Basic Queries

INSTRUCTOR

ID	NAME	AGE	DEPT_NAME	SALARY
E01	NITA	23	CSE	40000
E02	VISHAL	29	MECHANICAL	45000
E03	ANU	30	EEE	35000
E04	ABHISHEK	35	CSE	50000
E05	ARUN	25	EEE	50000
E06	AJITH	32	CSE	38000
E07	PREMA	45	PHYSICS	60000
E08	RAJANI	48	PHYSICS	55000

Basic Query Structure

- The SQL **data-manipulation language (DML)** provides the ability to query information, and insert, delete and update tuples
- A typical SQL query has the form:

select ***attr1,attr2..attrn***
from $r_1, r_2, \ldots, r_m$
where ***condition***

- The result of an SQL query is a relation.

The select Clause

- The SELECT statement is used to select data from a database

Example: **find the names of all instructors:**

select *name*
from *instructor*

- NOTE: SQL names are case insensitive (i.e., you may use upper- or lower-case letters.)
  - E.g. *Name* ≡ *NAME* ≡ *name*
  - Some people use upper case wherever we use bold font.

The select Clause (Cont.)

- SQL allows **duplicates** in relations as well as in query results.
- To force the **elimination of duplicates**, insert the keyword **distinct** after select**.**
- **Find the names of all departments from instructor, and remove duplicates**

```
select distinct dept_name
from instructor
```

- The keyword **all** specifies that duplicates not be removed.

```
select all dept_name
from instructor
```

The select Clause (Cont.)

- An asterisk in the select clause denotes "**all attributes**"

  select *
  from *instructor*

- The **select** clause can contain arithmetic expressions involving the operation, +, –, *, and /, and operating on constants or attributes of tuples.
- The query:

  select *ID, name, salary/12*
  from *instructor*

  would return a relation that is the same as the *instructor* relation, except that the value of the attribute *salary* is divided by 12.

The where Clause

- The **where** clause specifies **conditions** that the result must satisfy

- Comparison results can be combined using the logical connectives

  and, or, and **not.**

- Comparisons can be applied to results of arithmetic expressions.

- **To find all instructors in Comp. Sci. dept with salary > 40000**

  select *name*
  from *instructor*
  where *dept_name* = 'CSE' **and** *salary* > 40000

Operators in The WHERE Clause

Operator	Description
=	Equal
<>	Not equal. Note: In some versions of SQL this operator may be written as !=
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

1. Display the details of ‘NITA’ and ‘ANU’ from the instructor table
2. Display the details of all instructors whose age ranges between 23 and 100
3. Display the details of all instructors whose age is 25

ID	NAME	AGE	DEPT_NAME	SALARY
E01	NITA	23	CSE	40000
E02	VISHAL	29	MECHANICAL	45000
E03	ANU	30	EEE	35000
E04	ABHISHEK	35	CSE	50000
E05	ARUN	25	EEE	50000
E06	AJITH	32	CSE	38000
E07	PREMA	45	PHYSICS	60000
E08	RAJANI	48	PHYSICS	55000

- select * from INSTRUCTOR
  where name **in** (‘NITA‘, ‘ANU’)
- select * from INSTRUCTOR
  where name **not in** (‘NITA‘, ‘ANU’)
- select * from INSTRUCTOR
  where age **between** 23 and 100
- select * from INSTRUCTOR
  where age **not between** 23 and 100
- select * from INSTRUCTOR
  where **not** age=25

LIKE Operator

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%_%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

ORDER BY Syntax

The **ORDER BY** clause orders or sorts the result of a query according to the values in one or more specific columns.

- **SELECT** *column1, column2, ...*
  FROM table_name
  ORDER BY *column1, column2, ...* **ASC|DESC**;

- Display the details of instructor as the **ascending order** of salary

```
Select *
   from instructor
        order by salary
```

ID	NAME	AGE	DEPT_NAME	SALARY
E01	NITA	23	CSE	40000
E02	VISHAL	29	MECHANICAL	45000
E03	ANU	30	EEE	35000
E04	ABHISHEK	35	CSE	50000
E05	ARUN	25	EEE	50000
E06	AJITH	32	CSE	38000
E07	PREMA	45	PHYSICS	60000
E08	RAJANI	48	PHYSICS	55000

Ordering by more than one columns

- Display the instructor details with the following conditions:

1. Department name should be in descending order
2. Within each department salary should be in ascending order

ID	NAME	AGE	DEPARTMENT	SALARY
E08	RAJANI	48	**PHY**	**55000**
E07	PREMA	45	**PHY**	**60000**
E02	VISHAL	29	**MECH**	**45000**
E03	ANU	30	**EEE**	**35000**
E05	ARUN	25	**EEE**	**50000**
E06	AJITH	32	**CSE**	**38000**
E01	NITA	23	**CSE**	**40000**
E04	ABHISHEK	35	**CSE**	**50000**

Ordering by more than one columns

- Display the instructor details with the following conditions:

1. Department name should be in descending order
2. Within each department salary should be in ascending order

ID	NAME	AGE	DEPARTMENT	SALARY
E08	RAJANI	48	**PHY**	**55000**
E07	PREMA	45	**PHY**	**60000**
E02	VISHAL	29	**MECH**	**45000**
E03	ANU	30	**EEE**	**35000**
E05	ARUN	25	**EEE**	**50000**
E06	AJITH	32	**CSE**	**38000**
E01	NITA	23	**CSE**	**40000**
E04	ABHISHEK	35	**CSE**	**50000**

```
SELECT *
FROM INSTRUCTOR
ORDER BY DEPNAME DESC,SALARY;
```

Aggregate Functions

- These functions operate on the multiset of values of a column of a relation, and return a value

avg: average value
min: minimum value
max: maximum value
sum: sum of values
count: number of values

Aggregate Functions (Cont.)

- Find the average salary of instructors in the Computer Science department
  - **select avg** (*salary*)
    from *instructor*
    where *dept_name*= 'Comp. Sci.';
- Find the number of tuples in the *indtructor* relation
  - **select count** (*)
    from *course*;
- Display the count of instructors in each department

DEPT	DEPT_COUNT
PHY	2
EEE	2
CSE	3
MECH	1

GROUPBY Syntax

Display the count of instructors in each department

ID	NAME	AGE	DEPNAME	SALARY
E08	RAJANI	48	PHY	55000
E07	PREMA	45	PHY	60000
E02	VISHAL	29	MECH	45000
E03	ANU	30	EEE	35000
E05	ARUN	25	EEE	50000
E06	AJITH	32	CSE	38000
E01	NITA	23	CSE	40000
E04	ABHISHEK	35	CSE	50000

SELECT *column_name(s)*
FROM *table_name*
WHERE *condition*
GROUP BY *column_name(s)*

Select **depname as dept**,**count(*) as dept_count**
from instructor
group by depname;

Display the count of instructors in each department as dept_count and order the output as the ascending order of department name

ID	NAME	AGE	DEPNAME	SALARY
E08	RAJANI	48	**PHY**	**55000**
E07	PREMA	45	**PHY**	**60000**
E02	VISHAL	29	**MECH**	**45000**
E03	ANU	30	**EEE**	**35000**
E05	ARUN	25	**EEE**	**50000**
E06	AJITH	32	**CSE**	**38000**
E01	NITA	23	**CSE**	**40000**
E04	ABHISHEK	35	**CSE**	**50000**

DEPT	DEPT_COUNT
CSE	3
EEE	2
MECH	1
PHY	2

Select **depname as dept**,**count(*) as dept_count**
from instructor
group by depname
Order by depname ;

The from Clause

- The **from** clause lists the relations involved in the query
  - Corresponds to the Cartesian product operation of the relational algebra.
- Find the Cartesian product *instructor X teaches*

  select *
  from *instructor, teaches*
  - generates every possible instructor – teaches pair, with all attributes from both relations
- Cartesian product not very useful directly, but useful combined with where-clause condition (selection operation in relational algebra)

SalesPerson

SNUM	SNAME	CITY	COMM
1001	Peel	London	.12
1002	Serres	San Jose	.13
1004	Motika	London	.11
1007	Rifkin	Barcelona	.15
1003	AxelRod	New York	.10
1005	Fran	London	.26

Customers

CNUM	CNAME	CITY	RATING	SNUM
2001	Hoffman	London	100	1001
2002	Giovanni	Rome	200	1003
2003	Liu	San Jose	200	1002
2004	Grass	Berlin	300	1002
2006	Clemens	London	100	1001
2008	Cisneros	San Jose	300	1007
2007	Pereira	Rome	100	1004

Orders

ONUM	AMT	ODATE	CNUM	SNUM
3001	18.69	10/03/96	2008	1007
3003	767.19	10/03/96	2001	1001
3002	1900.10	10/03/96	2007	1004
3005	5160.45	10/03/96	2003	1002
3006	1098.16	10/03/96	2008	1007
3009	1713.23	10/04/96	2002	1003
3007	75.75	10/04/96	2002	1003
3008	4723 .00	10/05/96	2006	1001
3010	1309.95	10/06/96	2004	1002