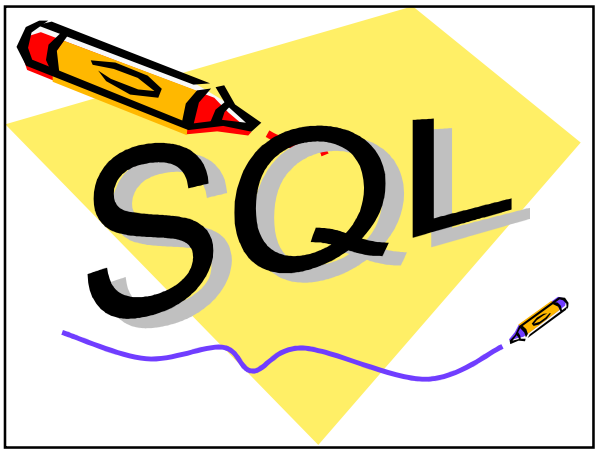

Outline

- Introduction to SQL
- Basic structure of SQL commands
- Data Definition
- Data Manipulation
- Aggregation

Introduction to SQL

- SQL is a transform-oriented language with two major components:
- the DDL for defining the database structure and
- the DML for retrieving and updating data.

Introduction to SQL

- SQL does not contain flow control commands.
- must be implemented using a programming or job-control language,
- or interactively by the decisions of the user.

Introduction to SQL

- SQL is relatively easy to learn.
- SQL is a nonprocedural language,
- you specify what information you require,
- rather than how to get it.

Introduction to SQL

- An ISO standard exists for SQL,
- making it both the formal and de facto standard language for relational databases.
- The most popular and widely implemented is referred to as SQL2 or SQL/92.

Basic structure of SQL commands

- SQL statement consists of reserved words and user-defined words.
- Reserved words are a fixed part of SQL and must be spelt exactly as required and cannot be split across lines.
- User-defined words are made up by user and represent names of various database objects such as relations, columns and views.

Basic structure of SQL commands

- Most components of an SQL statement are case insensitive
- SQL statements are more readable with indentation and lineation.

TABLE DEFINITION-

- The syntax for creating a table consists of an ordered set of attributes and a (possibly empty) set of constraints.
- CREATE TABLE table_name

  (col_name data_type [NULL | NOT NULL] [,...])

TABLE DEFINITION- example

- CREATE TABLE Employee

  (

  RegNo CHARACTER(6) PRIMARY KEY,

  FirstName CHARACTER(20) NOT NULL,

  Surname CHARACTER(20) NOT NULL,

TABLE DEFINITION- CONT

Dept CHARACTER (15) REFERENCES Department(DeptName) ON DELETE SET NULL

ON UPDATE CASCADE,

Salary NUMERIC(9) DEFAULT 0,

City CHARACTER(15),

UNIQUE (Surname,FirstName)

)

DROP DEFINITION

- A Table can be deleted from a database using the command "drop".
- The following syntax is used.
  - DROP TABLE name [RESTRICT | CASCADE];

DROP DEFINITION CONT:

- With RESTRICT (default), Table must be empty or operation fails.
- With CASCADE, SQL drops all dependent objects — and objects dependent on these objects

ALTER DEFINITION

- ALTER (alter domain ..., alter table ...)
- For example the commande:

  ALTER TABLE Department
  ADD COLUMN NoOfOffices NUMERIC(4);

Data Manipulation

- A query in SQL can consist of up to six clauses,
- but only the first two are mandatory.
- SELECT [DISTINCT | ALL]
  {* | [column_expression [AS new_name]] [,...] }

Data Manipulation

FROM table_name [alias] [, ...]
[WHERE condition]
[GROUP BY column_list]
[HAVING condition]
[ORDER BY column_list]

Data Manipulation CONT:

- SELECT specifies which columns are to appear in output.
- FROM specifies table(s) to be used.
- WHERE filters rows.

Data Manipulation CONT:

- GROUP BY forms groups of rows with same column value.
- HAVING filters groups subject to some condition.
- ORDER BY specifies the order of the output.

Employee Table

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000.00	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000.00	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000.00	B005

Example 1 -All Columns, All Rows

- List full details of all staff.
  - SELECT staffNo, fName, lName, address, position, sex, DOB, salary, branchNo
    FROM Employee;

All Columns, All Rows

- Can use * as an abbreviation for 'all columns':
- SELECT *
  FROM Employee;

Example 2 - Specific Columns, All Rows

- Produce a list of salaries for all staff, showing only staff number, first and last names, and salary.
- SELECT staffNo, fName, lName, salary
  FROM Employee;

Example 3 - Use of DISTINCT

- List the branch numbers.
- SELECT branchNo
- FROM Employee;

Example 3 Use of DISTINCT

- Use DISTINCT to eliminate duplicates:
- SELECT DISTINCT branchNo
  FROM Employee;

Example 4 - Specific Columns, Specific Rows.

- Find the salaries of employees named White.
- SELECT Salary as Remuneration
  FROM Employee
  WHERE Surname = 'White';

Example 5- All Columns, Specific Rows.

- Find all the information relating to employees named White.
- SELECT *
  FROM Employee
  WHERE Surname = 'Employee';

Example 6 - calculated field

- Produce a list of monthly salaries for all employees, showing staff number, first and last names, and salary details
- SELECT Snumber, Fname, Lname, Salary / 12 AS MonthlySalary
  FROM Employee;

Example 7 Comparison Search Condition

- List all staff with a salary greater than 10,000.
- SELECT staffNo, fName, lName, position, salary
  FROM Staff
  WHERE salary > 10000;

Example 8 Compound Comparison Search Condition

- Find all employees who are managers or supervisors.
- SELECT *
  FROM Employee
  WHERE position = 'Manager' OR position = 'Supervisor';

Could also write:

- SELECT staffNo, fName, lName, position
  FROM Staff
  WHERE position IN ('Manager', 'Supervisor');

Example 9 Range Search Condition

- List all Employees with a salary between 20,000 and 30,000.
- SELECT staffNo, fName, lName, position, salary
  FROM Employee
  WHERE salary BETWEEN 20000 AND 30000;

Could also write:

- SELECT staffNo, fName, lName, position, salary
- FROM Employee
  WHERE salary>=20000 AND salary <= 30000;

Simple join query

- Find the names of the employees and the cities in which they work.
- SELECT Employee.FirstName, Employee.Surname, Department.City
  FROM Employee, Department
  WHERE Employee.Dept = Department.DeptName;

Alternative JOIN Constructs

- SQL provides alternative ways to specify joins:
- FROM Employee E JOIN Department D ON E.Dept = D.DeptName
- FROM Employee JOIN Department USING DeptName
- FROM Employee NATURAL JOIN Department
- In each case, FROM replaces original FROM and WHERE. However, first produces table with two identical DeptName columns

Three Table Join

- For each branch, list staff who manage properties, including city in which branch is located and properties they manage.
- SELECT b.branchNo, b.city, s.staffNo, fName, lName, propertyNo
  FROM branch b, staff s, property_for_rent p
  WHERE b.branchNo = s.branchNo AND s.staffNo = p.staffNo
  ORDER BY b.branchNo, s.staffNo, propertyNo

Three Table Join

- Alternative formulation for FROM and WHERE:
- FROM (branch b JOIN Staff s USING branchNo) AS bs JOIN PropertyForRent p USING staffNo

Outer Joins

- To include unmatched rows in result table, use an Outer join.

Left Outer Join

- List branches and properties that are in
  same city along with any unmatched branches.
- SELECT b.*, p.*
  FROM Branch1 b LEFT JOIN
  PropertyForRent1 p ON b.bCity = p.pCity;

Right Outer Join

- List branches and properties in same city and any unmatched properties.
- SELECT b.*, p.*
  FROM Branch1 b RIGHT JOIN
  PropertyForRent1 p ON b.bCity = p.pCity;

Full Outer Join

- List branches and properties in same city and any unmatched branches or properties.
- SELECT b.*, p.*
  FROM Branch1 b FULL JOIN
  PropertyForRent1 p ON b.bCity = p.pCity;

Aggregation

- ISO standard defines five aggregate functions. These are:
  - COUNT returns number of values in a specified column.
  - SUM returns sum of values in a specified column.

Aggregation

- AVG returns average of values in a specified column.
- MIN returns smallest value in a specified column.
- MAX returns largest value in a specified column.

Aggregate functions

- Each operates on a single column of a table and return single value.
- COUNT, MIN, and MAX apply to numeric and non-numeric fields, but SUM and AVG may be used on numeric fields only.
- Apart from COUNT(*), each function eliminates nulls first and operates only on remaining non-null values.

Aggregate functions

- COUNT(*) counts all rows of a table, regardless of whether nulls or duplicate values occur.
- Can use DISTINCT before column name to eliminate duplicates.
- DISTINCT has no effect with MIN/MAX ***(it eliminates duplicates itself),***
- but may have with SUM/AVG, e.g.-
  - ***select avg(sal) from emp ≠ select avg(distinct sal) from emp;***

Aggregate functions

- Aggregate functions can be used only in SELECT list and in HAVING clause.

Aggregation examples

- The Count function
  - How many properties cost more than £350 per month for rent?
- SELECT Count(*) AS count
  FROM property
  WHERE property.Rent > 350;

Aggregation examples

- The Max, Min and Avg function
  - Find the minimum, maximum and average staff salary.
- SELECT MIN(salary) AS MIN, MAX(salary) AS MAX, AVG(salary) AS AVG
  FROM staff;

Aggregation examples

- Using the Group By clause
  - Find the number of staff working in each branch and the total of their salaries.
- SELECT bno, COUNT(sno) AS count, SUM(salary) AS sum
  FROM staff
  GROUP BY bno
  ORDER BY bno;

Aggregation examples

- Using predicates on grouping results
  - For each branch office with more than one member of staff, find the number of staff working in each branch and the sum of their salaries.

Aggregation examples

- SELECT bno, COUNT(sno) AS count, SUM(salary) AS sum
  FROM staff
  GROUP BY bno
  HAVING COUNT(SNO) > 1;

Multiple Grouping Columns

- Find number of properties handled by each staff member.
- SELECT s.branchNo, s.staffNo, COUNT(*) AS count
  FROM Staff s, PropertyForRent p
  WHERE s.staffNo = p.staffNo
  GROUP BY s.branchNo, s.staffNo
  ORDER BY s.branchNo, s.staffNo;

Subqueries

- Some SQL statements can have a SELECT embedded within them.
- A subselect can be used in WHERE and HAVING clauses of an outer SELECT, where it is called a *subquery* or *nested query*.
- Subselects may also appear in INSERT, UPDATE, and DELETEs.

Subquery with Equality

- List staff who work in branch at '163 Main St'.
- SELECT staffNo, fName, lName, position
  FROM Staff
  WHERE branchNo = (SELECT branchNo
  FROM Branch
  WHERE street = '163 Main St');

Subquery with Equality

- Inner SELECT finds branch number for branch at '163 Main St' ('B003').
- Outer SELECT then retrieves details of all staff who work at this branch.
- Outer SELECT then becomes:
- SELECT staffNo, fName, lName, position
  FROM Staff
  WHERE branchNo = 'B003';

Subquery with Aggregate

- List all staff whose salary is greater than the average salary, and show by how much.
- SELECT staffNo, fName, lName, position, salary – (SELECT AVG(salary) FROM Staff)
  As SalDiff
  FROM Staff
  WHERE salary > (SELECT AVG(salary) FROM Staff);

Subquery with Aggregate

- Cannot write 'WHERE salary > AVG(salary)'
- Instead, use subquery to find average salary (17000), and then use outer SELECT to find those staff with salary greater than this:
- SELECT staffNo, fName, lName, position, salary – 17000 As salDiff
  FROM Staff
  WHERE salary > 17000;

Subquery Rules

- ORDER BY clause may not be used in a subquery (although it may be used in outermost SELECT).
- Subquery SELECT list must consist of a single column name or expression, except for subqueries that use EXISTS.

Subquery Rules

- By default, column names refer to table name in FROM clause of subquery. Can refer to a table in FROM using an *alias*.
- When subquery is an operand in a comparison, subquery must appear on right-hand side.
- A subquery may not be used as an operand in an expression.

Questions