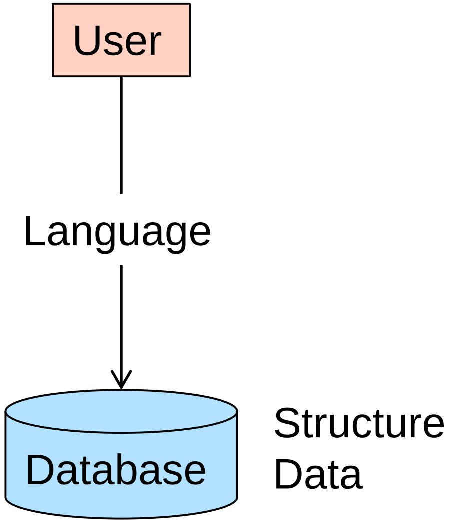
Database and Information Systems

Course Roadmap

Chapter 1	Introduction to Databases
Chapter 2	Integrity Constraints and ER Model
Chapter 3	Relational Databases and Schema Refinement
Chapter 4	Query Language
Chapter 5	Transaction and Concurrency Control
Chapter 6	Indexing

Introduction to Structure Query Language

- Structure Query Language (SQL): Domain specific and declarative language
  - Allow access and manipulate databases
  - Execute queries against a database
  - Retrieve data from a database
  - Insert records in a database
  - Update records in a database
  - Delete records from a database
  - Create new databases
  - Create new tables in a database
  - Create views in a database
  - Set permissions on tables and views



History

- In 1970, E.F. Codd develops relational database concepts: published a research paper while working at IBM San Jose Research Laboratory
- During 1974-1979, Sequel (Structured English Query Language) was created at IBM, renamed to SQL Later
- In 1979, Oracle markets first DB with SQL
- In 1986, ANSI SQL standards were released and updated later wards (1989, 1992, 1999, 2003, …)
- Most DBMS are SQL-99 compliant, with partial SQL-2003 compliant
- Now Database major players: Oracle, IBM, Microsoft, MySQL

Sub Language

- Data Definition Language
  - Used to define structure of a table
- Data Manipulation Language
  - Used to manipulate database records
- Data Query Language
  - Used to access required data from database tables
- Data Control Language
  - Used for transaction based operations and security

Data Definition Language

The SQL data-definition language (DDL) allows the specification of information about relations, including:

- The schema for each relation (or table)
- The type of values associated with each attribute
- The Integrity constraints
- Examples
  - Create Table
  - Drop Table
    - Delete table
  - Alter Table
    - Add and remove columns in a table
  - Truncate
    - Delete all the data inside a table
  - Integrity constraints: primary key, foreign key, alternate key
  - Rename

ID	Roll No	Department

Table: Student

Create Table Construct

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

  create table *r*

  $(A_1\ D_1,\ A_2\ D_2,\ \ldots,\ A_n\ D_n,$
  (integrity-constraint$_1$),
  ...,
  (integrity-constraint$_k$))

  - *r* is the name of the relation
  - each A_i is an attribute name in the schema of relation *r*
  - D_i is the data type of values in the domain of attribute A_i
- Example:

```
create table instructor (
                 ID              char(5),
                 name            varchar(20),
                 dept_name       varchar(20),
                 salary          numeric(8,2)
                 primary key (ID)
                 foreign key (dept_name) references dept (dept_name));
```

Domain Types in SQL

- **char(n).** Fixed length character string, with user-specified length *n.*
- **varchar(n).** Variable length character strings, with user-specified maximum length *n.*
- **int.** Integer (a finite subset of the integers that is machine-dependent).
- **numeric(p,d).** Fixed point number, with user-specified precision of *p* digits, with *d* digits to the right of decimal point. (ex., **numeric**(3,1), allows 44.5 to be stores exactly, but not 444.5 or 0.32)

Integrity Constraints in Create Table

- Types of integrity constraints
  - **primary key** (A_1, ..., A_n)
  - **foreign key** (A_m, ..., A_n) **references** *r*
  - **not null**
- SQL prevents any update to the database that violates an integrity constraint.
- Example:

```
create table instructor (
        ID          char(5),
        name        varchar(20) not null,
        dept_name   varchar(20),
        primary key (ID),
        foreign key (dept_name) references dept (dept_name));
```

More Constraints in SQL

- Unique
  - No duplicate values in a column
- Default
  - salary int default 10000
- Check
  - Fix the domain
  - Example: check (age >50)
- Not Null
  - Mandatory value
- Primary Key
  - Unique + Not Null
  - Example: Student Roll No in Institute Database
- Foreign Key
  - For referential integrity

And a Few More Relation Definitions

-
  ```
  create table student (
          ID              varchar(5),
          name            varchar(20) not null,
          dept_name       varchar(20),
          tot_cred        numeric(3,0),
          primary key (ID),
          foreign key (dept_name) references dept (dept_name));
  ```

-
  ```
  create table takes (
          ID              varchar(5),
          course_id       varchar(8),
          sec_id          varchar(8),
          semester        varchar(6),
          year            numeric(4,0),
          grade           varchar(2),
          primary key (ID, course_id, sec_id, semester, year) ,
          foreign key (ID) references  student (ID));
  ```

Data Manipulation Language

- Data Manipulation Language (DML) is used to manipulate the data records
- **Insert**
  - **insert into** *Student* **values** (‘4', ‘Ram', ‘CS');
- **Delete**
  - Remove all tuples from the *student* relation
    - **delete from** *Student*
  - Remove student with ID 2
    - **delete from** *Student* where *ID = 2;*
- **Update**
  - Update a tuple from the *student* relation
    - update Student set Department = ‘CS' where ID = 3;

ID	Roll No	Department
1	Rahul	CS
2	Suresh	EE
3	Kesav	ME

Table: Student

Data Query Language

- Data Query Language (DQL) is used to access required data from database tables
- A typical SQL query has the form:

select $A_1, A_2, ..., A_n$
from $r_1, r_2, ..., r_m$
where P

  - A_i represents an attribute
  - R_i represents a relation
  - P is a predicate.
- The result of an SQL query is a relation.
- Other DQL commands
  - Group By
  - Having

Data Control Language

- Data Control Language (DCL) is used for transaction based operations and security
- Operations
  - Grant
    - Give privileges to a user over table
  - Revoke
    - Remove privileges from user over table
  - Rollback
    - If transaction is failed, rollback it
  - Commit
    - Transactions completed successfully, save in database
  - Save Point
    - Save some part of execution in DB

Alter Command in SQL

- Alter is use to change the schema or structure or relation or table
- Function of Alter command
  - Add columns
    - alter table student add address varchar (10);
  - Remove columns
    - alter table student drop column address;
  - Modify data type
    - alter table student modify ID varchar (10);
  - Add constraints
    - alter table student add primary key (name);
  - Remove constraints
    - alter table student drop primary key;
  - Rename column/table
    - alter table student rename column id to roll_no;
    - alter table student rename to stu;

Difference between Alter and Update

Alter	Update
DDL	DML
Make changes in relation or table structure	Make changes in data
alter table employee add address varchar (10);	update employee set salary = salary*2 where ID =1;

ID	name	salary
1	rahul	1000000
2	rohan	2000000
3	rakesh	3000000

Table: employee

Difference between Delete, Drop, and Truncate

Delete	**Drop**	**Truncate**
DML Command	DDL Command	DDL Command
Delete all rows. delete from Student;	Delete table structure. drop table Student;	Delete all rows. truncate table Student;
Can give condition. delete from Student where ID = 1;	No condition	No condition
Can rollback before commit. Use logs	No Rollback	No rollback
Slower	Faster	Faster

ID	**Roll No**	**Department**
1	Rahul	CS
2	Suresh	EE

Table: Student

Modification of the Database

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

Deletion

- Delete all instructors

```
delete from instructor;
```

- Delete all instructors from the Finance department

```
delete from instructor
where dept_name= 'Finance’;
```

- *Delete all tuples in the* ***instructor*** *relation for those instructors associated with a* ***department*** *located in the Watson building.*

```
delete from instructor
where dept name in (select dept name
                    from department
                    where building = 'Watson');
```

Insertion

- 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*);

Insertion (Cont.)

- Make each student in the Music department who has earned more than 144 credit hours an instructor in the Music department with a salary of 18,000 INR.

```
insert into instructor
    select ID, name, dept_name, 18000
   from   student
   where   dept_name = 'Music' and total_cred > 144;
```

- The **select from where** statement is evaluated fully before any of its results are inserted into the relation.

Updates

- Give a 5% salary raise to all instructors

```
update instructor
   set salary = salary * 1.05
```

- Give a 5% salary raise to those instructors who earn less than 70000

```
update instructor
    set salary = salary * 1.05
   where salary < 70000;
```

- Give a 5% salary raise to instructors whose salary is less than average

```
update instructor
set salary = salary * 1.05
where salary <  (select avg (salary)
                       from instructor);
```

Aggregate Functions

- Aggregate Functions
  - Max, Min, Count, Avg, Sum

E_id	E_name	Dept	Salary
1	Ram	HR	10000
2	Amrit	MRKT	20000
3	Ravi	HR	30000
4	Nitin	MRKT	30000
5	Varun	IT	50000
6	Sandy	TESTING	NULL

Table: Emp

- Max
  - Find maximum salary
    - Select Max(Salary) from Emp;
  - Find employee name who is getting maximum salary?

Aggregate Functions

- Aggregate Functions
  - Max, Min, Count, Avg, Sum

E_id	E_name	Dept	Salary
1	Ram	HR	10000
2	Amrit	MRKT	20000
3	Ravi	HR	30000
4	Nitin	MRKT	30000
5	Varun	IT	50000
6	Sandy	TESTING	NULL

Table: Emp

- Max
  - Find maximum salary
    - Select Max(Salary) from Emp;
  - Find employee name who is getting maximum salary: Use of Nested or SubQuery
    - Select E_name from Emp where Salary = (Select Max(Salary) from Emp); Here, inner query execute before outer query

Aggregate Functions

- Max
  - Select E_name from Emp where Salary = (Select Max(Salary) from Emp);

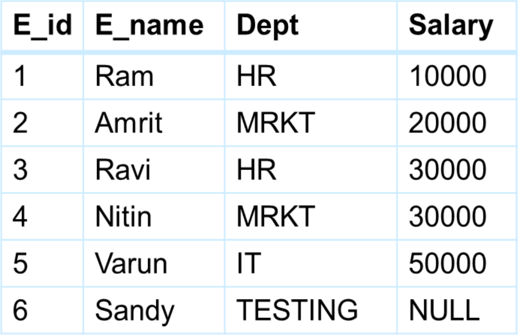
10000 = 50000 False

20000 = 50000 False

30000 = 50000 False

40000 = 50000 False

50000 = 50000 True

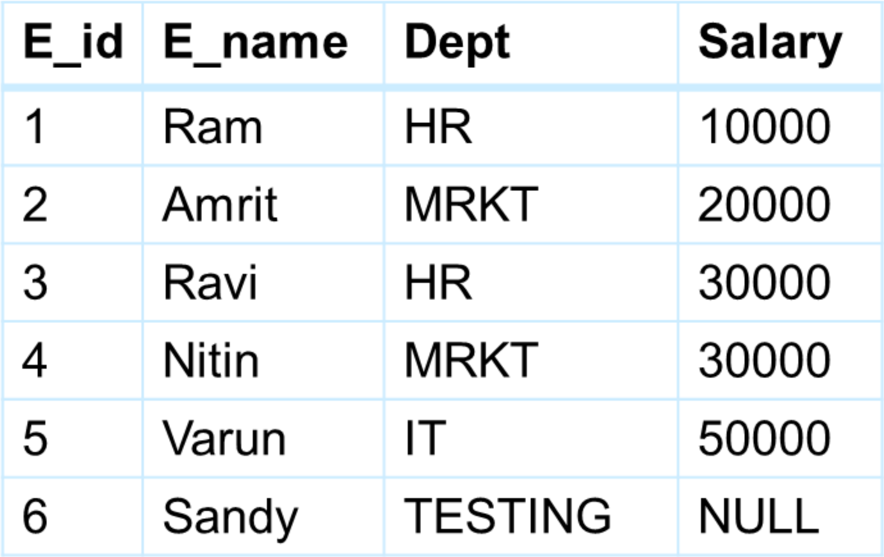


E_id	E_name	Dept	Salary
1	Ram	HR	10000
2	Amrit	MRKT	20000
3	Ravi	HR	30000
4	Nitin	MRKT	30000
5	Varun	IT	50000
6	Sandy	TESTING	NULL

Table: Emp

- Count: count total records or rows
  - Select Count(*) from Emp;
- Sum: sum on columns containing numerical values
  - Select Sum(Salary) from Emp;
- Avg: average on columns containing numerical values
  - Select Avg(Salary) from Emp;
  - It discords NULL value

Aggregate Functions



E_id	E_name	Dept	Salary
1	Ram	HR	10000
2	Amrit	MRKT	20000
3	Ravi	HR	30000
4	Nitin	MRKT	30000
5	Varun	IT	50000
6	Sandy	TESTING	NULL

Table: Emp

- Select Min(Salary) from Emp; ??
- Select Avg(Distinct(Salary)) from emp; ??
- Select Count(Salary) from Emp; ??
- Select Count(Distinct(Salary)) from Emp; ??

In and Not In

- In and Not In are used when one value is compared with multiple values
- Examples
  - Find detail of employee whose address is either Delhi or Indore, or Pune
    - Select * from Emp where Address In (‘Delhi’, ‘Indore’, ‘Pune’);
  - Similarly Not In is also used
    - Select * from Emp where Address Not In (‘Delhi’, ‘Indore’, ‘Pune’);

Eid	Ename	Address
1	Ravi	Indore
2	Varun	Delhi
3	Nitin	Pune
4	Robin	Bangalore
5	Ammy	Indore

Table: Emp

SQL Queries and SubQueries

- Use of *IN* in sub-queries or nested queries
  - Find the name of employees who are working on a project
    - Select Ename from Emp where Eid in (Select Distinct Eid from Project);

Eid	Ename	Address
1	Ravi	Indore
2	Varun	Delhi
3	Nitin	Pune
4	Robin	Bangalore
5	Ammy	Indore

Table: Emp

Eid	Pid	Pname	Location
1	P1	IOT	Bangalore
5	P2	Big Data	Delhi
3	P3	Retail	Mumbai
4	P4	Android	Hyderabad

Table: Project

    - Select Ename from Emp where Eid **not in** (Select Distinct Eid from Project); ??
- Note: SQL executes innermost subquery first, then next level

SQL Queries and SubQueries

- Write a SQL query to find second highest salary from Emp Table
  - Select Max(Salary) from Emp where Salary <> (Select Max(Salary) from Emp);

E_id	E_name	Dept	Salary
1	Ram	HR	10000
2	Amrit	MRKT	20000
3	Ravi	HR	30000
4	Nitin	MRKT	30000
5	Varun	IT	50000

Table: Emp

- Write a SQL query to find employee name who is taking second highest salary
  - Select E_name from Emp where Salary = (Select Max(Salary) from Emp where Salary <> (Select Max(Salary) from Emp));

SQL Queries and SubQueries: Group By Clause

- Write a query to display all the department names along with number of employees working in that department
  - Select Dept, count(Dept) from Emp Group By(Dept);

Aggregate Function

E_id	E_name	Dept	Salary
1	Ram	HR	10000
2	Amrit	MRKT	20000
3	Ravi	HR	30000
4	Nitin	MRKT	30000
5	Varun	IT	50000

Table: Emp

HR
HR
MRKT
MRKT
IT

Group By Intermediate Result

HR	2
MRKT	2
IT	1

Group By with count

- Group by groups rows that have the same values
  - Can use aggregate functions with *group by*
- Example: Find branch-wise student names

SQL Queries and SubQueries: Having Clause

- Write a query to display all the department names where number of employees are less than two
  - Select Dept from Emp Group By(Dept) having count(*) <2;

E_id	E_name	Dept	Salary
1	Ram	HR	10000
2	Amrit	MRKT	20000
3	Ravi	HR	30000
4	Nitin	MRKT	30000
5	Varun	IT	50000

Table: Emp

  - Find the name of employee(s) who is/are working in the department where number of employees are less than two?

SQL Queries and SubQueries: Having Clause

- Write a query to display all the department names where number of employees are less than two
  - Select Dept from Emp Group By(Dept) having count(*) <2;

E_id	E_name	Dept	Salary
1	Ram	HR	10000
2	Amrit	MRKT	20000
3	Ravi	HR	30000
4	Nitin	MRKT	30000
5	Varun	IT	50000

Table: Emp

  - Now employee name also can be found using nested query
    - Select E_name from Emp where Dept in (Select Dept from Emp Group By(Dept) having count(*) <2);

Correlated SubQuery

- Subquery that uses value from outer query
  - Follows top to bottom approach
    - First row of outer query compares with all the rows of inner query
  - Called Synchronized Query
- Example
  - Find all employees detail who work in a department
    - Select * from Emp where exists (Select * from Dept where Emp.eid= Dept.eid)

Returns true or false

Eid	Name	Address
1	A	Delhi
2	B	Pune
3	A	Chd
4	B	Delhi
5	C	Pune
6	D	Mumbai
7	E	Hyd

Table: Emp

Did	Dname	Eid
D1	HR	1
D2	IT	2
D3	MRKT	3
D4	Testing	4

Table: Dept

Can write query using *in* ?

Exist and Not Exist SubQueries

- Find the detail of employee who is working on at least one project
  - Select * from Emp where exists (Select Eid from Project where Emp.Eid = Project.Eid)
- Find the detail of employee who is not working on any project
  - Select * from Emp where not exists (Select Eid from Project where Emp.Eid = Project.Eid)

<u>Eid</u>	**Ename**	**Address**
1	Ravi	Indore
2	Varun	Delhi
3	Nitin	Pune
4	Robin	Bangalore
5	Ammy	Indore

Table: Emp

Eid	**<u>Pid</u>**	**Pname**	**Location**
1	P1	IOT	Bangalore
5	P2	Big Data	Delhi
3	P3	Retail	Mumbai
4	P4	Android	Hyderabad

Table: Project

Correlated SubQuery – Nth Highest Salary

- Find N-th highest salary
  - Will use correlated nested query, it processes top to bottom
  - Select ID, Salary from Emp e1 where N-1 = (Select count (distinct Salary) from Emp e2 where e2.Salary > e1.Salary)
  - Where e1 and e2 are alias of Emp Table

ID	Salary
1	10000
2	20000
3	20000
4	30000
5	40000
6	50000

Table: Emp e1

ID	Salary
1	10000
2	20000
3	20000
4	30000
5	40000
6	50000

Table: Emp e2

Difference between Joins, Nested SubQuery and Correlated SubQuery

❑ Example: Find the detail of employee who is working on any department

Nested SubQuery	**Correlated SubQuery (or Correlated SubQuery)**	**Joins**
Bottom Up Approach	Top Down Approach	Cross Product + Condition
Select * from Emp where eid in (Select eid from Dept);	Select * from Emp where exits (Select eid from Dept where Emp.eid = Dept.eid);	Select Emp.eid, Emp.name from Emp , Dept where Emp.eid = Dept.eid;

eid	**name**
1	A
2	B
3	C
4	D
5	E

Table: Emp

dept no	**name**	**eid**
D1	IT	1
D2	HR	2
D3	MRKT	3

Table: Dept

References

- Silberschatz, Abraham, Henry F. Korth, and Shashank Sudarshan. *Database system concepts*. Vol. 6. New York: McGraw-Hill, 1997.
- Ramez Elmasri, Shamkant B. Navathe. *Fundamentals of Database Systems.* Edition 6. Pearson, 2010.