Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

MIT-WPU
॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥

CS312 Database Management Systems

School of Computer Engineering and Technology

CS312 Database Management Systems

Course Objectives:

1) Understand and successfully apply logical database design principles, including E-R diagrams and database normalization.
2) Learn Database Programming languages and apply in DBMS application
3) Understand transaction processing and concurrency control in DBMS
4) Learn database architectures, DBMS advancements and its usage in advance application

- **Course Outcomes:**

1) Design ER-models to represent simple database application scenarios and Improve the database design by normalization.
2) Design Database Relational Model and apply SQL , PLSQL concepts for database programming
3) Describe Transaction Processing and Concurrency Control techniques for databases
4) Identify appropriate database architecture for the real world database application

LABORATORY ASSIGNMENT NO: 03

Create Table Construct

- SQL relation is defined using the **create table** command:

  create table *r* (A_1 D_1, A_2 D_2, ..., A_n D_n,
  (integrity-constraint$_1$),
  ...,
  (integrity-constraint$_k$))

- **Example:**

  create table *instructor* (
  ID **char**(5),
  n*ame* **varchar**(20)**,**
  dept_name **varchar**(20),
  salary **decimal**(8,2))

ID	*name*	*dept_name*	*salary*

Data Manipulation Language (DML)

DML commands are used to make modifications of the Database like,

- Insertion of new tuples into a given relation
- Deletion of tuples from a given relation.
- Updation of values in some tuples in a given relation

INSERT Query

- Add a new tuple to *course*

  insert into *course*

  values (’CS-437’, ’Database Systems’, ’Comp. Sci.’, 4);

- or equivalently

  insert into *course* (*course_id*, *title*, *dept_name*, *credits*)

  values (’CS-437’, ’Database Systems’, ’Comp. Sci.’, 4);

- Add a new tuple to *student* with *tot_creds* set to null

  insert into *student*
  values (’3003’, ’Green’, ’Finance’, *null*);

DELETE Query

- Delete all instructors from the Finance department

  delete from *instructor*
  where *dept_name*= ’Finance’;

- Delete all tuples in the *student* relation.

  delete from *student;*

UPDATE Query

- Increase salaries of instructors whose salary is over $100,000 by 3%, and all others by a 5%
  - Write two **update** statements:

```
update instructor
    set salary = salary * 0.03
    where salary > 100000;

update instructor
    set salary = salary * 0.05
    where salary <= 100000;
```

  - Can be done better using the **case** statement (next slide)

SELECT Query

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

Select ------  Mandatory Clause

From

Where

Group by  Optional Clauses(Use as per need)

Having

Order by

E.g. ***SELECT * FROM*** *Student****;***

  - The result of an SQL query is a relation.

SELECT Query (Cont..)

- An attribute can be a literal with **from** clause

select 'A' from instructor

  - Result is a table with one column and *N* rows (number of tuples in the *instructors* table), each row with value "A".

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.
  - For common attributes (e.g., *ID*), the attributes in the resulting table are renamed using the relation name (e.g., *instructor.ID*)
- Cartesian product not very useful directly, but useful combined with where-clause condition (selection operation in relational algebra).

The WHERE Clause

- The **where** clause specifies conditions that the result must satisfy
  - Corresponds to the selection predicate of the relational algebra.
- To find all instructors in Comp. Sci. dept

  select *name*
  from *instructor*
  where *dept_name* = 'Comp. Sci.'
- Comparison results can be combined using the logical connectives **and, or,** and **not**
  - To find all instructors in Comp. Sci. dept with salary > 80000

    select *name*
    from *instructor*
    where *dept_name* = 'Comp. Sci.' **and** *salary* > 80000
- Comparisons can be applied to results of arithmetic expressions.

Where Clause Predicates

- SQL includes a **between AND** comparison operator
- Example: Find the names of all instructors with salary between $90,000 and $100,000 (that is, >= $90,000 and <= $100,000)

select *name*
from *instructor*
where *salary* **between** 90000 **and** 100000

Null Values

- It is possible for tuples to have a null value, denoted by *null*, for some of their attributes
- *null* signifies an unknown value or that a value does not exist.
- The result of any arithmetic expression involving *null* is *null*
  - Example: 5 + *null* returns null
- The predicate **is null** can be used to check for null values.
  - Example: Find all instructors whose salary is null.

select *name*
from *instructor*
where *salary* **is null**

Renaming table in Select clause

- The SQL allows renaming relations and attributes using the **as** clause:

  old-name **as** *new-name*

- Find the names of all instructors who have a higher salary than some instructor in 'Comp. Sci'.

  select distinct *T.name*
  from *instructor* **as** *T*, *instructor* **as** *S*
  where *T.salary* > *S.salary* **and** *S.dept_name* = *'Comp. Sci.'*

- Keyword **as** is optional and may be omitted

  instructor **as** *T* ≡ *instructor T*

SQL Operators

- SQL Arithmetic Operators
- SQL Comparison Operators
- SQL Logical Operators

Arithmetic Operators

- 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.
  - Can rename "s*alary/12"* using the **as** clause:

    select *ID*, *name*, *salary/12* **as** *monthly_salary*

SQL Comparison Operators

- *Select * from employee*
  where salary = 90000;

- *Select * from employee*
  where salary <>100000;

- *Select * from employee*

where salary >=90000 and salary<=100000

Operator	Description
=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
<>	Not equal to

SQL Logical Operators

Operator	Description
ALL	TRUE if all of the subquery values meet the condition
AND	TRUE if all the conditions separated by AND is TRUE
ANY	TRUE if any of the subquery values meet the condition
BETWEEN	TRUE if the operand is within the range of comparisons
EXISTS	TRUE if the subquery returns one or more records
IN	TRUE if the operand is equal to one of a list of expressions
LIKE	TRUE if the operand matches a pattern
NOT	Displays a record if the condition(s) is NOT TRUE
OR	TRUE if any of the conditions separated by OR is TRUE
SOME	TRUE if any of the subquery values meet the condition

Logical Operators(Cont..)

- **ALL**

  SELECT * FROM Products WHERE Price > ALL (SELECT Price FROM Products WHERE Price > 500);

- **AND**

  SELECT * FROM Customers WHERE City = "London" AND Country = "UK";

- **ANY**

  SELECT * FROM Products WHERE Price > ANY (SELECT Price FROM Products WHERE Price > 50);

- **BETWEEN AND**

  SELECT * FROM Products WHERE Price BETWEEN 50 AND 60;

- **EXISTS**

  SELECT * FROM Products WHERE EXISTS (SELECT Price FROM Products WHERE Price > 50);

Logical Operators(Cont..)

- **IN**

  SELECT * FROM Customers WHERE City IN ('Paris','London');

- **LIKE**

  SELECT * FROM Customers WHERE City LIKE 's%';

- **NOT**

  SELECT * FROM Customers WHERE City NOT LIKE 's%';

- **OR**

  SELECT * FROM Customers WHERE City = "London" OR Country = "UK";

- **SOME**

SELECT * FROM Products WHERE Price > SOME (SELECT Price FROM Products WHERE Price > 20);

SQL Functions

- **Single Row Functions** : Operate on each row and return one output for each row.
  - Date Functions, String Functions such as length or case conversion functions like UPPER, LOWER.
  - Number functions such as ROUND, TRUNC, and MOD etc.
- **Multi Row Functions** : Aggregate Function/Group Functions : Operates on Group of rows and return output for the complete set of rows. Also known as Group functions.
  - Min, Max, Count, Sum, Avg etc.
- SQL Single Row Functions can be used in Select Clause, Where Clause, Group By Clause, Order By clause
- SQL Multi Row Functions can be used in Select Clause, Group By Clause, Having Clause.

String Function : Use in Select, Where, group by , having , order by Clause

Function	Meaning
Char_length(string)	Return number of characters in argument
Concat(expr1,expr2)	Return concatenated string
Instr(expr1,expr2)	Return the index of the first occurrence of substring
Lower(expr1)	Return the argument in lowercase
Left(expr1,count)	Return the leftmost number of characters from string
Lpad(expr1,length,expr2)	left-pads a string with another string, to a certain length
Ltrim()	Remove leading spaces
Substr(string,startpos,length)	extracts a substring from a string (starting at any position).
LOCATE(*substring*, *string*, *start*)	returns the position of the first occurrence of a substring in a string
STRCMP(*string1*, *string2*)	compares two strings. Returns 0,1,-1
Upper(string)	Convert the text to upper-case
Trim(string)	removes leading and trailing spaces from a string.

DATE Function : Use in Select, Where, group by having Clause, order by clause

Function	Meaning
DATE_ADD(date, INTERVAL value addunit)	Adds a specified time interval to a date.
CURDATE() function	returns the current date. as "YYYY-MM-DD" (string)
DATEDIFF(date1, date2)	returns the number of days between two date values
DATE_SUB(date, INTERVAL value interval)	subtracts a time/date interval from a date and then returns the date
DAY(date)	returns the day of the month for a given date
DAYNAME(date)	returns the weekday name for a given date.
SYSDATE()	returns the current date and time.

Single row Function : String (Pattern matching)

- Patterns are case sensitive; that is, uppercase characters do not match lowercase characters, or vice-versa.
- To illustrate pattern matching, we consider the following examples:
  - **Percent (%):** The % character matches any substring.
  - **Underscore (_):** The character matches any character in the string.
- 'Intro%' matches any string beginning with "Intro".
- '%Comp%' matches any string containing "Comp" as a substring, for example, 'Intro. to Computer Science', and 'Computational Biology'.
- '---' matches any string of exactly three characters.
- ' ---%' matches any string of at least three characters.

Pattern matching examples……

- Syntax is

select <column name> from <table_name >where <column_name > ***like/ not like*** *'pattern';*

- To find the records starting with ‘Luck’
  - *SELECT * FROM student WHERE city LIKE 'Luck%';*
- To find the names not starting with ‘Luck’
  - *SELECT name FROM student WHERE city NOT LIKE 'Luck%';*
- To find the names ending with ‘ly’
  - *SELECT * FROM student WHERE city LIKE '%fy’ ;*
- Find names containing a y
  - *SELECT * FROM student WHERE city LIKE '%y%‘;*
- To find names containing exactly five characters
  - *SELECT * FROM student WHERE city LIKE '_____';*

Ordering the Display of Tuples

- List in alphabetic order the names of all instructors

  select distinct *name*
  from *instructor*
  order by *name*

- We may specify **desc** for descending order or **asc** for ascending order, for each attribute; ascending order is the default.

  Example: **order by** *name* **desc**

- Can sort on multiple attributes

  Example: **order by** *dept_name*, *name*

Aggregate Functions

Type	Use	Functions
Single –row functions	Operate on a single column of a relation of single row n the table returning single value as an output	String functions, Date Functions
Multiple –row functions	Act on a multiple row in the relation returning single value as an output	Avg, min, max, sum, count

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

Aggregate Functions Examples

Find the average salary of instructors in the Computer Science department

- **select avg** (*salary*),min(salary), max(salary),sum(salary)
  from *instructor*
  where *dept_name*= 'Comp. Sci.';

Find the number of tuples in the *course* relation

- **select count** (*) **from instructor**;

ID	*name*	*dept_name*	*salary*
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000

Aggregate Functions – Group By

Find the average salary of instructors in each department

- **select** *dept_name*, **avg** (*salary*) **as** *avg_salary*
  from *instructor*
  group by *dept_name*;

ID	*name*	*dept_name*	*salary*
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000

dept_name	*avg_salary*
Biology	72000
Comp. Sci.	77333
Elec. Eng.	80000
Finance	85000
History	61000
Music	40000
Physics	91000

Aggregation (Cont.)

Attributes in **select** clause outside of aggregate functions must appear in **group by** list

- /* erroneous query */
  select *dept_name*, *ID*, **avg** (*salary*)
  from *instructor*
  group by *dept_name*;

Discuss why query is erroneous, [Hint :refer last table]

Aggregate Functions – Having Clause

Find the names and average salaries of all departments whose average salary is greater than 42000

select *dept_name*, **avg** (*salary*) **as** *avg_salary*
from *instructor*
group by *dept_name*
having avg (*salary*) > 42000;

ID	*name*	*dept_name*	*salary*
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000

Note: predicates in the **having** clause are applied after the formation of groups whereas predicates in the **where** clause are applied before forming groups

Null Values and Aggregates

- To find the total all salaries

  select sum *(salary)* ***from*** *instructor*

  - Above statement ignores null amounts
  - Result is *null* if there is no non-null amount

- All aggregate operations except **count(*)** ignore tuples with null values on the aggregated attributes
  - What if collection has only null values?
    - count returns 0
    - all other aggregates return null

Perform DML Statements

Create tables with all constraints and perform SQL- DML (Insert, Update, Delete), SQL Select- Logical IN, Negation , NULL, Comparison Operators. Where Clause, Between AND, Exists, ALL, LIKE command and solve queries.

Exercises -Batch A B C D

Create Database

- Create a database called COMPANY consisting of 2 tables
  i. EMP
  ii. DEPT
- **EMP Table Fields**

Column name	Data type	Description
EMPNO	Number	Employee number
ENAME	Varchar	Employee name
JOB	Char	Designation
MGR	Number	Manager's Emp. number
HIREDATE	Date	Date of joining
SAL	Number	Basic Salary
COMM	Number	Commission
DEPTNO	Number	Department Number

Create Database

- **DEPT Table Fields**

Column name	Data type	Description
DEPTNO	Number	Department number
DNAME	Varchar	Department name
LOC	Varchar	Location of department

Employee table sample records

EmpNo	Ename	Job	MGR	HireDate	Salary	Commission	Deptno
7369	Smith	Clerk	7902	17/12/80	800	300	20
7499	Allen	Salesman	7698	20/02/81	1600	300	30

Dept table sample records

DeptNo	Dname	Loc
10	Accounting	New York
20	Research	Dallas
30	Sales	Chicago
40	Operations	Boston

Queries- SET 1

1) List the number of employees and average salary for employees in department 20.

2) List name, salary and PF amount of all employees. (PF is calculated as 10% of basic salary)

3) List the employee details in the ascending order of their basic salary.

4) List the employee name and hire date in the descending order of the hire date.

5) List employee name, salary, PF, HRA, DA and gross; order the results in the ascending order of gross. HRA is 50% of the salary and DA is 30% of the salary.

6) List the department numbers and number of employees in each department.

7) Increment the Salary of salesman by 10% of basic salary.

8) List the total salary, maximum and minimum salary and average salary of the employees, for department 20.

9) List the employees whose names contains 3rd letter as 'I'.

10) List the maximum salary paid to a salesman.

11) Increase the salary of salesman by 10% of their current salary.

Queries- SET 2

1. List the employee names and his annual salary dept wise.
2. Find out least 5 earners of the company.
3. List the records from emp whose deptno is not in dept
4. List those employees whose sal is odd value.
5. List the employees whose sal contain 3 digits.
6. List the employees who joined in the month of ‘DEC’
7. List the employees whose names contains ‘A’
8. List the maximum, minimum and average salary in the company.
9. Write a query to return the day of the week for any date(or HIRE_DATE) entered in format ‘DD-MM-YY’
10. Count the no of characters in employee name without considering spaces for each name.
11. List the employees who are drawing less than 1000. sort the output by salary.

Queries- Set 3

1. Write a query in SQL to display the unique designations for the employees.
2. Delete Employees who joined in Year 1980.
3. Increase the salary of Managers by 20% of their current salary.
4. List employees not belonging to department 30, 40, or 10.
5. List the different designations in the company.
6. List the names of employees who are not eligible for commission.
7. List employees whose names either start or end with “S”.
8. List employees whose names have letter “A” as second letter” in their names.
9. List the number of employees working with the company.
10. List the emps with hiredate in format June 4,1988.
11. List the salesmen who get the commission within a range of 200 and 500.

Queries- Set 4

1. List names of employees who are more than 2 years old in the company.
2. List the employee details in the ascending order of their basic salary.
3. Display the employees who have more salary as that of Smith
4. Increment the salary of Emp no. 7499 by 10% of his current salary.
5. List the employees whose salary is between 10000 and 25000.
6. List the names of employees who are not eligible for commission.
7. Increment the Salary of Clerk by 10% of basic salary.
8. List the total salary, maximum and minimum salary and average salary of the employees jobwise.
9. Delete the Employee whose name starts with P.
10. List the employees whose designation is “Clerk” and commission is > 500.
11. List employees belonging to department 20, 30, 40.

MIT-WPU
॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥

Thank You!