SQL – Simple Queries

Chapter 3.1

V3.01

Introduction

- SQL is the Structured Query Language
- It is used to interact with the DBMS (database management system)
- SQL can
  - Create Schemas in the DBMS
  - Alter Schemas
  - Add data
  - Remove data
  - Change data
  - Access Data

DSL

- SQL is a Data Sub Language
- This is a combination of two languages
  - DDL – Data Definition Language
  - DML – Data Manipulation Language

- The main way of accessing data is using the DML command SELECT.
- The abilities of the SELECT command forms the majority of this material on SQL

Database Models

A data model comprises

- a data structure
- a set of integrity constraints
- operations associated with the data structure

Examples of data models include:

- hierarchic
- network
- Relational (E. F. Codd)

Relational Databases

The relational data model comprises:

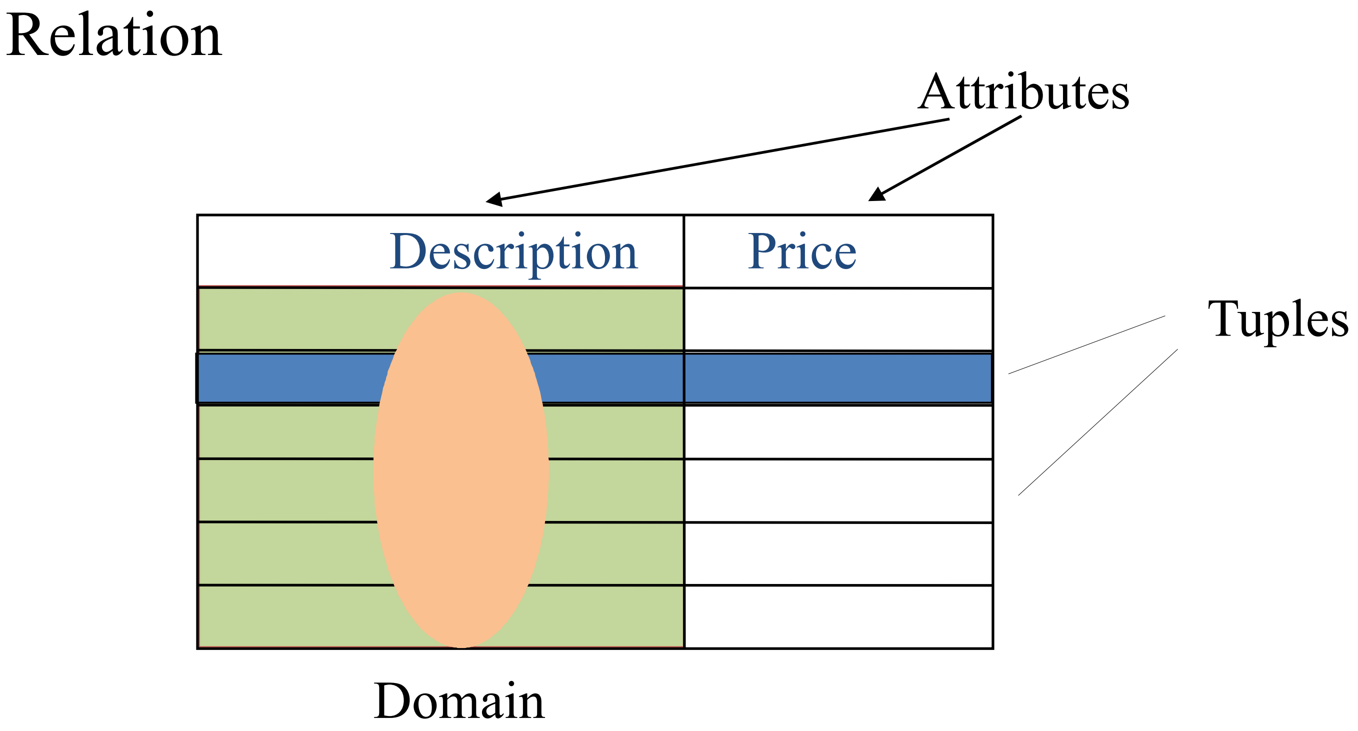
- relational data structure
- relational integrity constraints
- relational algebra or equivalent (SQL)
  - SQL is an ISO language based on relational algebra – the operations
  - relational algebra is a mathematical formulation

Relational Data Structure

A relational data structure is a collection of tables, or relations.

- A relation is a collection of rows or tuples
- A tuple is a collection of columns or attributes
- A domain is a pool of values from which the actual attribute values are taken.

Relational Structure cont

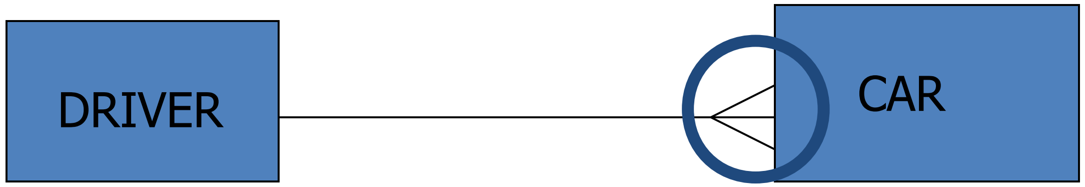


Domain and Integrity Constraints

- Domain Constraints
  - limit the range of values of an attribute
  - specify uniqueness and 'nullness' of an attribute
  - specify a default value for an attribute when no value is provided.
- Entity Integrity
  - every tuple is uniquely identified by a unique non-null attribute, the primary key.
- Referential Integrity
  - rows in different tables are correctly related by valid key values ('foreign' keys refer to primary keys).

Example Database

- In order to better understand SQL, all the example queries make use of a simple database.
- The database is formed from 2 tables, CAR and DRIVER.
- Each car may be owned by a DRIVER.
- A DRIVER may own multiple CARs.



DRIVER

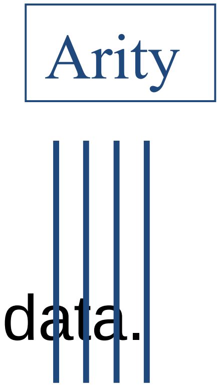
<u>**NAME**</u>	**DOB**
Jim Smith	11 Jan 1980
Bob Smith	23 Mar 1981
Bob Jones	3 Dec 1986

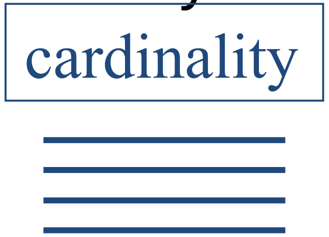
CAR

<u>**REGNO**</u>	MAKE	COLOUR	PRICE	***OWNER***
F611 AAA	FORD	RED	12000	Jim Smith
J111 BBB	SKODA	BLUE	11000	Jim Smith
A155 BDE	MERCEDES	BLUE	22000	Bob Smith
K555 GHT	FIAT	GREEN	6000	Bob Jones
SC04 BFE	SMART	BLUE	13000	

- Each column holds data of a particular type
  - Integer, string, decimal, blobs
  - The range of values can be further constrained
- If a column in a row contains no data, it is NULL.
- Null can indicate no possible value, or unavailable data.

- All rows (tuples) must differ from each other in some way
- Cardinality is the number of rows of a table
- Arity is the number of columns of a table





Edinburgh Napier UNIVERSITY

Primary Keys & Entity Integrity

- A Primary Key is a group of one **or more** columns which, when taken together, is **unique** in the table

- No **part** of a primary key can be NULL.

- In our example,
  - DRIVER: the primary key is NAME
  - CAR: the primary key is REGNO
- In our example this means that no two drivers can have the same name. In the real world this would be a problem, but this is just an example.

Referential Integrity

- Note that there is a link between CAR and DRIVER via the attribute OWNER.
- If there is a value in OWNER, then this value must also appear somewhere in DRIVER (attribute NAME).
- If you change a driver's NAME in DRIVER, you must make sure the same change is made in OWNER of CAR.
- **The DBMS enforces the rules!**
- If you try to break the rules the DBMS reports the problem as a REFERENTIAL INTEGRITY error.

SQL Basics

Edinburgh Napier UNIVERSITY

- Basic SQL statements include
  - CREATE – a data structure DDL
  - SELECT – read one or more rows from a table DML
  - INSERT – one of more rows into a table DML
  - DELETE – one or more rows from a table DML
  - UPDATE – change the column values in a row DDL
  - DROP – a data structure DDL
  - ALTER – a data structure DDL

- In this lecture the focus is on SELECT.

Simple SELECT

- SELECT column FROM tablename
- SELECT column1, column2, column3, … FROM tablename1, tablename2, …
- SELECT * from tablename

  e.g. SELECT * from CAR;   -- gives

REGNO	MAKE	COLOUR	PRICE	OWNER
F611 AAA	FORD	RED	12000	Jim Smith
J111 BBB	SKODA	BLUE	11000	Jim Smith
A155 BDE	MERCEDES	BLUE	22000	Bob Smith
K555 GHT	FIAT	GREEN	6000	Bob Jones
SC04 BFE	SMART	BLUE	13000	

SELECT regno from CAR;

REGNO
F611 AAA
J111 BBB
A155 BDE
K555 GHT
SC04 BFE

SELECT colour, owner from CAR;

COLOUR	OWNER
RED	Jim Smith
BLUE	Jim Smith
BLUE	Bob Smith
GREEN	Bob Jones
BLUE	

Formatting

- SPACES do not matter
- NEWLINES do not matter
- Good practice to put **;** at the end of the query.
- CaSE (except between single quotes) does not matter.
- The following are all valid:

```
SELECT REGNO FROM CAR;
SElecT regno
   From car
;
SELECT Regno
FROM Car;
```

Comments

- To give you the ability to make notes in queries you are allowed to have comments.
- Comments are not executed (they are ignored)
- A comment starts with -- and ends with a newline
- They are only permitted within a query.

```
SELECT regno -- The registration number
FROM car -- The car storage table
;
```

SELECT filters

- You can have rules in your queries
- These rules are tested for each row your query produces
- If the rule is true, the row is displayed
- If the rule is false, the row is not displayed
- The rule starts with WHERE

```
SELECT columns
FROM table
WHERE rule
```

Simple Rule

- A simple rule might be to look for a car with a colour of RED.
- The rule would be *colour = 'RED'*

SELECT regno FROM CAR from CAR

REGNO
F611 AAA
J111 BBB
A155 BDE
K555 GHT
SC04 BFE

SELECT regno

WHERE colour = 'RED'

REGNO
F611 AAA

Note

- Things between quotes are CASE SENSITIVE.
- ‘RED’ is not the same as ‘Red’ or ‘red’
- Rules which mention fields car
whether the fields appear on t T
line or not.

REGNO
F611 AAA

```
SELECT regno from CAR
WHERE colour = 'RED'
```

Comparisons

- Valid comparisons include =,!=,<>,<,<=,>,>=

– Colour = 'RED'	The colour must be red
– Colour != 'RED'	The colour is not red
– Colour <> 'RED'	Same result as !=
– Price > 10000	More than 10000
– Price >= 10000	More than or equal to 10000
– Price < 10000	Cheaper than 10000
– Price <=10000	Cheaper or the same as 10000

- Numbers – You may say '10000' **or** 10000 in Oracle SQL
- "Strings" and dates must always have quotes…

DATE

- You can use all the normal comparators with dates.

```
SELECT name,dob
from driver
```

NAME	DOB
Jim Smith	11 Jan 1980
Bob Smith	23 Mar 1981
Bob Jones	3 Dec 1986

```
SELECT name,dob from driver
where DOB = ‘3 Jan 1986’
```

NAME	DOB
Bob Jones	3 Dec 1986

- The tricky part with dates is remembering that dates get bigger as you move into the future.
- DATE1>DATE2 indicates DATE1 is in the future after DATE2.

(i,.e. 2007 > 2006 and Mar > Jan)

SELECT name,dob from driver

WHERE DOB >= ‘1 Jan 1981’

NAME	DOB
Bob Smith	23 Mar 1981
Bob Jones	3 Dec 1986

DATE Syntax

- Date must be in quotes
- Each DBMS handles dates in a slightly different way
- Dates like ‘1 Jan 2003’ work quite well.
- Oracle permits dates like ‘1-Jan-2003’
- Oracle also permits dates like ‘1-Jan-03’
  - If you type this it will assume 2003.
  - If you mean 1984 type 1984 not –84.
- You must always specify a day and a month. If you do not the DBMS will report an error.

BETWEEN (is inclusive)

- When dealing with dates sometimes you want to test to see if a field value falls *between* two dates.
- The easiest way to do this is with BETWEEN

- Find all drivers born between 1995 and 1999

    SELECT name,dob from driver

    WHERE DOB BETWEEN ‘1 Jan 1985’ AND ’31 Dec 1999’
- Between works for other things, not just dates…

    SELECT regno from CAR
    where price BETWEEN 5000 AND 10000;

NULL

- NULL indicates that something has no value
- It is not a value, and you cannot use normal comparison operators.
- For instance, looking for cars without owners…

Wrong: SELECT regno from car where owner = NULL

Wrong: SELECT regno from car where owner = ‘NULL’

- Instead there are two special operators,
  - IS NULL, or
  - IS NOT NULL

Edinburgh Napier UNIVERSITY

```
SELECT regno from car
WHERE OWNER is null
```

REGNO
SC04 BFE

Has no owner

```
SELECT regno from car
WHERE OWNER is not null
```

REGNO
F611 AAA
J111 BBB
A155 BDE
K555 GHT

LIKE

- Sometimes you want to have a rule involving partial strings, substrings, or *wildcards*
- LIKE does this, and is a slot-in replacement for ‘=‘
- If the string contains ‘%’ or ‘_’, LIKE uses them to support wildcards.

  % - Matches zero or more characters in the string

  _ - Matches exactly 1 character in the string

Examples

- Name LIKE ‘Jim Smith’ e.g. Jim Smith
- Name LIKE ‘_im Smith’ e.g. Tim Smith
- Name LIKE ‘_ _ _ Smith’ e.g. <u>Bob</u> Smith
- Name LIKE ‘% Smith’ e.g. Frank Smith
- Name LIKE ‘% S%’ e.g. Brian Smart
- Name LIKE ‘Bob %’ e.g. Bob Martin
- Name LIKE ‘%’ i.e. matches anyone

- LIKE is more expensive than =
- If you are not using wildcards, always use = rather than