Database Management Systems

SQL Query Language (1)

Topics

- Introduction
  - SQL History
- Domain Definition
  - Elementary Domains
  - User-defined Domains
- Creating Tables
- Constraint Definition
- INSERT Query
- SELECT Query
- Attribute Expressions
- Logical Expressions in Queries

SQL

- SQL stands for Structured Query Language
- First proposed by IBM Research 1974
- First implementation SQL/DS, IBM 1981
- Most DBMS systems support base functionality but add some new properties

Domains

- Domains specify the content type of attributes
  - e.g. Attribute *"Employee name"* gets its value from the domain of ***strings*** *(Each name is a string)*

- Two categories of domains are:
  - Elementary (predefined by the standard)
  - User-defined

Elementary Domains - Character

- Character domain is used with single character or string attributes
- Strings may have variable length
- Syntax:
  - ***character*** or ***char*** to define single character attributes
  - ***character (n)*** or ***char (n)*** defines a fixed length string
  - ***character varying*** or ***varchar*** defines a variable length string

Elementary Domains – Exact Values

- Exact numeric domains are used with exact values, integer, or numbers with a fractional part
- Four types are:
  - numeric [(*Precision [, Scale])] where precision is the total number of digits and scale is the number of digits after decimal point*
    - *e.g. numeric (5,2) shows numbers like 455.12*
  - decimal [(*Precision [, Scale])] (same as numeric)*
  - integer
  - smallint

Elementary Domains – Real Values

- Real Value Domains are used for non-exact numeric values
- Real value domains are based on a floating point representation
- Three types available
  - ***float [(Precision)]*** *e.g. float(6) (total number of digits is 6)*
  - ***double precision*** (two times the precision of float)
  - ***real***

Elementary Domains – Date, Time, and Intervals

- Date and Time are used for temporal instant attributes
  - e.g. Date *(stores day, month and year values)*
  - *A date value is given as*
    - MM/DD/YY or MM/DD/YYYY in USA
    - *YY.MM.DD or YYYY.MM.DD in ANSI (and more..)*
  - Time *(stores hour, minute, and second values)*
  - *Time value is given as hh:mm:ss*

Intervals

- Temporal Interval domains are defined by

  Interval *First* ***To*** *Last*

  e.g. Interval 1990 To   1999 (year)

  Interval 1985.1 To 2009.10 (year and month)

  Interval 11:20:10 To 12:25:40 (hour, minute and second)

User-defined Domains

- A user-defined domain is given by
  - name
  - elementary domain
  - default value
- Syntax:
  - ***create domain*** *DomainName* **as** *elementaryDomain*

    [***default*** *DefaultValue]*

  e.g. create domain StudentName as char(30)

  e.g. create domain money as numeric(19,2) default 0

Table Definition

- An SQL table consists of
  - an ordered set of attributes
  - an optional set of constraints
- ***create table*** statement
  - defines a relation schema
  - creates an empty table

Create Table

- Syntax:

create table *TableName*
(
AttributeName$_1$ *Domain [DefaultValue] [Constraints],*
AttributeName$_2$ *Domain [DefaultValue] [Constraints] ,*
...
AttributeName$_n$ *Domain [DefaultValue] [Constraints]*
[*Other Constraints]*
)

Example

```
create table Employee
(
      RegNo character(6) primary key,
      FirstName character(20) not null,
      Surname character(20) not null,
      Dept character (15)
 )
```

Primary Key Constraint

- Primary key constraint is defined by ***primary key*** keyword as
  - After defining the attribute in the table (as a constraint)
  - At the end of create table statement. This method is used when primary key has more than one attribute in it.

Example 1

```
 CREATE   TABLE   Persons
(
     P_Id integer PRIMARY KEY,
     LastName varchar NOT NULL,
     FirstName varchar,
     Address varchar,
     City varchar
)
```

Example 2

```
 CREATE    TABLE Persons
(
      P_Id integer NOT NULL,
      LastName varchar(255) NOT NULL,
      FirstName varchar(255),
      Address varchar(255),
     City varchar(255),
     PRIMARY KEY (P_Id,LastName)
)
```

Uniqueness Constraint

- We may define an attribute or a group of attributes as unique. This constraint will not allow repeated values for that attribute(s)

- Syntax
  - Use ***unique*** after defining the attribute
  - Use ***unique(*** *attribute_name* ***)*** at the end of table definition.

Example

```
create table Employee
(
    RegNo character(6) primary key,
    FirstName character(20) not null,
    Surname character(20) not null,
    Dept character (15)
    Salary numeric(9) default  0,
    City character(15),
    unique(Surname, FirstName)
)
```

Foreign Key Constraint

- Foreign keys are defined using ***references*** keyword
- Syntax ***references*** External_Table(Attribute)

Example

```
create table Employee
(
  RegNo character(6) primary key,
  FirstName character(20) not null,
  Surname character(20) not null,
  Dept character (15)
  references Department(DeptName),
  Salary numeric(9) default 0,
  City character(15),
  unique(Surname,FirstName)
)
```

NULL Constraint

- An attribute can be defined to always have a value using ***not null*** constraint

- e.g.

```
FirstName character(20) not null,
Surname character(20) not null unique,
```

INSERT Query

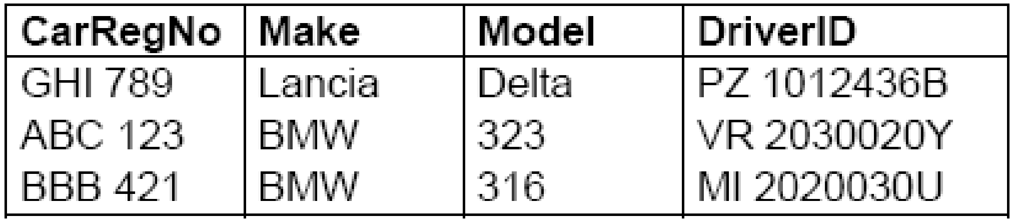
- INSERT is used to add a record (tuple) to the table.
- Syntax
  - ***Insert Into*** table_name [(Attributes)] ***values*** (list of values)

  e.g.

```
insert into Department(DeptName, City)
            values('Production','Atlanta')
```

Example

Car:

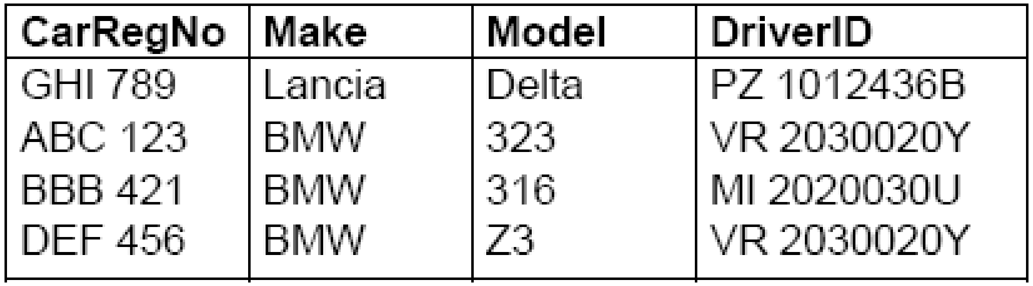


CarRegNo	Make	Model	DriverID
GHI 789	Lancia	Delta	PZ 1012436B
ABC 123	BMW	323	VR 2030020Y
BBB 421	BMW	316	MI 2020030U

Insert into Car (CarRegNo, Make, Model, DriverID)

Values (‘DEF 456’, ‘BMW’, ‘Z3’, ‘VR 2030020Y’)

Car:



CarRegNo	Make	Model	DriverID
GHI 789	Lancia	Delta	PZ 1012436B
ABC 123	BMW	323	VR 2030020Y
BBB 421	BMW	316	MI 2020030U
DEF 456	BMW	Z3	VR 2030020Y

INSERT Query Properties

- The ordering of the attributes (if present) and of values is important (first value with the first attribute, and so on)
- If *AttributeList is omitted, all the relation attributes are* considered, in the order in which they appear in the table definition
- If *AttributeList does not contain all the relation attributes, to the* remaining attributes it is assigned the default value (if defined) or the null value

SELECT Query

- Select is used for retrieving records from tables
- The simplest form of Select query is:

```
SELECT attribute_list
  FROM  Table_name
  WHERE  Condition
```

Condition eliminates some of the records from the list.

Example

CarRegNo	Make	Model	DriverID
GHI 789	Lancia	Delta	PZ 1012436B
ABC 123	BMW	323	VR 2030020Y
BBB 421	BMW	316	MI 2020030U
DEF 456	BMW	Z3	VR 2030020Y

```
SELECT CarRegNo, DriverID
 FROM Car
 WHERE  Make = ‘BMW’
```

CarRegNo	DriverID
ABC 123	VR 2030020Y
BBB 421	MI 2020030U
DEF 456	VR 2030020Y

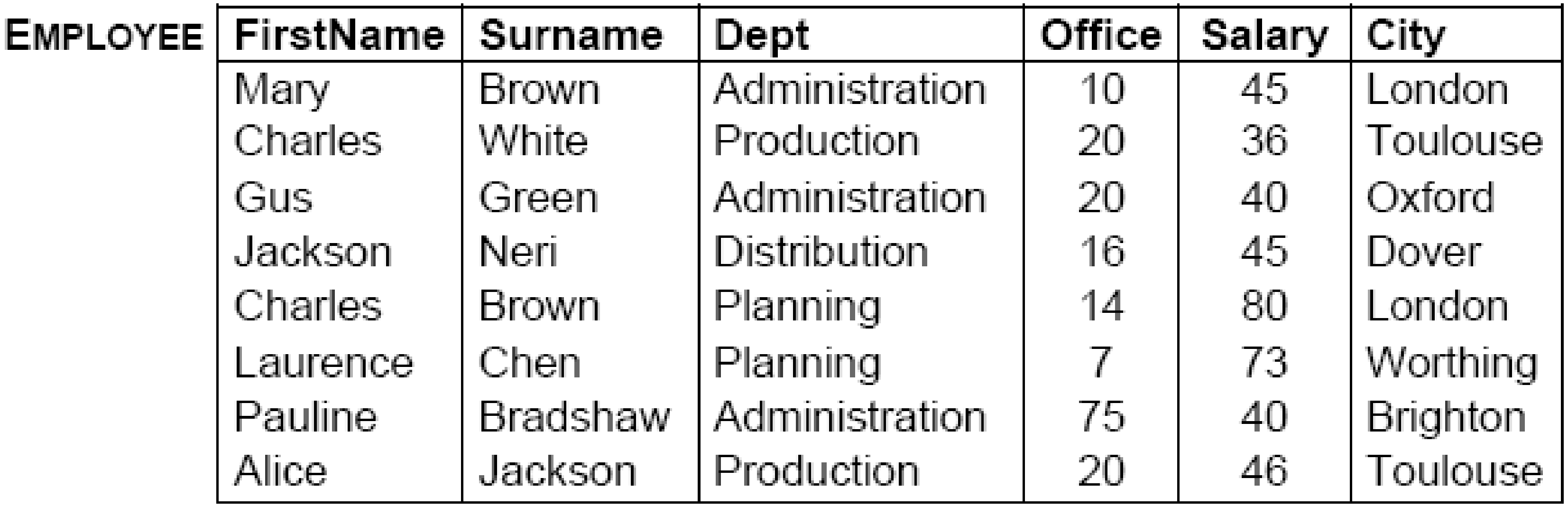
Selecting All Attributes

- To select all attributes we can use * in place of the attribute list

```
SELECT  *
FROM  table_name
WHERE  condition
```

SELECT * is the same as select operation defined in relational algebra

Example



EMPLOYEE

FirstName	Surname	Dept	Office	Salary	City
Mary	Brown	Administration	10	45	London
Charles	White	Production	20	36	Toulouse
Gus	Green	Administration	20	40	Oxford
Jackson	Neri	Distribution	16	45	Dover
Charles	Brown	Planning	14	80	London
Laurence	Chen	Planning	7	73	Worthing
Pauline	Bradshaw	Administration	75	40	Brighton
Alice	Jackson	Production	20	46	Toulouse

Example (Cont.)

Find the salaries of employees named Brown:

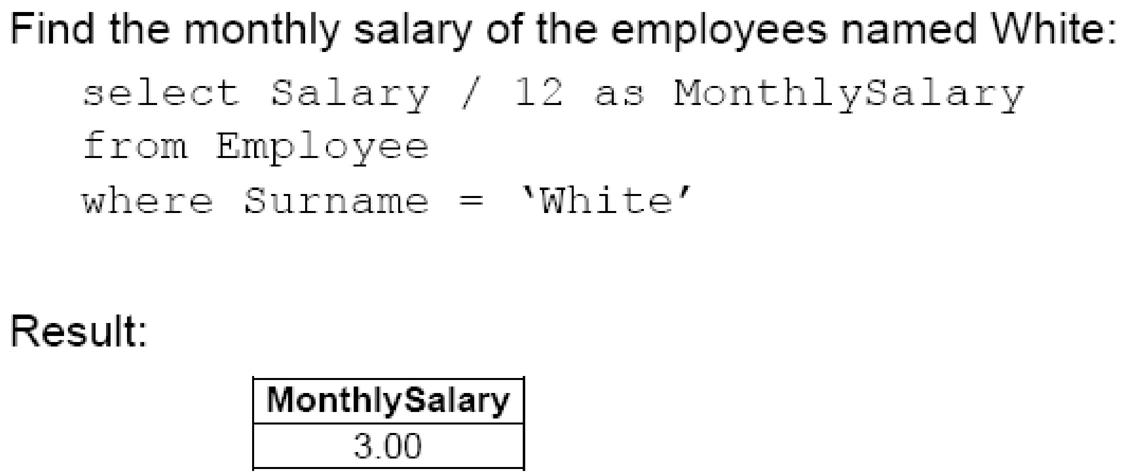
```
select Salary
from Employee
where Surname = 'Brown'
```

Result:

Salary
45
80

Attribute Expressions

- Attributes can be written as expressions. In this case a name can be assigned to the resulted table attribute using ***as*** *new_name*.



Find the monthly salary of the employees named White:

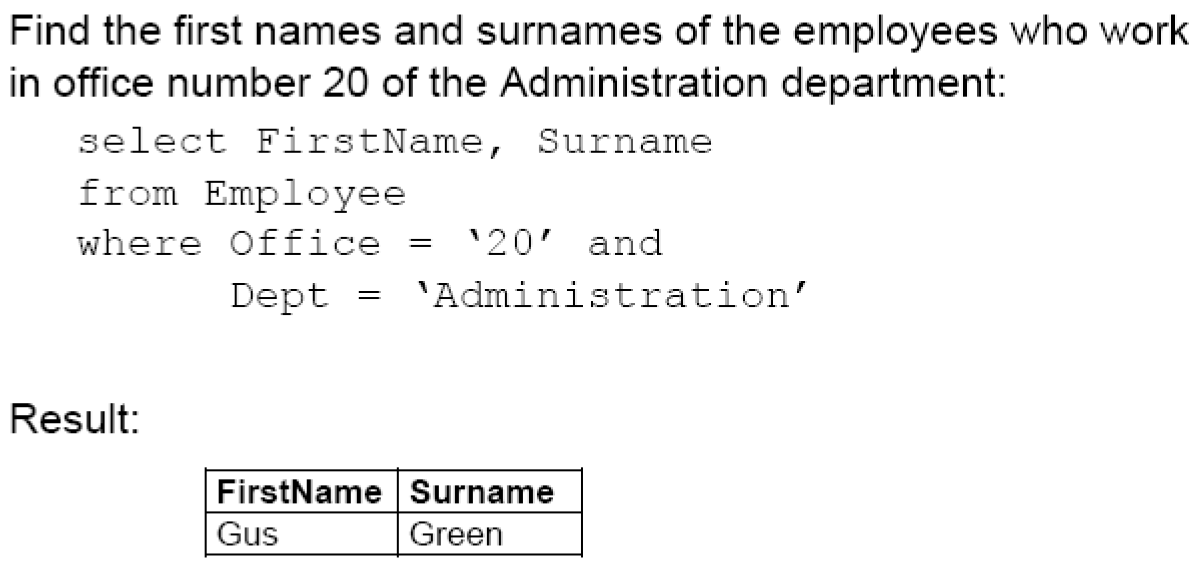
```
select Salary / 12 as MonthlySalary
from Employee
where Surname = 'White'
```

Result:

MonthlySalary
3.00

Logical Expressions in SELECT Query

- The condition part of a select query can be written using logical expressions with AND, OR and NOT



Find the first names and surnames of the employees who work in office number 20 of the Administration department:

```
select FirstName, Surname
from Employee
where Office = '20' and
      Dept = 'Administration'
```

Result:

FirstName	Surname
Gus	Green

Example

Find the first names and surnames of the employees who work in either the Administration or the Production department:

```
select FirstName, Surname
from Employee
where Dept = 'Administration' or
      Dept = 'Production'
```

Result:

FirstName	Surname
Mary	Brown
Charles	White
Gus	Green
Pauline	Bradshaw
Alice	Jackson

Example

- Find the names and surnames of all employees working in Administration department in London and earning more than 45, or working in Production department in Oxford and earning less than 30.

Solution

```
 SELECT Name, Surname
 FROM Employee
 WHERE  (Dept='Administration' AND City
='London'
                  AND Salary > 45 )
                         OR
                (Dept='Production' AND City
='Oxford'
                  AND Salary < 30 )
```

Summary

- SQL query language is designed to write queries in relational databases
- Each attribute is defined by using a domain
- CREATE TABLE is used to create a new table
- INSERT is used to add records to a table
- SELECT is used to retrieve data from a table

Questions?