DBMS

Dr. Media A. Ibrahim
Assist.Lec: Areen Hamad
Software Engineering
ISE Department
Media.ibrahim@epu.edu.iq
Areen.hamad@epu.edu.iq

2024-2025

Database Query, Relational Query Languages

Outlines

- Query Languages .
- Introduction To SQL.
- The SQL Environment.
- Relational Algebra.
- Five fundamental operations in the relational algebra.

Query Languages

- A query language is a language in which a database user requests information from the database.
- Query languages can be broadly categorized into two groups: procedural languages and nonprocedural languages.
- Procedural language:

  Ex: relational algebra

- Nonprocedural languages

  Ex: SQL

What is SQL?

- is a standard query language for relational databases
- It support logical data model concepts, such as relations, keys, ...
- Supported by major brands, e.g. , Oracle, MS SQL Server, Sybase, ...
- 3 versions: SQL1 (1986), SQL2 (1992), SQL 3 (1999)
- Can express common data intensive queries

SQL Environment

- **Catalog**
  - A set of schemas that constitute the description of a database
- **Schema**
  - The structure that contains descriptions of objects created by a user (base tables, views, constraints)
- **Data Definition Language (DDL)**
  - Commands that define a database, including creating, altering, and dropping tables and establishing constraints
- **Data Manipulation Language (DML)**
  - Commands that maintain and query a database
- **Data Control Language (DCL)**
  - Commands that control a database, including administering privileges and committing data

SQL Environment(cont.)

- **Data Definition Language (DDL)**

  CREATE TABLE tablename to create a table in the database

  DROP TABLE tablename to remove a table from the database

  ALTER TABLE tablename to add or remove columns from a table in the database

  CREATE INDEX

- **Data Manipulation Language (DML)**

  SELECT to select rows of data from a table
  INSERT to insert rows of data into a table
  UPDATE to change rows of data in a table
  DELETE to remove rows of data from a table

- **Data Control Language (DCL)**

  GRANT to grant a privilege to a user
  REVOKE to revoke (remove) a privilege from a user

```sql
CREATE TABLE Students (
    ID INT PRIMARY KEY,
    Name VARCHAR(50),
    Age INT
);
```

Example:

Select all students older than 18:

```sql
SELECT * FROM Students WHERE Age > 18;
```

SQL Joins

INNER JOIN: Returns rows with matching values in both tables.

LEFT JOIN: Returns all rows from the left table and matched rows from the right table.

RIGHT JOIN: Returns all rows from the right table and matched rows from the left table.

FULL OUTER JOIN: Returns rows when there is a match in either table.

Example:

Retrieve all students and their corresponding courses using a join:

```sql
SELECT Students.Name, Courses.CourseName
FROM Students
INNER JOIN Enrollments ON Students.ID = Enrollments.StudentID
INNER JOIN Courses ON Enrollments.CourseID = Courses.CourseID;
```

Aggregate Functions

COUNT(): Returns the number of rows.

SUM(): Returns the sum of a numeric column.

AVG(): Returns the average value.

GROUP BY: Groups rows that share a common attribute.

Example:
Calculate the average age of students:

```sql
SELECT AVG(Age) FROM Students;
```

Relational Algebra

- The *relational algebra* is a procedural query language. It consists of set operations which are either unary or binary, meaning that either one or two relations are operands to the set operations.
- Each of the set operations produces a relation as its output.
- There are five fundamental operations in the relational algebra and several additional operations which are defined in terms of the five fundamental operations.
- There is also a *rename* operation which is sometimes referred to as a fundamental operation.
- The five fundamental operations are: *select, project, union, set difference,* and *Cartesian product*.

Selection Operator

Type: unary
Symbol: Greek letter sigma, σ
General form: $\sigma_{(\text{predicate})}(\text{relation instance})$
Schema of result relation: same as operand relation
Size of result relation (tuples): $\leq |\text{operand relation}|$
Examples:
$\sigma_{(\text{major = "CS"})}(\text{students})$
$\sigma_{(\text{major = "CS" and hair-color = "brown"})}(\text{students})$
$\sigma_{(\text{hours-attempted > hours-earned})}(\text{students})$

- The select operation selects tuples from a relation instance which satisfy a specified predicate.
- In general, a predicate, may contain any of the logical comparative operators, which are =, ≠, <, ≤, >, ≥. Furthermore, several predicates may be combined using the connectives *and* (∧), *or* (∨), and *not* (¬).

Selection Operator Examples

R

A	B	C	D
a	a	yes	1
b	d	no	7
c	f	yes	34
a	d	no	6
a	c	no	7
b	b	no	69
c	a	yes	24
d	d	yes	47
h	d	yes	34
e	c	no	26
a	a	yes	5

$r = \sigma_{(A = \text{'a'})}(R)$

A	B	C	D
a	a	yes	1
a	d	no	6
a	c	no	7
a	a	yes	5

Projection Operator

Type: unary
Symbol: Greek letter pi, π
General form: $\pi_{(\text{attribute-list})}(\text{relation instance})$
Schema of result relation: specified by <attribute-list>
Size of result relation (tuples): $\leq |$ operand relation $|$
Examples:
$\pi_{(\text{student-id, name, major})}(\text{students})$
$\pi_{(\text{name, advisor})}(\text{students})$
$\pi_{(\text{name, gpa, hours-attempted})}(\text{students})$

- The project operation can be viewed as producing a vertical cross-section of the operand relation.
- If the operation produces duplicate tuples, these are typically removed from the result relation in keeping with its set-like characteristics.

Projection Operator Examples

R

A	B	C	D
a	a	yes	1
b	d	no	7
c	f	yes	34
a	d	no	6
a	c	no	7
b	b	no	69
c	a	yes	24
d	d	yes	47
h	d	yes	34
e	c	no	26
a	a	yes	5

$r = \pi_{(A, C)}(R)$

A	C
a	yes
b	no
c	yes
a	no
d	yes
h	yes
e	no

Union Operator

Type: binary
Symbol: union symbol, $\cup$
General form: $r \cup s$, where *r* and *s* are union compatible
Schema of result relation: schema of operand relations
Size of result relation (tuples): $\leq \max\{|r|+|s|\}$
Examples:
$r \cup s$ $\quad\quad$ $\pi_{(a, b)}(r) \cup \pi_{(a, b)}(s)$

- The union operation provides a means for extracting information which resides in two operand relations which must be *union compatible*. Union compatibility requires that two conditions hold:
  1. Relations *r(R)* and *s(S)* in the expression $r \cup s$ must be of the same degree (*arity)*. That is, they must have the same number of attributes.
  2. The domains of the *i*th attribute of *r(R)* and the *i*th attributes of *s(S)* must be the same, for all *i*.

Union Operator Examples

T

A	B
a	a
b	d
c	f
a	d
a	c

X

A	B
a	a
b	d
a	c

r = T ∪ X

A	B
a	a
b	d
c	f
a	d
a	c

Set Difference Operator

Type: binary
Symbol: −
General form: r − s, where *r* and *s* are union compatible
Schema of result relation: schema of operand relation
Size of result relation (tuples): ≤ | relation r |
Examples: r − s

- The set difference operation allows for the extraction of information contained in one relation that is not contained in a second relation.
- As with the union operation, the set difference operation requires that the two operand relations be union compatible.

Set Difference Operator Examples

R

A	B	D
a	a	1
b	d	7
c	f	34
a	d	6
a	c	7

S

X	Y	Z
a	m	4
b	c	22
a	d	16
a	c	7

r = R − S

E	F	G
a	a	1
b	d	7
c	f	34
a	d	6

Cartesian Product Operator

Type: binary
Symbol: ×
General form: r × s (no restrictions on *r* and *s*)
Schema of result relation: schema *r* × schema *s* with renaming
Size of result relation (tuples): > | relation *r* | and > | relation *s* |
Examples:
r × s

- The Cartesian product operation allows for the combining of any two relations into a single relation.
- Recall that a relation is by definition a subset of a Cartesian product of a set of domains, so this gives you some idea of the behavior of the Cartesian product operation.

Cartesian Product Operator Examples

T

A	B
a	a
b	d

X

A	B
a	a
b	d
a	c
c	a

r = T × X

T.A	T.B	X.A	X.B
a	a	a	a
a	a	b	d
a	a	a	c
a	a	c	a
b	d	a	a
b	d	b	d
b	d	a	c
b	d	c	a

Intersection Operator

Type: binary
Symbol: ∩
General form: r ∩ s where *r* and *s* are union compatible relations
Schema of result relation: schema of operation relation
Size of result relation (tuples): ≤ |r|
Definition: r ∩ s ≡ r − (r − s)
Example:

$$(\pi_{(p\#)}(SPJ)) \cap (\pi_{(p\#)}(P))$$

- The intersection operation produces the set of tuples that appear in **both** operand relations.

Intersection Operator Examples

R

A	B	C	D
a	a	yes	1
b	d	no	7
c	f	yes	34
a	d	no	6

S

E	F	G	H
a	a	yes	1
b	r	yes	3
c	f	yes	34
m	n	no	56

$r = R \cap S$

A	B	C	D
a	a	yes	1
c	f	yes	34

Join Operators

- As we saw in some of our earlier query expression which involved the Cartesian product operator, we had to provide additional selection operations to remove those combinations of tuples that resulted from the Cartesian product which weren't related (they didn't make sense like when a shipment of a specific part was combined with part information but the part information didn't belong to the part that was being shipped).
- This occurs so commonly that an operation which is a combination of the Cartesian product and selection operations was developed called a *join operation*.
- There are several different join operations which are called, *theta-join, equijoin, natural join, outer join,* and *semijoin*. We will examine each of these operations and explore the conditions of their use.

Theta-Join and Equijoin Operators

Type: binary
Symbol/general form: $r \bowtie_{(predicate)} s$
Schema of result relation: concatenation of operand relations
Definition: $r \bowtie_{(predicate)} s \equiv \sigma_{(predicate)}(r \times s)$
Examples:

$r \bowtie_{(color='blue' \text{ AND } size=3)} s$ an equijoin

$r \bowtie_{(color='blue' \text{ AND } size>3)} s$ a theta-join

- The theta-join operation is a shorthand for a Cartesian product followed by a selection operation.
- The equijoin operation is a special case of the theta-join operation that occurs when all of the conditions in the predicate are equality conditions.

Theta-Join Operator Examples

R

A	B	C	D
a	a	yes	1
b	d	no	7
c	f	yes	34
a	d	no	6

S

E	F	G	H
a	a	yes	1
b	r	yes	3
c	f	yes	34
m	n	no	56

$r = R \bowtie_{(R.B < S.F)} S$

A	B	C	D	E	F	G	H
a	a	yes	1	b	r	yes	3
a	a	yes	1	c	f	yes	34
a	a	yes	1	m	n	no	56
b	d	no	7	b	r	yes	3
b	d	no	7	c	f	yes	34
b	d	no	7	m	n	no	56
c	f	yes	34	b	r	yes	3
c	f	yes	34	m	n	no	56
a	d	no	6	b	r	yes	3
a	d	no	6	c	f	yes	34
a	d	no	6	m	n	no	56

Natural Join Operator

Type: binary

Symbol/general form: $r * s$

Schema of result relation: concatenation of operand relations with only one occurrence of commonly named attributes

Definition: $r * s \equiv r \; (r.commonattributes = s.commonattributes) \; s$

Examples: $s * spj * p$

- The natural-join operation performs an equijoin over all attributes in the two operand relations which have the same attribute name.
- The degree of the result relation of a natural-join is sum of the degrees of the two operand relations less the number of attributes which are common to both operand relations. (In other words, one occurrence of each common attribute is eliminated from the result relation.)
- The natural join is probably the most common of all the forms of the join operation. It is extremely useful in the removal of extraneous tuples. Those attributes which are commonly named between the two operand relations are commonly referred to as the *join attributes*.

Natural Join Operator Examples

R

A	B	C	D
a	a	yes	1
b	r	no	7
c	f	yes	34
a	m	no	6

S

B	M	G	H
a	a	yes	1
b	r	yes	3
a	f	yes	34
m	n	no	56

r = R * S

A	B	C	D	M	G	H
a	a	yes	1	a	yes	1
a	a	yes	1	f	yes	34
a	m	no	6	n	no	56

Outer Join Operator

Type: binary

Symbol/general form: left-outer-join: r ⟕ s right-outer-join: r ⟖ s

full outer join: r ⟗ s

Schema of result relation: concatenation of operand relations

Definition:

r ⟕ s ≡ natural join of r and s with tuples from r which do not have a match in s included in the result. Any missing values from s are set to null.

r ⟖ s ≡ natural join of r and s with tuples from s which do not have a match in r included in the result. Any missing values from r are set to null.

r ⟗ s ≡ natural join of r and s with tuples from both r and s which do not have a match are included in the result. Any missing values are set to null.

Examples: Let r(A,B) = {(a, b), (c, d), (b,c)} and let

s(A,C) = {(a, d), (s, t), (b, d)}

then r ⟕ s = (A,B,C) = {(a,b,d), (b,c,d), (c,d,null)},

r ⟖ s = (A,B,C) = {(a,b,d), (b,c,d), (s,null,t)}, and

r ⟗ s = (A,B,C) = {(a,b,d), (b,c,d), (s,null,t), (c,d,null)},

Outer Join Operator Examples

R

A	B	C
1	2	3
4	5	6
7	8	9

$r = R \supset\!\!\!\!\subset S$

A	B	C	D
1	2	3	10
1	2	3	11
4	5	6	null
7	8	9	null
null	6	7	12

S

B	C	D
2	3	10
2	3	11
6	7	12

$r = R \supset\!\!\!\subset S$

A	B	C	D
1	2	3	10
1	2	3	11
4	5	6	null
7	8	9	null

$r = R \subset S$

B	C	D	A
2	3	10	1
2	3	11	1
6	7	12	null

Semi Join Operator

Type: binary
Symbol/general form: $r \ltimes_{(predicate)} s$
Schema of result relation: schema of r
Definition: $r \ltimes_{(predicate)} s \equiv \pi_{(attributes\ of\ r)}(r \bowtie_{(predicate)} s)$
Examples: see next page

- The semi-join operation performs a join of the two operand relations and then projects over the attributes of the left-hand operand relation.
- The primary advantage of the semi-join operation is that it decreases the number of tuples that need to be handled to form the join. This is particularly useful in a distributed environment.
- In its general form, which is shown above, the semi-join is a semi-theta-join. The expected variants of a semi-equijoin and a semi-natural-join are defined in a similar fashion.

Semi Join Operator Examples

R

A	B	C	D
a	A	yes	1
B	r	no	7
c	f	yes	34
a	m	no	6

S

B	M	C
a	e	yes
b	r	yes
a	f	no
r	n	no

$r = R \ltimes_{(R.B > S.M)} S$

A	B	C	D
b	r	no	7
c	f	yes	34
a	m	no	6

Division Operator

Type: binary
Symbol/general form: r ÷ s where r({A}) and s({B})
Schema of result relation: C where C = A − B
Definition: $r \div s \equiv \pi_{(A-B)}(r) - (\pi_{(A-B)}((\pi_{(A-B)}(r) \times s) - r))$
Examples:
Let r(A,B,C) = {(a,b,c), (a,d,d), (a,b,d), (a,c,c), (a,d,d)}
and s(C) = {(c), (d)}
then: r ÷s = t(A,B) = {(a,b)}

Requirements for the division operation:

1. Relation r is defined over the attribute set A and relation s is defined over the attribute set B such that $B \subseteq A$.
2. Let C be the set of attributes in $A - B$.

Given these constraints the division operation is defined as: a tuple t is in $r \div s$ if for every tuple t_s in s there is a tuple t_r in r which satisfies both:

$t_r[C] = t_s[C]$ and $t_r[A-B] = t[A-B]$

Division Operator Examples

R

A	B	C	D
a	f	yes	1
b	r	no	1
a	f	yes	34
e	g	yes	34
a	m	no	6
b	r	no	34

S

D
1
34

$r = R \div S$

A	B	C
a	F	yes
b	r	no

Conclusion

- A query is a “question” posed to a database
- Queries are expressed in a high-level declarative manner

 Examples:

- Mouse click on a map symbol (e.g. road) may mean
- Typing a keyword in a search engine (e.g. google, yahoo) means

Exercise:

- Write a query to:

  1-Select students who are older than 21.

  Increase the salary of employees in a specific department by 10%.

  2-Create a transaction that transfers money between two bank accounts and handles failures.