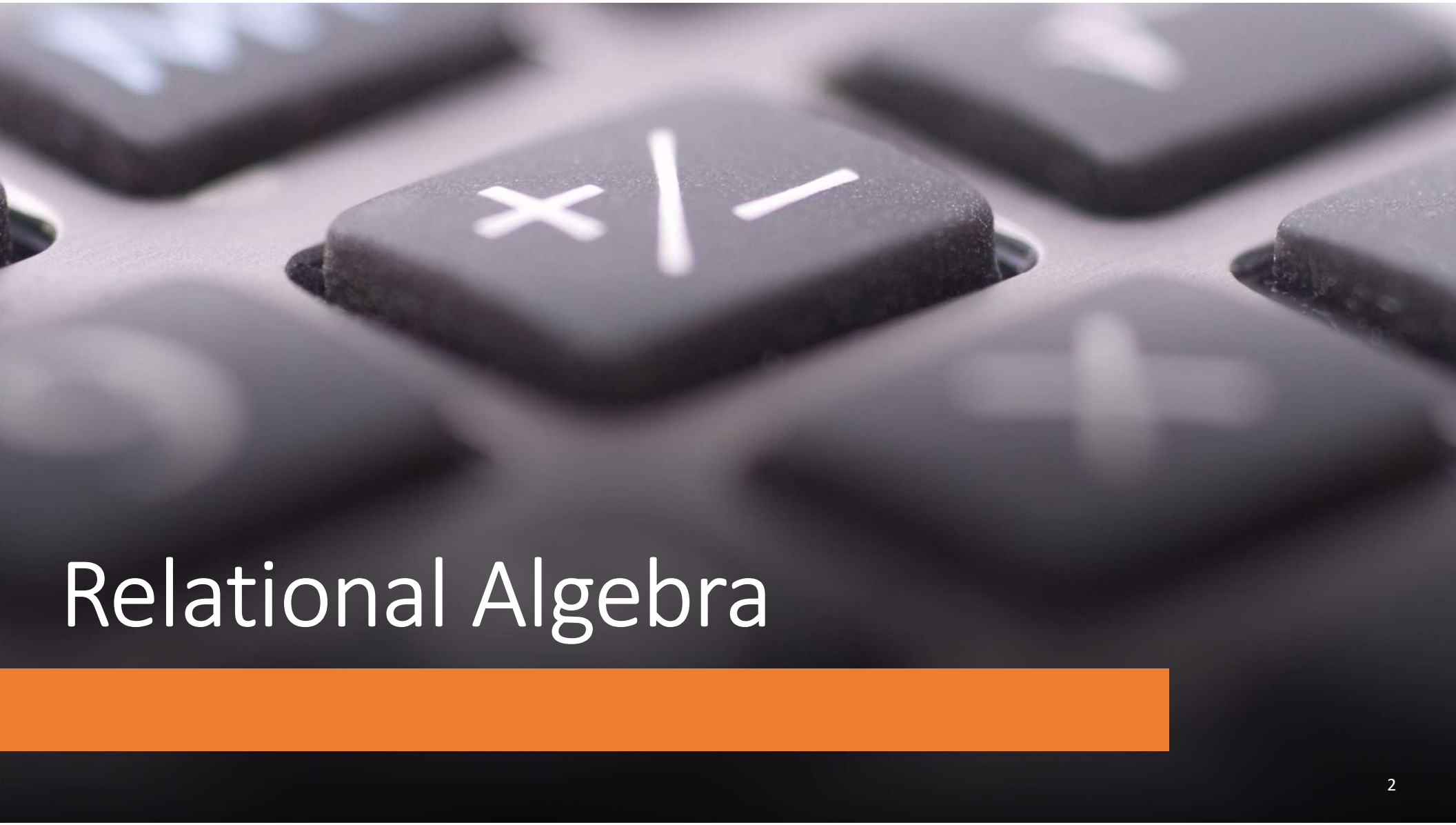
Data Management and Database Design

DMDD 6210

Week #4

Northeastern University



Relational Algebra

Relational Algebra

- It is a procedural query language –
  - which takes instance of relation as input to yields instance of relation as output.
  - It uses **operators** to perform queries

- So, Any operation that we perform on **Relations** produces ***another Relation***

- Relation with **Operator** gives Relations

Unary and Binary Operators

- An operator that expects more than one input is Binary operator
  - Example: Addition, AND, Subtraction, Division, Multiplication

- Unary
  - Factorial (!)
  - ++
  - -- (2 minus symbols)
- Binary
  - ==
  - <
  - >
  - !=

Relational Algebra

Relational Algebra is **operational** , **Simple** to use and can be used to show execution plans

Basic Operations are –

- *Unary operators*
  - Selection (σ)
  - Projection (π)
  - Rename (ρ)
- *Binary operators*
  - Cross product (X)
  - Difference (–)
  - Union (U)
  - Intersection (∩)

Selection – σ

- The Selection operation works on –
  - Relation **R** to select tuples that satisfy the condition (Predicate)
  - Input and output has same schema
  - Condition applied on the schema attributes
  - Reference of attribute can be either by position or name

Selection – σ

List all employees from relation **staff** with salary > $1000

$\sigma_{\texttt{salary > 1000}}$ **(staff)**

- Input relation is **Staff**
- Predicate/condition is **salary > 1000**

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

Projection – π

- Projection –
  - Greek letter Pi
  - Produces a new relation with only some of the attributes of relation **r** and removes duplicate records/tuples.
- Example –
  - Show names and salaries for all employees showing
    - staffNo
    - fName
    - lName
    - Salary

$$\pi_{\text{staffNo,fName, lName, salary}}(\text{STAFF})$$

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

Projection – π

- What is the output of the below –

$$\pi_{deptno}(emp)$$

Employee_Name	deptno
Ryan	10
Smith	10
Scott	30
James	40
Mark	40

σ combined with π

- We can combine selection and projection to get desired set of tuples and attributes –

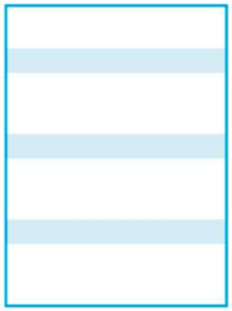
- Example –
  - Show staff's first and last names with salaries equal less than 10,000

$$\pi_{fName,\ lName}(\sigma_{salary<10000}(STAFF))$$

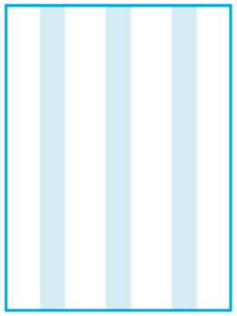
STAFF

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

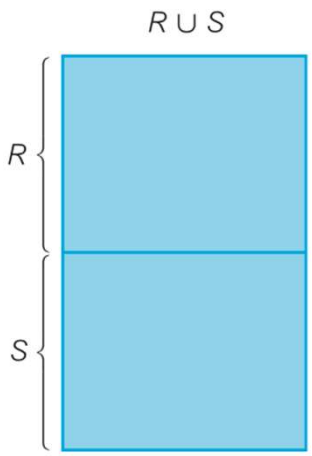
Illustration of Operations

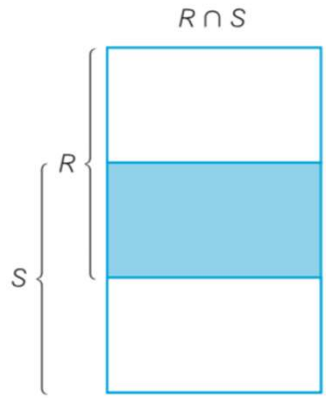


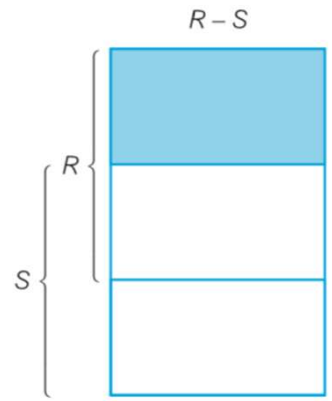
(a) Selection

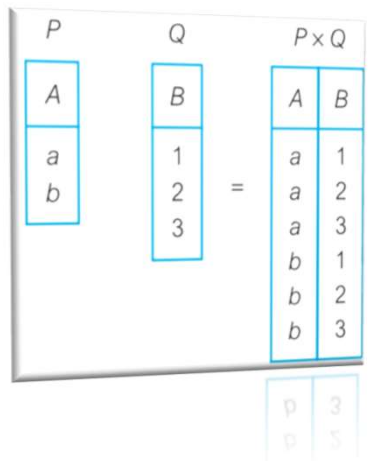


(b) Projection









Union – U

- The 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 eliminated
  - **r** and **s** must be union compatible
  - should have same number of fields
  - Corresponding fields should have same domains (datatypes)

  - Example –
    - List all names with IDs from S1, S2

      πsname (S1) **U** sname (S2)

S1

ID	sname
1	Dustin
2	Steve
3	Ryan
4	David

S2

ID	sname
5	Dustin
1	Joe
6	Mark

Returns Unique names (6 rows with unique names)

Intersection – ∩

- The Intersection operation defines a relation consisting the set of all tuples that are in both **r** and **s**
  - **r** and **s** must be union compatible
- Example –
  - List student names who are part of both S1 and S2
    - S1 ∩ S2

ID	sname
1	Dustin
3	Ryan

S1

ID	sname
1	Dustin
2	Steve
3	Ryan
4	David

S2

ID	sname
1	Dustin
3	Ryan
6	Mark

Difference

- Tuples that are in relation **r**, but not in **s**
- Example –
  - List student names who are part of S1 only

    (S1 – S2)

ID	sname
2	Steve
4	David

S1

ID	sname
1	Dustin
2	Steve
3	Ryan
4	David

S2

ID	sname
1	Dustin
3	Ryan
6	Mark

Cartesian product – X

- The Cartesian product (Cross product) operation defines a relation
  - That is concatenation of every row of relation **r** with every row of relation **s**.
  - Naming conflict could ariase

S1

ID	emp_name
1	Dustin
2	Steve
3	Ryan
4	David

S2

ID	department
10	HR
20	Finance

Each row of S1 paired with each row of S2

(S1 X S2)

ID1	emp_name	ID	department
1	Dustin	10	HR
1	Dustin	20	Finance
2	Steve	10	HR
2	Steve	20	Finance
3	Ryan	10	HR
3	Ryan	20	Finance
4	David	10	HR
4	David	20	Finance

Question

- Find employee names who belong to department 20. (Write relational Algebra Equation)

eID	emp_name
1	Dustin
2	Steve
3	Ryan

S1

dID	department
10	HR
20	Finance

S2

dID	eID
10	1
20	2
20	3

S3

Question

eID	emp_name
1	Dustin
2	Steve
3	Ryan

S1

ID	department
10	HR
20	Finance

S2

dID	eID
10	1
20	2
20	3

S3

eID	emp_name	dID	eID
1	Dustin	10	1
2	Steve	10	1
3	Ryan	10	1
1	Dustin	20	2
2	Steve	20	2
3	Ryan	20	2
1	Dustin	20	3
2	Steve	20	3
3	Ryan	20	3

S1 X S3

$$\pi_{\text{emp\_name}}(\sigma_{\text{dID=20 } \wedge \text{ s1.eid=s3.eid}}(S1 X S3))$$

Rename – ρ

- ρ is the symbol for Rename

  It is Greek letter Rho.

Actors

actor	ayear
Cage	1964
Hanks	1956
Maguire	1975
McDormand	1957

$$\rho_{Stars(name,yob)}(\textbf{Actors})$$

Stars

name	yob
Cage	1964
Hanks	1956
Maguire	1975
McDormand	1957

Rename – ρ

- Resolve conflicts by giving names (Alias conflict columns)
- Give names to results of sub expressions

In this cross product result, eID conflict occurs.

To avoid that, we use rename

σ(S(1→eID1, 4→eID2**),** S1XS3**)**

So resulting schema will be as below

eID1, emp_name, dID, eID2

with output relation name S

eID	emp_name	dID	eID
1	Dustin	10	1
2	Steve	10	1
3	Ryan	10	1
1	Dustin	20	2
2	Steve	20	2
3	Ryan	20	2
1	Dustin	20	3
2	Steve	20	3
3	Ryan	20	3

S1 X S3

Joins

- Very useful operation
- Most used on relations to combined
- Types –
  - Natural join
  - Outer join

Natural Join

- Compound operations involving
  - Selection
  - Projection (optionally)
  - Cross product
- Most commonly used join is a Natural join ($\bowtie$)

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

EMPNO	ENAME	DEPTNO
7369	SMITH	20
7499	ALLEN	30
7521	WARD	30
7566	JONES	20

EMPNO	ENAME	DEPTNO	DNAME
7369	SMITH	20	RESEARCH
7566	JONES	20	RESEARCH
7521	WARD	30	SALES
7499	ALLEN	30	SALES

Notation Algebra Vs SQL

Relational Algebra	SQL
$\pi_{a, b}$	SELECT a, b
$\sigma_{(d > e) \wedge (f = g)}$	WHERE d > e AND f = g
$p \times q$	FROM p, q
$\pi_{a, b}\, \sigma_{(d > e) \wedge (f = g)} (p \times q)$	SELECT a, b FROM p, q WHERE d > e AND f = g; {must always have SELECT even if all attributes are kept, can be written as: SELECT *}
renaming	AS {or blank space}

Notation Algebra / SQL

Relational Algebra	SQL
$p \cup q$	SELECT * FROM p UNION SELECT * FROM q
$p - q$	SELECT * FROM p EXCEPT SELECT * FROM q
$p \cap q$	SELECT * FROM p INTERSECT SELECT * FROM q

Exercise

Reference: Ramakrishnan & Gehrke

1. Find movies made after 1997
2. Find movies made by Hanson after 1997
3. Find all movie titles and their ratings
4. Find all actors and directors
5. Find Coen's movies with McDormand
6. Find movies with Maguire but not McDormand
7. Find actors who have acted in Coen's movie

Movies

title	director	myear	rating
Fargo	Coen	1996	8.2
Raising Arizona	Coen	1987	7.6
Spiderman	Raimi	2002	7.4
Wonder Boys	Hanson	2000	7.6

Actors

actor	ayear
Cage	1964
Hanks	1956
Maguire	1975
McDormand	1957

Acts

actor	title
Cage	Raising Arizona
Maguire	Spiderman
Maguire	Wonder Boys
McDormand	Fargo
McDormand	Raising Arizona
McDormand	Wonder Boys

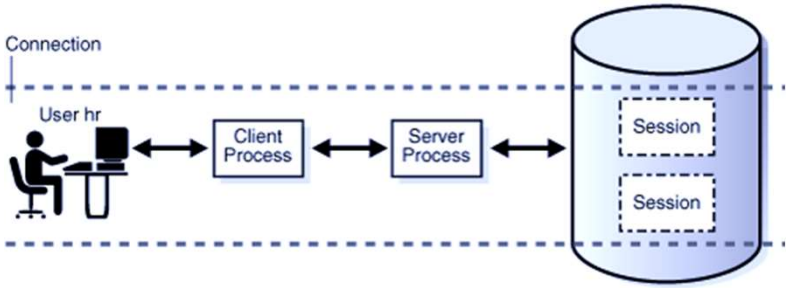
Directors

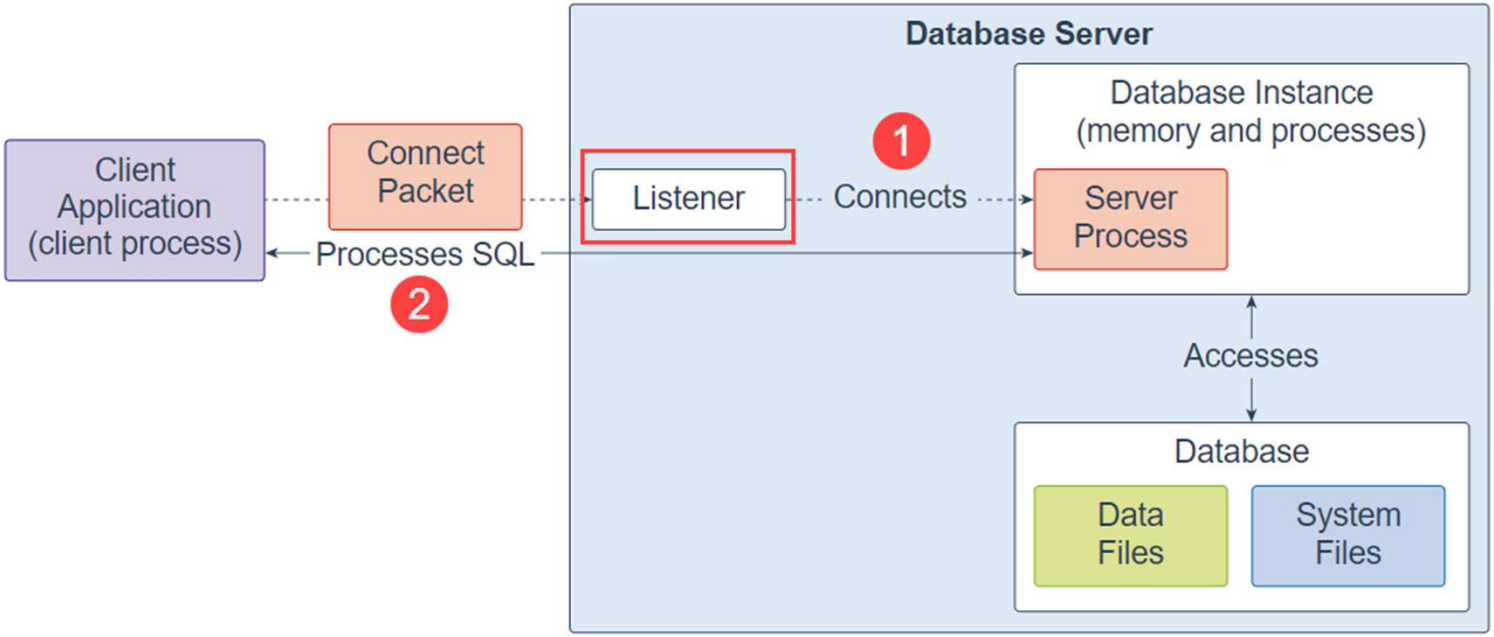
director	dyear
Coen	1954
Hanson	1945
Raimi	1959

Database Architecture

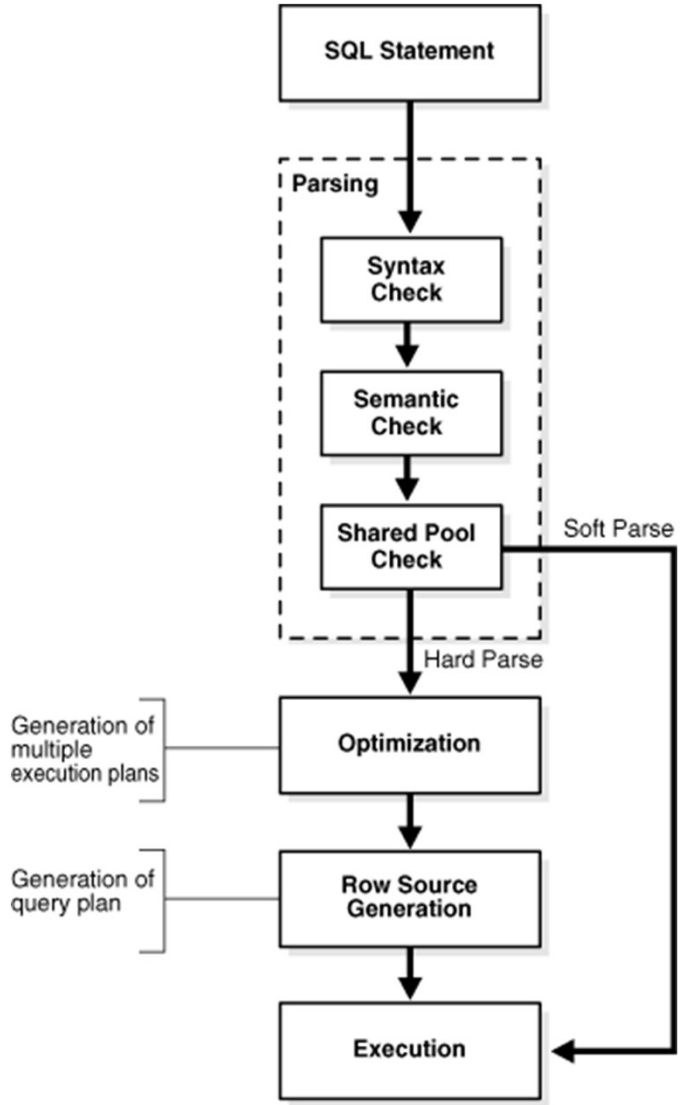
Database Architecture

- Oracle database consists of Database with an Instance
- Instance is combination of Memory and processes
- Database is set of files that store Data


Connection
User hr
Client Process
Server Process
Session
Session


Database Server
Database Instance (memory and processes)
Client Application (client process)
Connect Packet
Listener
1
Connects
Server Process
Processes SQL
2
Accesses
Database
Data Files
System Files

SQL Processing



Database Architecture

- Physical storage structure
  - Datafiles
    - Contains actual data
  - Control files
    - Metadata file contains info such as database name, where the files are located
  - Online redo log files
    - Made up of Redo entry which records changes that are applied to Data

SQL

Language for Relational Databases

Declarative language

We Ask Questions (queries)
→ DB Answers (Result set)

Conditional retrieval

Relational Operators
=
>
<
>=
<=
<> or !=

Logical Operators
AND
OR
NOT

Special Operators	
IN	Checking value in a set
BETWEEN	Checking a value within a range
LIKE	Matching pattern from a column

When a **WHERE** clause is used, the database examines each row to determine if condition is true.

Query a Relation

Select rows from **EMP –**

```
SELECT    empno,
          ename,
          deptno,
FROM      EMP;

---------------------------

SELECT    *
FROM      EMP;
```

Ordering results

- SQL uses **ORDER BY** clause to impose order on result of Query
- **ORDER BY** is used with **SELECT** statement
- **Sorts / Orders** output according to values in selected columns
- One or more columns can be specified in **ORDER BY** clause
- Order can be **ASC** or **DESC** (Default is **ASC** order)
- This **MUST** be the **last** clause in **SELECT** statement
- Example SELECT * FROM emp ORDER BY empno

Types of Join's

- Natural Join
- Outer join
  - There are 3 types of outer joins
    - Left outer join –
      - keeps every tuple in the left-hand relation in the result
    - Right outer join –
      - keeps every tuple in the left-hand relation in the result
    - Full outer join –
      - keeps all tuples in both relations

Query multiple Relations

7782	CLARK	10	ACCOUNTING	NEW YORK
7934	MILLER	10	ACCOUNTING	NEW YORK
7839	KING	10	ACCOUNTING	NEW YORK
7902	FORD	20	RESEARCH	DALLAS
7788	SCOTT	20	RESEARCH	DALLAS
7566	JONES	20	RESEARCH	DALLAS
7369	SMITH	20	RESEARCH	DALLAS
7876	ADAMS	20	RESEARCH	DALLAS
7521	WARD	30	SALES	CHICAGO
7654	MARTIN	30	SALES	CHICAGO
7844	TURNER	30	SALES	CHICAGO
7900	JAMES	30	SALES	CHICAGO
7499	ALLEN	30	SALES	CHICAGO
7698	BLAKE	30	SALES	CHICAGO

Select rows to **DEPT, EMP**

```
SELECT    a.empno,
          a.ename,
          a.deptno,
          b.dname,
          b.loc
FROM      scott.EMP a
JOIN scott.DEPT b
ON  a.deptno = b.deptno
;
-- we can also say "INNER JOIN"
```

Joins

- Natural Join
- Outer join
  - There are 3 types of outer joins
    - Left outer join –
      - keeps every tuple in the left-hand relation in the result
    - Right outer join –
      - keeps every tuple in the left-hand relation in the result
    - Full outer join –
      - keeps all tuples in both relations

Outer Join

- When joining 2 relations –
  - A row in one relation doesn't have matching record in another relation
  - We may want to show such non-matching records in the result set
- The Outer join in which tuples from **DEPT** that do not have matching values in **EMP** are also included in the result set

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

EMP

EMPNO	ENAME	DEPTNO
7369	SMITH	20
7499	ALLEN	30
7566	JONES	20

DEPTNO	DNAME	ENAME
10	ACCOUNTING	
20	RESEARCH	SMITH
20	RESEARCH	JONES
30	SALES	ALLEN
40	OPERATIONS	

Working with NULL values

- NULL values are not 0 (Zero) or a Blank
- It represents an unknown
- It cannot be compared using relational operators
- To locate *NULL* values –
  - A special operator **"IS"** is used with keyword **"NULL"**
- Example

```
SELECT ename
  FROM emp
 WHERE comm IS NULL;
```

NULL Exercise

List employee names who are not eligible for commission

List name and designation of the employee who does not report to anybody

List employee who doesn’t belong to any department

List employees who are eligible for commission

List details of employees whose salary is greater than 2000 with no commission

Working with LIKE operator

- **LIKE** operator is used only with **CHAR** and **VARCHAR2** datatypes
- Used to match a pattern (Wildcard search)
- **%** represents sequence of zero or more characters
- **'_'** (Underscore) stands for any single character
- Both **%** and _ are used with **LIKE** operator to specify a pattern

Exercise

- List name and salary of employees whose salary is more than 1000
- List names of CLERK's working in department 20
- List employee names of job type analysts, salesmen
- List names of employees who are not managers
- List name of employees whose employee numbers are 7369, 7521, 7934, 7788
- List employee names who doesn't belong to department 20, 30
- List employee name and salary whose salary is between 1000 and 2000
- List different jobs available in EMP table

Working with NULL values

- NULL values are not 0 (Zero) or a Blank
- It represents an unknown
- It cannot be compared using relational operators
- To locate *NULL* values –
  - A special operator **"IS"** is used with keyword **"NULL"**
- Example

```
SELECT ename
  FROM emp
 WHERE comm IS NULL;
```

NULL Exercise

- List employee names who are not eligible for commission
- List name and designation of the employee who does not report to anybody
- List employee who doesn’t belong to any department
- List employees who are eligible for commission
- List details of employees whose salary is greater than 2000 with no commission

Working with LIKE operator

- **LIKE** operator is used only with **CHAR** and **VARCHAR2** datatypes
- Used to match a pattern (Wildcard search)
- **%** represents sequence of zero or more characters
- **'_'** (Underscore) stands for any single character
- Both **%** and **_** are used with **LIKE** operator to specify a pattern

Exercise

- List name and salary of employees whose salary is more than 1000
- List names of CLERK's working in department 20
- List employee names of job type analysts, salesmen
- List names of employees who are not managers
- List name of employees whose employee numbers are 7369, 7521, 7934, 7788
- List employee names who doesn't belong to department 20, 30
- List employee name and salary whose salary is between 1000 and 2000
- List different jobs available in EMP table

PSUDO columns

- These are **<u>not</u> actual** columns and are not stored in table
- These behave like **columns**
- We can select for these columns however, We **<u>cannot</u> Insert/Update/Delete**
- These columns are allowed **only** in **SQL**

- **SYSDATE**
- **ROWNUM**
- **ROWID**

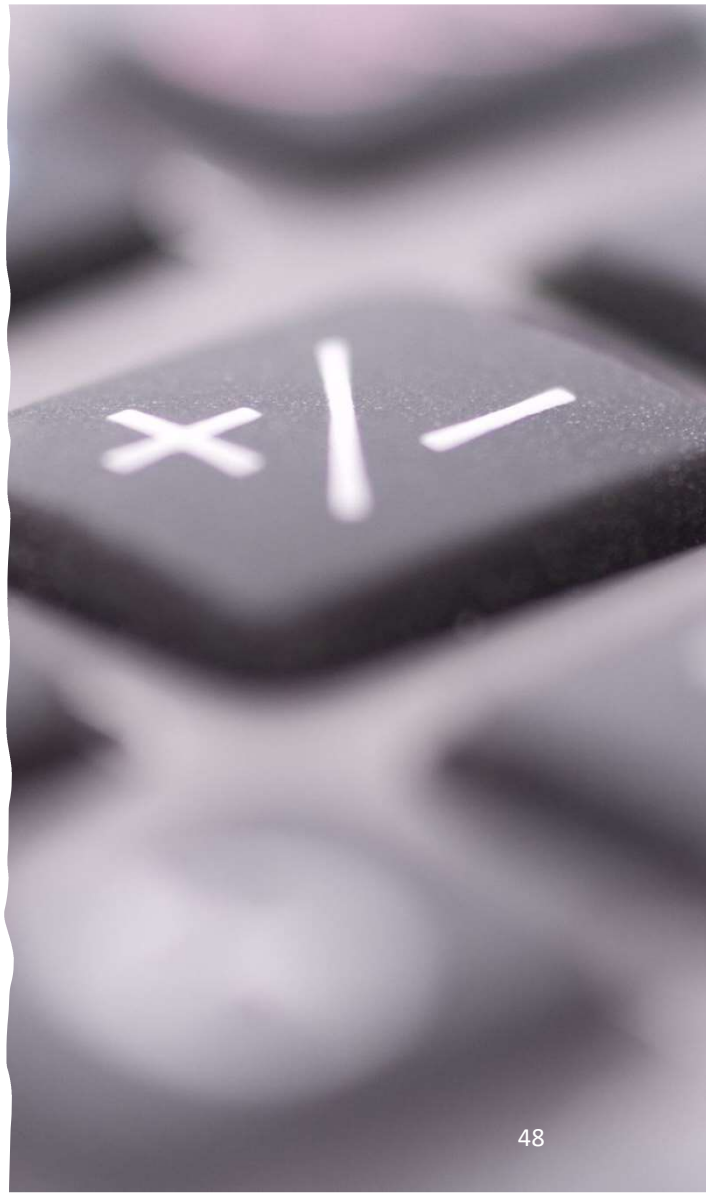
- **SYSTIMESTAMP**
- **NEXTVAL**
- **CURVAL**
- **USER**

PSEUDO columns

```sql
select rowid,
       rownum,
       user as logged_in_user,
       sysdate as current_system_date,
       systimestamp as current_system_dtstamp,
       a.empno,
       a.ename
from scott.emp a
;
```

ROWID	ROWNUM	LOGGED_IN_USER	CURRENT_SYSTEM_DATE	CURRENT_SYSTEM_DTSTAMP	EMPNO	ENAME
AAATJZAAVAAAADzAAA	1	APEX_PUBLIC_USER	02-FEB-19	02-FEB-19 03.33.25.693957 AM +00:00	7839	KING
AAATJZAAVAAAADzAAB	2	APEX_PUBLIC_USER	02-FEB-19	02-FEB-19 03.33.25.693957 AM +00:00	7698	BLAKE
AAATJZAAVAAAADzAAC	3	APEX_PUBLIC_USER	02-FEB-19	02-FEB-19 03.33.25.693957 AM +00:00	7782	CLARK
AAATJZAAVAAAADzAAD	4	APEX_PUBLIC_USER	02-FEB-19	02-FEB-19 03.33.25.693957 AM +00:00	7566	JONES
AAATJZAAVAAAADzAAE	5	APEX_PUBLIC_USER	02-FEB-19	02-FEB-19 03.33.25.693957 AM +00:00	7788	SCOTT
AAATJZAAVAAAADzAAF	6	APEX_PUBLIC_USER	02-FEB-19	02-FEB-19 03.33.25.693957 AM +00:00	7902	FORD
AAATJZAAVAAAADzAAG	7	APEX_PUBLIC_USER	02-FEB-19	02-FEB-19 03.33.25.693957 AM +00:00	7369	SMITH
AAATJZAAVAAAADzAAH	8	APEX_PUBLIC_USER	02-FEB-19	02-FEB-19 03.33.25.693957 AM +00:00	7499	ALLEN
AAATJZAAVAAAADzAAI	9	APEX_PUBLIC_USER	02-FEB-19	02-FEB-19 03.33.25.693957 AM +00:00	7521	WARD
AAATJZAAVAAAADzAAJ	10	APEX_PUBLIC_USER	02-FEB-19	02-FEB-19 03.33.25.693957 AM +00:00	7654	MARTIN
AAATJZAAVAAAADzAAK	11	APEX_PUBLIC_USER	02-FEB-19	02-FEB-19 03.33.25.693957 AM +00:00	7844	TURNER
AAATJZAAVAAAADzAAL	12	APEX_PUBLIC_USER	02-FEB-19	02-FEB-19 03.33.25.693957 AM +00:00	7876	ADAMS
AAATJZAAVAAAADzAAM	13	APEX_PUBLIC_USER	02-FEB-19	02-FEB-19 03.33.25.693957 AM +00:00	7900	JAMES
AAATJZAAVAAAADzAAN	14	APEX_PUBLIC_USER	02-FEB-19	02-FEB-19 03.33.25.693957 AM +00:00	7934	MILLER

Functions



Functions

- Used to manipulate data and can be used to perform complex calculations
- Can modify individual data items
- Can alter data formats to display
- 2 types of functions
  - Column functions
    - Arithmetic functions
    - Character functions
    - Date functions

  - Group functions (Aggregation)

Arithmetic functions

• ABS(n)	Returns Absolute value	abs(-96)	=> 96
• MOD(m,n)	Returns remainder of m divided by n	mod(5,2)	=> 1
• POWER(m,n)	Returns m raised to power n	power(5,2)	=> 25
• SIGN(n)	returns -1 if n is negative, 1 if n is positive	sign(-90.987)	=> -1
• TRUNC(m, [n])	Truncates m to n decimal places.	trunc(90.987,1)	=> 90.9
• ROUND(m, [n])	Rounds column value m to n decimals	round(90.48,1)	=> 90.5
• FLOOR(n)	finds largest integer <= n	floor(98.99)	=> 98

Character functions

- CONCAT(string1, string)
- INITCAP(string)
- LOWER(string)
- UPPER(string)
- LPAD(char1, n [, char2])
- RPAD(char1, n [, char2])

Character functions

```
select chr(65), initcap('hello world!') initcap, lower('HELLO') lower, upper('world') upper, concat('Hello ', 'World!') concat  from dual;
select dname, lpad(dname,15,' ') lpad_dname, rpad(dname,15,' ') rpad_dname from scott.dept where dname in ('SALES');
select '_____Scott_____' str, ltrim('_____Scott_____','_') str_after_ltrim, rtrim('_____Scott_____','_') str_after_ltrim from dual;
select replace('CAT COLD','C','M') from dual;
```

CHR(65)	INITCAP	LOWER	UPPER	CONCAT
A	Hello World!	hello	WORLD	Hello World!

Download CSV

DNAME	LPAD_DNAME	RPAD_DNAME
SALES	SALES	SALES

Download CSV

STR	STR_AFTER_LTRIM	STR_AFTER_LTRIM
_____Scott_____	Scott_____	_____Scott

Download CSV

REPLACE('CATCOLD','C','M')
MAT MOLD

SUBSTR function

```
select substr('Relational-Database',-8) from dual;
Output: Database

select substr('Relational-Database',8) from dual;
Output: nal-Database

select substr('Relational-Database',-8,5) from dual;
Output: Datab
```

```
select length('Hello World') len,
       substr('Operations',2,4) substr,
       instr('California','a',) instr
from dual;
```

LEN	SUBSTR	INSTR
11	pera	2

Few more function

How to find greatest number in the list of numbers 9,10,34,123,7,0?

Note: These numbers are not in form of tuples

```
select greatest(9,10,34,123,7,0)
from    DUAL
/
```

How to find least number between 2 columns?

```
select least(sal, nvl(comm,0)) as least_value
from    emp
/
```

Date functions

Date functions

- **Date functions**
  - TRUNC – Returns date in truncated format.
  - ADD_MONTHS – Add / Subtract months from date
  - TO_CHAR – Converts the ate to character format
  - TO_DATE – Converts character string representing date to date format
  - MONTHS_BETWEEN – Returns number of months between 2 dates

Date functions

```
select systimestamp, trunc(systimestamp)
from dual
```

SYSTIMESTAMP	TRUNC(SYSTIMESTAMP)
11-FEB-19 01.59.51.627853 AM +00:00	11-FEB-19

```
select hiredate,ADD_MONTHS(hiredate,4), ADD_MONTHS(hiredate,-3)
from scott.emp;
```

HIREDATE	ADD_MONTHS(HIREDATE,4)	ADD_MONTHS(HIREDATE,-3)
17-NOV-81	17-MAR-82	17-AUG-81
01-MAY-81	01-SEP-81	01-FEB-81

```
select MONTHS_BETWEEN('01-JAN-2018', '01-MAR-2018'),
       MONTHS_BETWEEN('01-MAR-2018', '01-JAN-2018')
from DUAL;
```

MONTHS_BETWEEN('01-JAN-2018','01-MAR-2018')	MONTHS_BETWEEN('01-MAR-2018','01-JAN-2018')
-2	2

Date Formats

Date Format	Details	Example
DD-MON-YY	Number Day – DD 3 letter Month – MON 2 letter Year. – YY	Oracle default date format 28-JAN-2019
DDD D DAY	Number of days passed since 1st Jan for that year Number of days passed in week Name of Day	Lets say if today is Monday then we it returns 2 **Select to_char(sysdate, 'D') from dual;**
YYYY YYY YY Y YEAR	Each Y represents a number from year so, 4Ys 4 digit year. 3 Y's last 3 numbers from Year Year spelled out	2019 019 19 9 select to_char(sysdate, 'YEAR') from dual; **TWENTY NINETEEN**
HH AM/PM HH24	Hour of the day with AM/PM sufix Hour of day in 24hr format	09 PM 21
MI SS	Minutes of Hour Seconds of Minute	

Date function RR and YY format

- RR format Lets you store 20th century dates in the 21st century using only two digits.
- YY format of date function considers year from 1900 onwards

```
SQL> insert into x values (1, to_date('10-MAR-99','DD-MON-YY'));

1 row created.

SQL> insert into x values (1, to_date('10-MAR-99','DD-MON-RR'));

1 row created.

SQL> select * from x;

        ID DOH
---------- ---------
         1 10-MAR-99
         1 10-MAR-99
```

```
SQL> select id, to_char(doh, 'DD-MON-RRRR') from x;

        ID TO_CHAR(DOH
---------- -----------
         1 10-MAR-2099
         1 10-MAR-1999
```

Aggregation functions

- The aggregate functions produce single value for an entire group of table
- These are used to produce summarized results and operate on set of rows
  - **COUNT**
  - **SUM**
  - **MAX**
  - **MIN**
  - **AVG**

Note In all the above functions, NULLs are ignored

Examples

- Average salary of Salesman from Employee table

```
SQL> select avg(sal) from emp where job='SALESMAN';
  AVG(SAL)
----------
      1400
```

- What is the Maximum salary paid for Clerk's

```
SQL> select max(sal) from emp where job='CLERK';
  MAX(SAL)
----------
      1300
```

Examples

Maximum salary under each job category

```
select job, max(sal) from emp;
```

- True
- False

Examples

- Maximum salary under each job category

```
SQL> select job, max(sal) from emp;
select job, max(sal) from emp
       *
ERROR at line 1:
ORA-00937: not a single-group group function
```

Aggregate functions compute a single value from a multiset of inputs

Grouping the Result

- **GROUP BY** clause is used to divide rows into smaller groups
- Used with **SELECT** clause and can group multiple attributes
- SQL groups the result *after it retrieves rows from table*
- Conditional retrieval of rows from a grouped results is possible using **HAVING** clause
- If any **AGGREGATE** function is present in **SELECT**
  - Then it is applied to **GROUPED** result
- **ORDER BY** clause can be used to order the final result

Examples

- What is the Maximum salary under each job category

```
SQL> select job,
            max(sal)
       from emp
     group by job;

    JOB     MAX(SAL)
--------- --------
CLERK           1300
SALESMAN        1600
PRESIDENT 5000
MANAGER         2975
ANALYST         3000
```

Remove Duplicates

DISTINCT keyword removes duplicates

```
Select distinct job
from emp;
```

Group by can also be used to remove duplicates

```
Select job
from emp
group by job;
```

Thinking Time

Find count of departments that currently have one or more employees

```
Select count(deptno) from emp;
Select count(distinct(deptno)) from emp;
```

Which one
Is correct and why?

Exercise

1. List all employee names along with their manager names. Make sure to print the employee who has no manager.
2. List employees whose names start with capital **"S"**
3. List names of employees whose names have exactly 5 characters
4. List names of employee having **"I"** as second character
5. List name , total of salary and commission as total_income of all employees
6. List name, sal and Calculate 10% bonus amount based on their salary for each employee

Exercise

- List average salary and number of employees working in each department
  - **select deptno,avg(sal),count(*) from emp group by deptno;**

- List Department numbers and number of employees, total salary payable, maximum and minimum salary in each department
  - **select deptno, count(*), sum(sal), max(sal), min(sal) from emp group by deptno;**

- List average salary of all departments employing more than 5 people
  - **select deptno,avg(sal) from emp group by deptno having count(*) >=5;**

- List all jobs names from EMP table where maximum salary is >= 2000
  - **select job from emp group by job having max(sal)>=2000;**

Quiz

```
SQL> DESC LEAD
 Name
 ----------------------
 COMPANY       VARCHAR2(30)
 PHONE         VARCHAR2(20)

SQL> select * from lead;
COMPANY                       PHONE
------------------------ ---------------
Acme Production               5553214321
Basic Apparel          002495559875432
Century Movies         456123789
Danish Design          004566554433
Ewok Emporium          86427531
```

Phone number has to be used as account ID in CUSTOMER Table.

Challenge is –

account ID is a VARCHAR2(10) in customer table

Expected output –

```
ACCOUNTID COMPANY
---------- ------------------------
456123789  Century Movies
4566554433 Danish Design
5553214321 Acme Production
5559875432 Basic Apparel
86427531   Ewok Emporium
```

Questions?