Persons.P_Id=Orders.P_Id
ORDER BY Persons.LastName
```

The result-set will look like this:

LastName	FirstName	OrderNo
Hansen	Ola	22456
Hansen	Ola	24562
Pettersen	Kari	77895
Pettersen	Kari	44678
		34764

The RIGHT JOIN keyword returns all the rows from the right table (Orders), even if there are no matches in the left table (Persons).

SQL RIGHT JOIN Example

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

The "Orders" table:

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	15

Now we want to list all the orders with containing persons - if any, from the tables above.

Figure:1.9:Example of Right Join

Full Join

- In SQL the **FULL** OUTER **JOIN** combines the results of both left and right outer **joins**and returns all (matched or unmatched) rows from the tables on both sides of the **join** clause.

Example: Full Join

SQL FULL JOIN Example

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

The "Orders" table:

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	15

Now we want to list all the persons and their orders, and all the orders with their persons.

We use the following SELECT statement:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
FULL JOIN Orders
ON Persons.P_Id=Orders.P_Id
ORDER BY Persons.LastName
```

The result-set will look like this:

LastName	FirstName	OrderNo
Hansen	Ola	22456
Hansen	Ola	24562
Pettersen	Kari	77895
Pettersen	Kari	44678
Svendson	Tove	
		34764

The FULL JOIN keyword returns all the rows from the left table (Persons), and all the rows from the right table (Orders). If there are rows in "Persons" that do not have matches in "Orders", or if there are rows in "Orders" that do not have matches in "Persons", those rows will be listed as well.

Figure:1.10:Example of full Join

Relational Calculus

- There are two versions of the relational calculus:
  - **Tuple relational calculus (TRC)**
  - **Domain relational calculus (DRC)**
- The calculus is non-procedural ('declarative') compared to the relational algebra
- Calculus has variables, constants, comparison ops, logical connectives and quantifiers.
  - TRC: Variables range over (i.e., get bound to) tuples.
  - DRC: Variables range over domain elements (= field values).
  - Both TRC and DRC are simple subsets of first-order logic.
- Expressions in the calculus are called formulas. An answer tuple is essentially an assignment of constants to variables that make the formula evaluate to true.

Tuple Relational Calculus

- Query: {T | P(T)}
  - T is tuple variable
  - P(T) is a formula that describes T
- Result, the set of all tuples t for which P(t) evaluates True.
  - Find all sailors with a rating above 7.
- Atomic formula
  - R.a op S.b , op is one of
  - R.a op constant

Domain Relational Calculus

- A logical language with variables ranging over tuples: { T | Cond }
  - Return all tuples T that satisfy the condition Cond.
- { T | R(T) }: returns all tuples T such that T is a tuple in relation R.
- { T.name | F ACULT Y (T) AND T.DeptId = 'CS' }.

  returns the values of name field of all faculty tuples with the value 'CS' in their department id field.

– The variable T is said to be free since it is not bound by a quantifier (for all, exists).

– The result of this statement is a relation (or a set of tuples) that correspond to all possible ways to satisfy this statement.

– Find all possible instances of T that make this statement true.

Assessment Pattern

S.No.	Item	Number/semester	Marks	System
1	MSTs	2	24 (12 each)	Combined tests
2	Quiz	1	4	Once online
3	Surprise test	1	3	Teacher decides
4	Assignments	3 (one per unit)	4	By teacher as per the dates specified
5	Tutorials	Depending on classes	3	In tutorial classes
6	Attendance	Above 90%	2	
Internal (division as mentioned above points 1-6)			**40**	
External			**60**	
Total			**100**	

REFERENCES

- “Data Structures and Algorithms Made Easy: Data Structures and Algorithmic Puzzles” by Narasimha Karumanchi
- “Data Structures Using C” by Aaron M. Tenenbaum
- “Data Structures and Algorithms” by Alfred V. Aho
- “Fundamentals of Data Structures in C by Ellis Horowitz, Sartaj Sahni, and Susan Anderson-Freed

THANK YOU

For queries

Email: Heenae7725@cumail.in