# Relational Algebra

## SWEN 304

## Trimester 2, 2017


**Lecturer: Dr Hui Ma**

Engineering and Computer Science

TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

**slides by: Pavle Morgan & Hui Ma**

# Secrets to Success (IPENZ)

When? Wednesday 9th August at 5:30pm to 7:30pm

Where? GBLT2, Old Government Building (Opposite Beehive), Pipitea Campus

Register at
**ipe.nz/secrets-to-success**

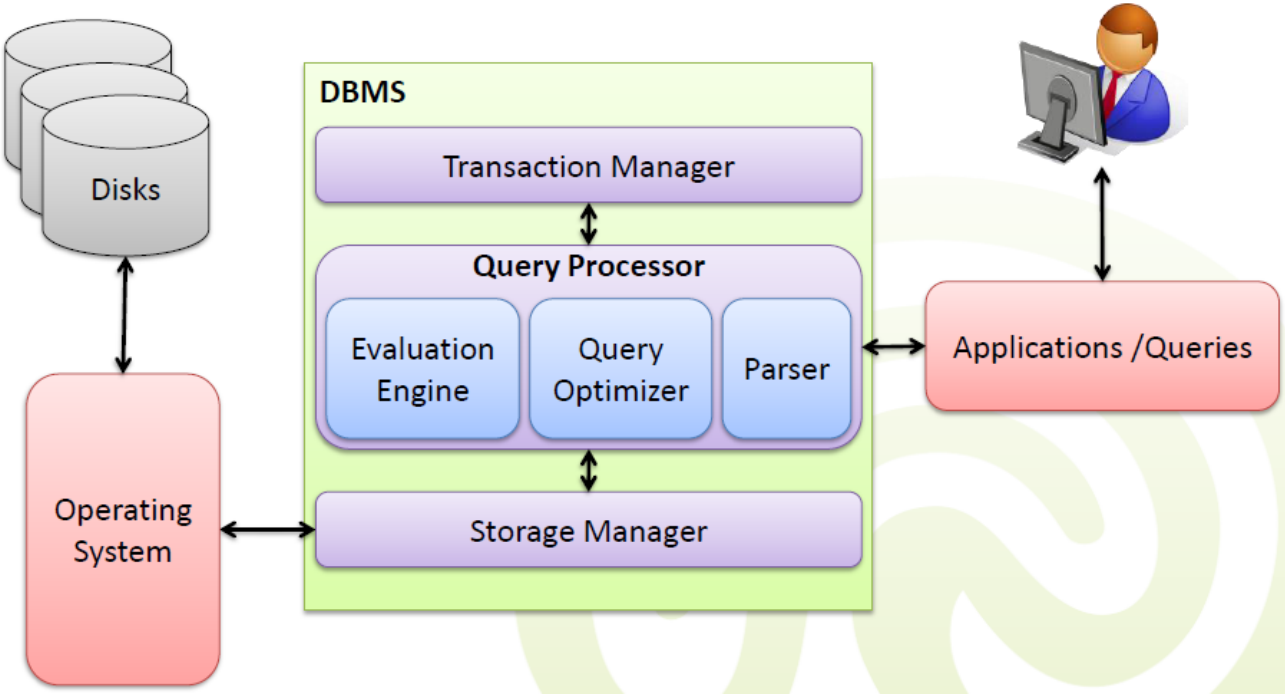
# Outline

- Basic relational algebra operations
- Set theoretic operations
- Additional operations

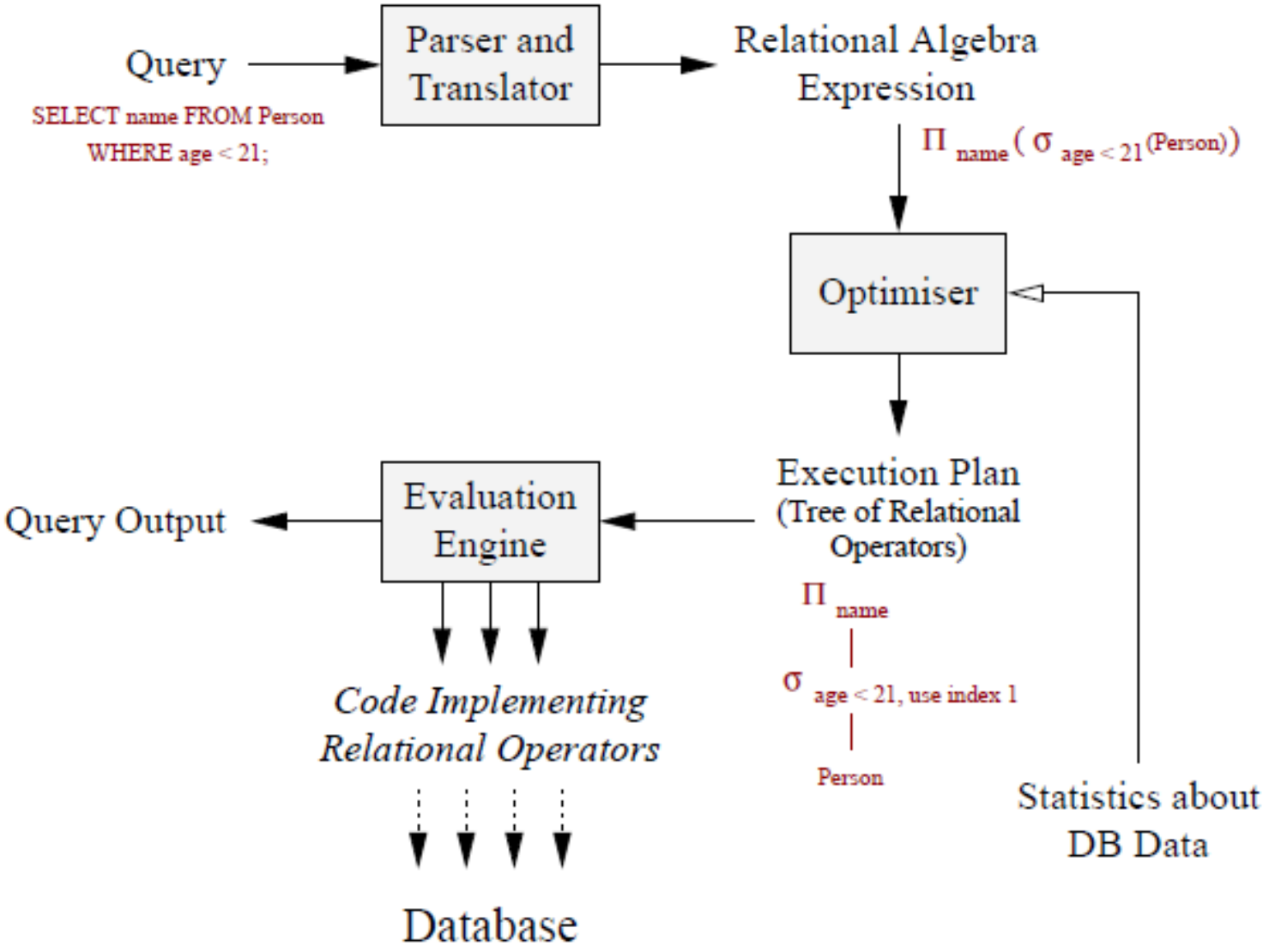
- *Reading: Chapters 6 of the textbook*

# Query Processing in DBMS

- Users/applications submit queries to the DBMS
- The DBMS processes queries before evaluating them
  - Recall: DBMS mainly use declarative query languages (such as SQL)
  - Queries can often be evaluated in different ways
  - SQL queries do not determine how to evaluate them

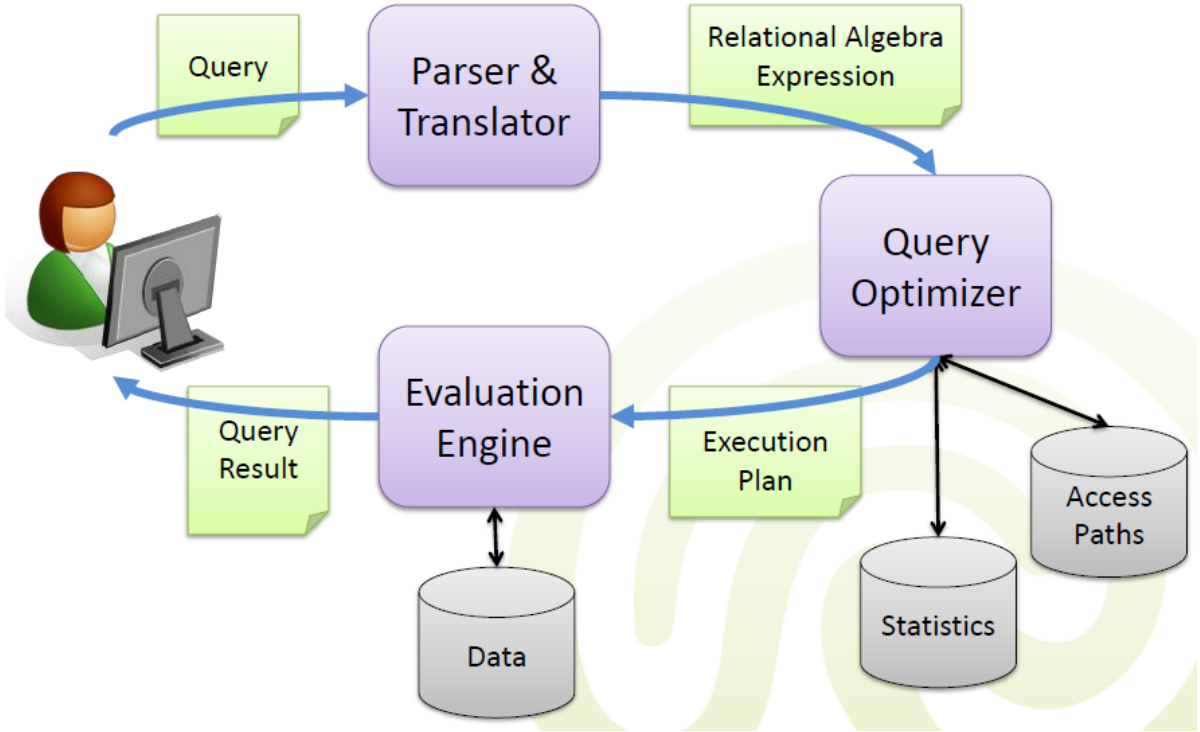


# Query Processing in DBMS[4,5]



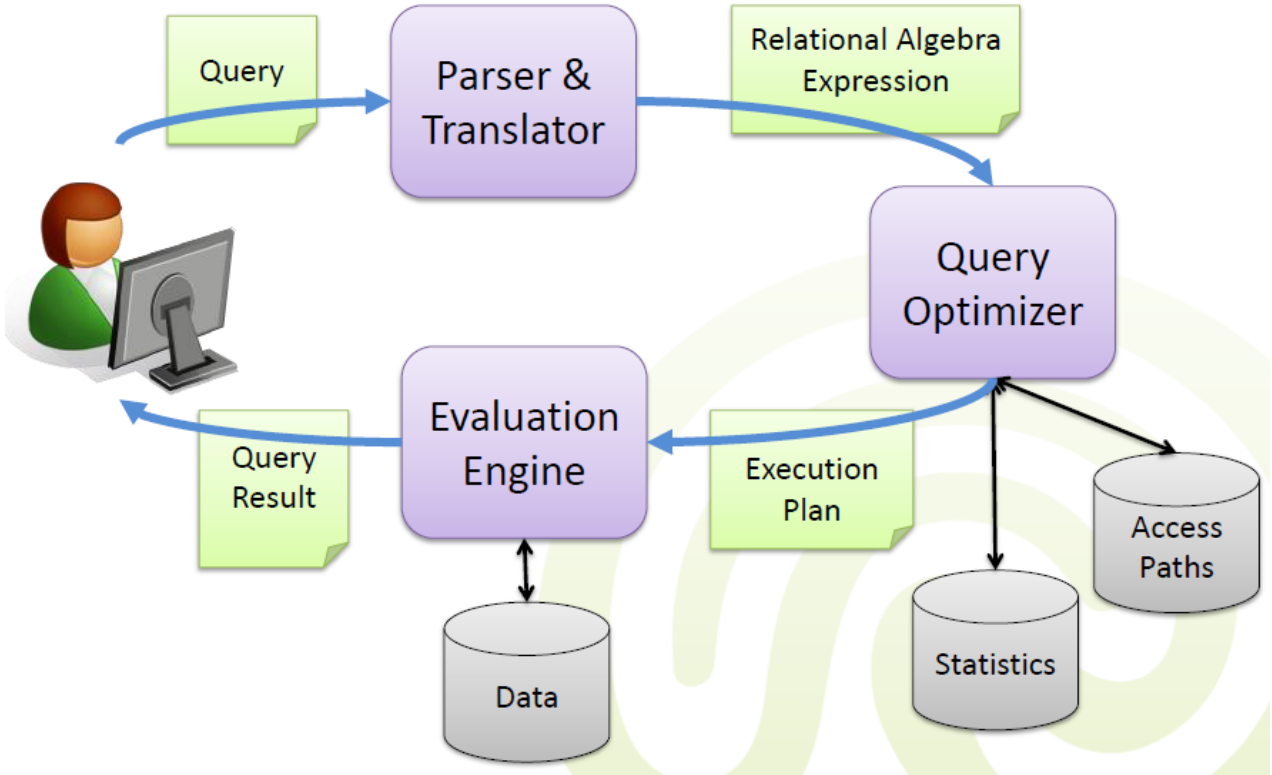
# Query Processing in DBMS

- The parser checks the syntax, e.g., verifies table names, data types
  - A scanner tokenizes the query (tokens for SQL commands, names, ...)
  - Either the query is executable or an error message is generated
    - (SQLCODE/SQLSTATE)



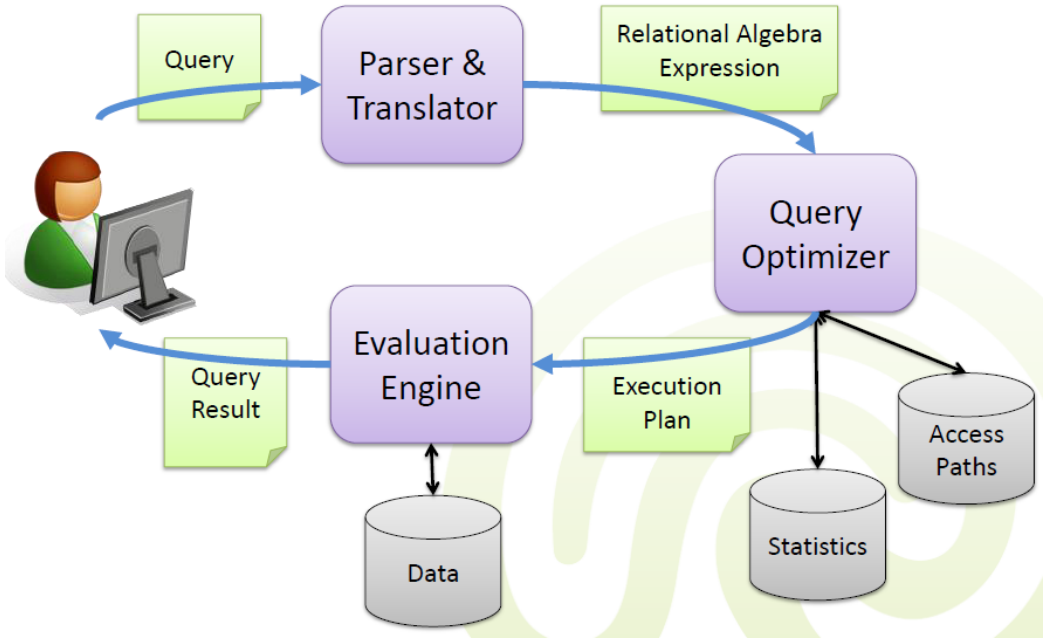
# Query Processing in DBMS

- The translator translates the query into relational algebra
  - Internal exchange format between DBMS components
  - Allows for symbolic calculation



# Query Processing in DBMS

- Relational Algebra was introduced by Codd (1970) with the relational data model
  - Provides formal foundations for relational model operations
  - Used as basis for implementing and optimizing queries in RDBMSs
  - Some of the concepts are incorporated into the SQL standard query language



# Relational Algebra

- A set of operations to manipulate (query and update) a relational database
  - Operations are applied onto relations
  - The result is a new relation
- Basic operations:
  - project , select, rename, and join
- Set theoretic operations:
  - union, intersect, set difference,
  - Cartesian product
- Additional relational operations:
  - aggregate operations (SUM, COUNT, AVERAGE), grouping, and
  - outer join

# A Sample Relational Database

Student

| Lname | Fname | StudId | Major |
|---|---|---|---|
| Smith | Susan | 131313 | Comp |
| Bond | James | 007007 | Math |
| Smith | Susan | 555555 | Comp |
| Cecil | John | 010101 | Math |

Course

| Pname | CourId | Points | Dept |
|---|---|---|---|
| DB Systems | C302 | 15 | Comp |
| Software Engineering | C301 | 15 | Comp |
| Discrete Math | M214 | 22 | Math |
| Programmes | C201 | 22 | Comp |

Grades

| StudId | CourId | Grad |
|---|---|---|
| 007007 | C302 | A+ |
| 555555 | C302 | ω |
| 007007 | C301 | A |
| 007007 | M214 | A+ |
| 131313 | C201 | B- |
| 555555 | C201 | C |
| 131313 | C302 | ω |
| 007007 | C201 | A |
| 010101 | C201 | ω |

# Project Operation

- Notation: $\pi_{AL}(R)$

  where $AL$ is an attribute sublist from $R$

- Project operation produces a new relation by retaining columns in $AL$ and dropping all the others
- If $AL = (A_1, \ldots, A_k)$, then $\pi_{AL}(R) = R[A_1, \ldots, A_k]$

- Example: $\pi_{\text{LName, FName}}(\text{Student})$:

StudentName

| LName | FName |
|---|---|
| Smith | Susan |
| Bond | James |
| Cecil | John |

# Select Operation

- select such a subset of tuples from a relation that satisfies a given condition
- Notation: $\sigma_c(R)$
  - Condition $c$ is a Boolean expression on attributes of $R$
  - Boolean expression is made up of clauses of the form $A\theta\, a$ or $A\ \theta\ B$, where
    - $a \in dom(A)$,
    - $\theta \in \{=, <, >, \leq, \geq, \neq\}$, and
    - $A, B \in R$
  - Clauses can be connected by Boolean operators $\neg$, $\wedge$, $\vee$ to form new clauses

# Select Operation: Examples

- $\sigma_{StudId = 007007}(Student)$

Student2

| LName | FName | StudId | Major |
|---|---|---|---|
| Bond | James | 007007 | Math |

$\sigma_{FName = 'Susan'}(Student)$

Student3

| LName | FName | StudId | Major |
|---|---|---|---|
| Smith | Susan | 131313 | Comp |
| Smith | Susan | 555555 | Comp |

# Numeric Properties of Select and Project

- Since we want to use relational algebra expressions in query optimization, we need numeric properties of relational algebra operations
- Relation $\pi_{AL}(R)$ is produced from $R$ by retaining columns in $AL$ and dropping duplicate tuples, hence:
  - $degree(\pi_{AL}(R)) = |AL| \leq |R|$ (number of attributes)
  - $|\pi_{AL}(R)| \leq |R|$ (number of tuples)
- Relation $\sigma_C(R)$ contains those tuples of $R$ that evaluate true for $C$, hence:
  - $degree(\sigma_C(R)) = degree(R)$ (number of attributes)
  - $\sigma_C(R) \subseteq R$ and $|\sigma_C(R)| \leq |R|$ (number of tuples)

# Combining Select and Project Operators

- $\pi_{AL}(\sigma_C(R))$ or $\sigma_C(\pi_{AL}(R))$
- For example,

$$\pi_{\text{FName, LName}}(\sigma_{\text{StudId = 007007}}(\text{Student}))$$

In SQL:

```
SELECT FName, LName
FROM Student
WHERE  StudentId = 007007;
```

Student4

| FName | LName |
|---|---|
| James | Bond |

# Rename Operation

- Notation: $\delta_{A1 \rightarrow B1,\ldots,Ak \rightarrow Bk}(R)$
  with $dom(B_i) = dom(A_i)$ for $i = 1, \ldots, k$
- A unary operation defined on relations $R$ with $A_1,\ldots,A_k \in R$
- schema: $(R - \{A_1, \ldots, A_k\}) \cup \{B_1, \ldots, B_k\}$
- Example: $\delta_{\text{FName} \rightarrow \text{FirstName,LName} \rightarrow \text{LastName}}(\text{Student4})$
- In SQL:

```
SELECT FName AS FirstName, LName AS LastName
FROM Student4;
```

Student5

| FirstName | LastName |
|---|---|
| James | Bond |

# Join Operation

- Join operation merges those tuples from two relations that satisfy a given condition
  - The condition is defined on attributes belonging to both of the relations to be joined
- Theta, equi, and natural join operations
- Theta, equi, and natural join are collectively called INNER joins
- In each of inner joins, tuples with null valued join attributes do not appear in the result
- OUTER joins include tuples with null valued join attributes into the result

# Theta Join Operation

- Notation: $R = R_1 \bowtie_{JC} R_2$
  - $R$ is the result of joining $R_1$ with $R_2$
  - Join condition $JC = jc_1 \wedge \ldots \wedge jc_n$
    - $jc_i = A \theta B,\ A \in R_1,\ B \in R_2,$
    - $\theta \in \{=, \neq, <, >, \leq, \geq\},$
    - $Dom(N_1, A) \subseteq Dom(N_2, B),$
    - $Range(N_1, A) \subseteq Range(N_2, B)$
  - $R_1 = \{A_1, \ldots, A_m\},\ R_2 = \{B_1, \ldots, B_n\},$
    $R = \{A_1, \ldots, A_m, B_1, \ldots, B_n\}$
  - $degree(R) = degree(R_1) + degree(R_2)$
  - $| r(R) | \leq | r(R_1) | \times | r(R_2) |$

# Equijoin Operation

- A special case of the theta join, when $\theta \in \{=\}$
  - Notation: $R = R_1 \bowtie_{JC} R_2$

  where $JC = jc_1 \wedge \ldots \wedge jc_n$

  $jc_i \equiv A = B,\ A \in R_1,\ B \in R_2,$
- For example,

  $\text{Student} \bowtie_{\text{StudId = StudId}} \text{Grades},$

  In SQL:

```
SELECT *
FROM Student s, Grades g
WHERE s.StudId = g.StudId;
```

# Equijoin Operation: Example

Superfluous column

Student_Grades

| Lname | Fname | StudId | StudId | Major | CourId | Grade |
|---|---|---|---|---|---|---|
| Smith | Susan | 131313 | 131313 | Comp | C201 | B- |
| Smith | Susan | 131313 | 131313 | Comp | C302 | ω |
| Bond | James | 007007 | 007007 | Math | C302 | A+ |
| Bond | James | 007007 | 007007 | Math | C301 | A |
| Bond | James | 007007 | 007007 | Math | M214 | A+ |
| Bond | James | 007007 | 007007 | Math | C201 | A |
| Smith | Susan | 555555 | 555555 | Comp | C201 | C |
| Smith | Susan | 555555 | 555555 | Comp | C302 | ω |
| Cecil | John | 010101 | 010101 | Math | C201 | ω |

# Natural Join Operation

- A special case of an equijoin operation, when join attributes have the same name ($R_1.X = R_2.X$)
  - Notation: $R = R_1 * R_2$
  - Formal definition:

$$R_1 * R_2 = \{ t\,[R_1 \cup R_2] \mid t\,[R_1] \in R_1 \wedge t\,[R_2] \in R_2 \}$$

  - $degree(R) = degree(R_1) + degree(R_2) - |X|$ (number of attributes)
  - $0 \leq |R_1 * R_2| \leq |R_1| \bullet |R_2|$ (number of tuples)

# Natural Join Operation: Example

- Query: Retrieve information of students and their grades
- Relational Algebra:

  Student * Grades

- In SQL:

```
SELECT * FROM Student NATURAL JOIN Grades;
```

# Natural Operation: Example

## Student * Grades

| Lname | Fname | StudId | Major | CourId | Grade |
|---|---|---|---|---|---|
| Smith | Susan | 131313 | Comp | C201 | B- |
| Smith | Susan | 131313 | Comp | C302 | ω |
| Bond | James | 007007 | Math | C302 | A+ |
| Bond | James | 007007 | Math | C301 | A |
| Bond | James | 007007 | Math | M214 | A+ |
| Bond | James | 007007 | Math | C201 | A |
| Smith | Susan | 555555 | Comp | C201 | C |
| Smith | Susan | 555555 | Comp | C302 | ω |
| Cecil | John | 010101 | Math | C201 | ω |

# Set Theoretic Operations

- Union, Intersect, Difference, Cartesian product

$$R = R_1 \Theta R_2$$

where $R_1 = (A_1, \ldots, A_n)$, $R_2 = (B_1, \ldots, B_m)$ are lists of attributes, and

$\Theta \in \{ \cup, \cap, -, \times \}$

- i.e.
  - $R = R_1 \cup R_2$
  - $R = R_1 \cap R_2$
  - $R = R_1 \times R_2$
  - $R = R_1 - R_2$

# Set Theoretic Operations

- For union, intersect and difference, attribute sets $R_1$ and $R_2$ have to be union compatible:
  - $|R_1| = |R_2|$,
  - $(\forall i \in \{1, \ldots, n\})(Dom(R_1, A_i) = Dom(R_2, B_i))$, and
  - $(\forall i \in \{1, \ldots, n\})(Range(R_1, A_i) = Range(R_2, B_i))$

- For cartesian product

  $R = R_1 \times R_2$,

  $degree(R_1 \times R_2) = degree(R_1) + degree(R_2)$,

  $|R_1 \times R_2| = |R_1| \bullet |R_2|$

# Question For You

- Consider the following relations

$R_1$

| A | B |
|---|---|
| 1 | 2 |
| 3 | 3 |
| 4 | 4 |

$R_2$

| B | C |
|---|---|
| 2 | 7 |
| 4 | 9 |
| ω | 0 |

- How many tuples will the Cartesian product $R_1 \times R_2$ return?

a) 6

b) 9

# Question For You

- Consider the following relations

$R_1$

| A | B |
|---|---|
| 1 | 2 |
| 3 | 3 |
| 4 | 4 |

$R_2$

| B | C |
|---|---|
| 2 | 7 |
| 4 | 9 |
| ω | 0 |

- How many tuples will the natural join $R_1 * R_2$ return?

a) 2

b) 6

c) 9

# Additional Relational Operations

- To enhance the power of relational algebra, there are some new operations introduced:
  - Aggregate functions (SUM, AVERAGE, MAX, MIN, COUNT),
  - Grouping, and
  - Outer join
- Aggregate functions (except COUNT) are defined on numeric attributes and applied on all tuples of a relation
- Grouping attributes are used in conjunction with aggregate functions, and a defined aggregate function is applied on tuples of each group
- Outer join extends the join operation to cope with null values

# Aggregate Functions and Grouping

- Notation:

$$<\text{grouping attributes}>\mathcal{F}_{<(\text{function, attribute}) \text{ list}>}(r(N))$$

where:

- <grouping attributes> is a list of attributes from $R$,
- $\mathcal{F}$ (pronounced as "script F") is the symbol used to denote aggregate operation, and
- <(function, attribute) list> is a list of pairs (aggregate function from {SUM, AVERAGE, COUNT, MIN, MAX }, attribute from $R$ )

- The resulting relation has columns for grouping attributes, and one column with the name of the form FUNCTION_ATTRIBUTE for each (function, attribute) pair

# Outer Join

- Introduced to include those tuples that don't match, or contain null values for join attributes into join relation
- Notations:

LEFT: ⟕ RIGHT: ⟖ and FULL outer join: ⟗

- Example:

$R_1$

| A | B |
|---|---|
| 1 | 2 |
| 5 | 6 |

⟕ B =C

$R_2$

| C | D |
|---|---|
| 2 | 7 |
| 2 | 9 |
| ω | 1 |

=

$R_1$ ⟕ B =C $R_2$

| A | B | C | D |
|---|---|---|---|
| 1 | 2 | 2 | 7 |
| 1 | 2 | 2 | 9 |
| 5 | 6 | ω | ω |

# Relational Algebra & SQL

- Each relational algebra query (except union) can be easily rewritten in SQL (for simplicity: assume global attribute names)
  - attribute selection $\sigma_{A=B}(R)$:

    SELECT * FROM $R$ WHERE $A = B$;
  - constant selection $\sigma_{A=c}(R)$:

    SELECT * FROM $R$ WHERE $A$ = c;
  - projection $\pi_{A1,\ldots,Ak}(R)$:

    SELECT DISTINCT $A_1, \ldots, A_k$ FROM $R$;

# Relational Algebra & SQL

- rename $\delta_{A1 \rightarrow B1,\ldots,Ak \rightarrow Bk}(R)$:

  SELECT $A_1$ AS $B_1$ ,. . . , $A_k$ AS $B_k$ FROM $R$;

- natural join $R_1 * R_2$ (with common attributes $A_1$, . . . ,$A_k$):

  SELECT * FROM $R_1$ NATURAL JOIN $R_2$;

- equijoin $R_1 \bowtie_{A1=B1,\ldots,Ak=Bk} R_2$:

  SELECT * FROM $R_1$, $R_2$ WHERE $R_1.A_1 = R_2.B_1$ AND . . . AND $R_1.A_k = R_2.B_k$;

- difference $R_1 - R_2$:

  SELECT * FROM $R_1$ EXCEPT SELECT * FROM $R_1$ ;

# Relational Algebra and SQL: Examples

- Project operation:
  - $\pi_{\text{LName, FName}}$ (Student)
  - SELECT DISTINCT LName, FName FROM Student;
- Selection operation:
  - $\sigma_{\text{FName = 'Susan'}}$ (Student)
  - SELECT * FROM Student WHERE FName = 'Susan';

# Summary

- Relational Algebra consists of several groups of operations
  - Unary Relational Operations
    - SELECT (symbol: $\sigma$ (sigma))
    - PROJECT (symbol: $\pi$ (pi))
    - RENAME (symbol: $\delta$ (delta))
  - Binary Relational Operations
    - JOIN (several variations of JOIN exist)
  - Relational Algebra Operations From Set Theory
    - UNION ( $\cup$ ), INTERSECTION ( $\cap$ ), DIFFERENCE (or MINUS, – )
    - CARTESIAN PRODUCT ( x )
  - Additional Relational Operations
    - OUTER JOINS,
    - AGGREGATE FUNCTIONS (These compute summary of information: for example, SUM, COUNT, AVG, MIN, MAX)

# References


1. Elmasri, Navathe. Fundamentals of database systems. Pearson, 2010
2. Ramakrishnan, Gehrke. Database Management Systems. McGraw-Hill, 2003
3. Silberschatz, Korth, Sudarshan. Database Systems Concepts. McGraw-Hill, 2002
4. Abiteboul, Hull, Vianu. Foundations of Databases. Addison Wesley, 1995
5. Connolly, Begg. Database Systems - A Practical Approach to Design, Implementation, and Management. Addison Wesley, 2002

# Next topic

- Query Optimization
  - Heuristic optimization
  - Cost-based optimization
- Readings
  - Chapter 19: Algorithms for Query Processing and Optimization
  - Chapters 17: Disk Storage, Basic File Structures, and Hashing (Sections: 13.2, to 13.8)
  - Chapter 18: Indexing Structures for Files (Sections: 14.1 to 14.5)
  - File Organization – COMP261