Edinburgh Napier UNIVERSITY

SQL – JOINs and VIEWs

Chapter 3.3

V4.0

Edinburgh Napier UNIVERSITY

Multiple source tables

- Using more than a single table of a database is usually essential.
- The basic form is to list all the needed tables in the **FROM** line.
- You do have to explain to the DBMS how the tables should be joined together…

Edinburgh Napier UNIVERSITY

JOIN conditions

SELECT * from driver;

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

3 Rows

Edinburgh Napier UNIVERSITY

SELECT * from car;

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	

5 Rows

SELECT *
FROM car, driver

Edinburgh Napier UNIVERSITY

Cartesian (or Cross) Product = 15 Rows

REGNO	MAKE	COLOUR	PRICE	OWNER	NAME	DOB
F611 AAA	FORD	RED	12000	Jim Smith	Jim Smith	11 Jan 1980
J111 BBB	SKODA	BLUE	11000	Jim Smith	Jim Smith	11 Jan 1980
A155 BDE	MERCEDES	BLUE	22000	Bob Smith	Jim Smith	11 Jan 1980
K555 GHT	FIAT	GREEN	6000	Bob Jones	Jim Smith	11 Jan 1980
SC04 BFE	SMART	BLUE	13000		Jim Smith	11 Jan 1980
F611 AAA	FORD	RED	12000	Jim Smith	Bob Smith	23 Mar 1981
J111 BBB	SKODA	BLUE	11000	Jim Smith	Bob Smith	23 Mar 1981
A155 BDE	MERCEDES	BLUE	22000	Bob Smith	Bob Smith	23 Mar 1981
K555 GHT	FIAT	GREEN	6000	Bob Jones	Bob Smith	23 Mar 1981
SC04 BFE	SMART	BLUE	13000		Bob Smith	23 Mar 1981
F611 AAA	FORD	RED	12000	Jim Smith	Bob Jones	3 Dec 1986
J111 BBB	SKODA	BLUE	11000	Jim Smith	Bob Jones	3 Dec 1986
A155 BDE	MERCEDES	BLUE	22000	Bob Smith	Bob Jones	3 Dec 1986
K555 GHT	FIAT	GREEN	6000	Bob Jones	Bob Jones	3 Dec 1986
SC04 BFE	SMART	BLUE	13000		Bob Jones	3 Dec 1986

Edinburgh Napier UNIVERSITY

Relationship

Edinburgh Napier UNIVERSITY

FOREIGN KEY link

Car key

Owner key

REGNO	MAKE	COLOUR	PRICE	OWNER	NAME	DOB
F611 AAA	FORD	RED	12000	Jim Smith	Jim Smith	11 Jan 1980
J111 BBB	SKODA	BLUE	11000	Jim Smith	Jim Smith	11 Jan 1980
A155 BDE	MERCEDES	BLUE	22000	Bob Smith	Jim Smith	11 Jan 1980
K555 GHT	FIAT	GREEN	6000	Bob Jones	Jim Smith	11 Jan 1980
SC04 BFE	SMART	BLUE	13000		Jim Smith	11 Jan 1980
F611 AAA	FORD	RED	12000	Jim Smith	Bob Smith	23 Mar 1981
J111 BBB	SKODA	BLUE	11000	Jim Smith	Bob Smith	23 Mar 1981
A155 BDE	MERCEDES	BLUE	22000	Bob Smith	Bob Smith	23 Mar 1981
K555 GHT	FIAT	GREEN	6000	Bob Jones	Bob Smith	23 Mar 1981
SC04 BFE	SMART	BLUE	13000		Bob Smith	23 Mar 1981
F611 AAA	FORD	RED	12000	Jim Smith	Bob Jones	3 Dec 1986
J111 BBB	SKODA	BLUE	11000	Jim Smith	Bob Jones	3 Dec 1986
A155 BDE	MERCEDES	BLUE	22000	Bob Smith	Bob Jones	3 Dec 1986
K555 GHT	FIAT	GREEN	6000	Bob Jones	Bob Jones	3 Dec 1986
SC04 BFE	SMART	BLUE	13000		Bob Jones	3 Dec 1986

Traditional JOIN

```
SELECT *
FROM   car,driver
WHERE owner = name;
```

REGNO	MAKE	COLOUR	PRICE	OWNER	NAME	DOB
F611 AAA	FORD	RED	12000	Jim Smith	Jim Smith	11 Jan 1980
J111 BBB	SKODA	BLUE	11000	Jim Smith	Jim Smith	11 Jan 1980
A155 BDE	MERCEDES	BLUE	22000	Bob Smith	Bob Smith	23 Mar 1981
K555 GHT	FIAT	GREEN	6000	Bob Jones	Bob Jones	3 Dec 1986

Car Table

Driver Table

Edinburgh Napier UNIVERSITY

Modern JOIN !

```
SELECT *
FROM   car JOIN driver ON ( owner = name )
;
```

REGNO	MAKE	COLOUR	PRICE	OWNER	NAME	DOB
F611 AAA	FORD	RED	12000	Jim Smith	Jim Smith	11 Jan 1980
J111 BBB	SKODA	BLUE	11000	Jim Smith	Jim Smith	11 Jan 1980
A155 BDE	MERCEDES	BLUE	22000	Bob Smith	Bob Smith	23 Mar 1981
K555 GHT	FIAT	GREEN	6000	Bob Jones	Bob Jones	3 Dec 1986

OUTER JOIN

- Consider the last line of the *unconstrained* join…

REGNO	MAKE	COLOUR	PRICE	OWNER	NAME	DOB
SC04 BFE	SMART	BLUE	13000		Bob Jones	3 Dec 1986

- This is a car without an owner.
- Sometimes we want to see the rows that fail the join condition due to NULL values.
- To see NULL joins we use an OUTER JOIN.
- The JOIN discussed up to this point is known as an INNER JOIN (i.e. a "normal" join).
- We will discuss this more in the relational algebra section.

Edinburgh Napier UNIVERSITY

- Really, **outer join** means we want to force all the rows in one of the tables to appear in the result.
- There are some variants to OUTER JOIN, depending on which rows you want to keep.

- Consider this:

FROM car JOIN driver on (driver = name)

To the RIGHT of the JOIN

To the LEFT of the JOIN

- If you want all the rows in CAR to always be in the answer, you need a LEFT OUTER JOIN
- If you want all the rows in DRIVER to always be in the answer, you need a RIGHT OUTER JOIN

- In our example the NULL causing the problem is in OWNER, which is in the table CAR.
- CAR is to the left of the word JOIN
- We need a LEFT OUTER JOIN (which is written as just LEFT JOIN).

```
SELECT *
FROM   car LEFT JOIN driver ON ( owner = name )
;
```

Edinburgh Napier UNIVERSITY

```
SELECT *
FROM   car LEFT JOIN driver ON ( owner = name )
;
```

REGNO	MAKE	COLOUR	PRICE	OWNER	NAME	DOB
F611 AAA	FORD	RED	12000	Jim Smith	Jim Smith	11 Jan 1980
J111 BBB	SKODA	BLUE	11000	Jim Smith	Jim Smith	11 Jan 1980
A155 BDE	MERCEDES	BLUE	22000	Bob Smith	Bob Smith	23 Mar 1981
K555 GHT	FIAT	GREEN	6000	Bob Jones	Bob Jones	3 Dec 1986
SC04 BFE	SMART	BLUE	13000			

Car Table

Driver Table

FULL OUTER JOIN

- With a LEFT OUTER join, you keep all the rows from the left.
- With a RIGHT OUTER join, you keep all the rows from the right.
- What if you want to keep all the rows from both sides?
  - You need FULL OUTER join, written as FULL JOIN.
- Consider a new row, David Davis, added to DRIVER. David owns no cars.

NAME	DOB
Jim Smith	11 Jan 1980
Bob Smith	23 Mar 1981
Bob Jones	3 Dec 1986
David Davis	1 Oct 1975

Example: LEFT

```
SELECT *
FROM   car LEFT JOIN driver ON ( owner = name )
;
```

REGNO	MAKE	COLOUR	PRICE	OWNER	NAME	DOB
F611 AAA	FORD	RED	12000	Jim Smith	Jim Smith	11 Jan 1980
J111 BBB	SKODA	BLUE	11000	Jim Smith	Jim Smith	11 Jan 1980
A155 BDE	MERCEDES	BLUE	22000	Bob Smith	Bob Smith	23 Mar 1981
K555 GHT	FIAT	GREEN	6000	Bob Jones	Bob Jones	3 Dec 1986
SC04 BFE	SMART	BLUE	13000			

Car Table

Driver Table

Example: Right

```
SELECT *
FROM   car RIGHT JOIN driver ON ( owner = name )
;
```

REGNO	MAKE	COLOUR	PRICE	OWNER	NAME	DOB
F611 AAA	FORD	RED	12000	Jim Smith	Jim Smith	11 Jan 1980
J111 BBB	SKODA	BLUE	11000	Jim Smith	Jim Smith	11 Jan 1980
A155 BDE	MERCEDES	BLUE	22000	Bob Smith	Bob Smith	23 Mar 1981
K555 GHT	FIAT	GREEN	6000	Bob Jones	Bob Jones	3 Dec 1986
					David Davis	1 Oct 1975

Example: Full

```
SELECT *
FROM   car FULL JOIN driver ON ( owner = name )
;
```

REGNO	MAKE	COLOUR	PRICE	OWNER	NAME	DOB
F611 AAA	FORD	RED	12000	Jim Smith	Jim Smith	11 Jan 1980
J111 BBB	SKODA	BLUE	11000	Jim Smith	Jim Smith	11 Jan 1980
A155 BDE	MERCEDES	BLUE	22000	Bob Smith	Bob Smith	23 Mar 1981
K555 GHT	FIAT	GREEN	6000	Bob Jones	Bob Jones	3 Dec 1986
SC04 BFE	SMART	BLUE	13000			
					David Davis	1 Oct 1975

Naming and the Dot Notation

- Sometimes column names can be ambiguous.
- Take two tables, called ALPHA and BRAVO. Lets assume each table has a column called NAME.

SELECT name from ALPHA, BRAVO;

This query fails, since NAME is in both tables the DBMS cannot work out which NAME you want.

- You can tell the DBMS by putting the table name in front of the column name, separated by a dot.

SELECT alpha.name from ALPHA,

Edinburgh Napier UNIVERSITY

Aliases

- If you are writing a big query, you can find yourself typing the same long table names again and again.
- This can be quite prone to errors.
- SQL allows us to rename tables for the **duration of a query**.
- You put the new name immediately after the table name in FROM, separated by a space.
- Rather than:

```
SELECT car.owner FROM car;
```

- You can say

```
SELECT c.owner FROM car c;
```

- This also works for JOIN.

```
SELECT c.regno, c.owner, d.dob
FROM car c JOIN driver d on (c.owner =
   d.name)
```

- It makes it much easier for the user to work out what values are coming from which table.

Edinburgh Napier UNIVERSITY

Selfjoin

- A selfjoin, or an *equijoin*, is where the same table is used twice in a FROM line.
- It indicates a need to use a single table in two different ways simultaneously.
- Consider the question:

"Who drives a car the same colour as Bob Smith"?

SELECT colour FROM car WHERE owner = ‘Bob Smith’;

Colour
BLUE

SELECT owner FROM car
WHERE colour = ‘BLUE’
AND owner != ‘Bob Smith’
AND owner not null;

owner
Jim Smith

Edinburgh Napier UNIVERSITY

Solution – a self join

```
SELECT other.owner
FROM car bobsmith, car other
WHERE bobsmith.colour = other.colour  -- 1
AND bobsmith.owner = 'Bob Smith'
        -- 2
AND bobsmith.owner != other.owner
        -- 3!
AND other.owner NOT NULL                -- 4
;
```

Edinburgh Napier UNIVERSITY

VIEWS

- VIEWS are to a database what subroutines are to a programming language (i.e., hidden code executed when called).
- A view allows us to store a query in the database, so that we can access it later, by name.
- We treat views in the same way as a normal table when writing queries.

CREATE VIEW *ViewName* (attribute names)
AS *select attribute(s) from table(s);*

- Lets write a query to tell us:

How many drivers and how many cars are in the database?

- We could write them separately…

```
SELECT count(*) from DRIVER;
SELECT count(*) from CAR;
```

- However this is two queries, and we want to do it in one query…
- Lets store the two queries in two different VIEWs.

optional

CREATE VIEW count1 (total) **AS** *select count(*) from driver;*

CREATE VIEW count2 (total) **AS** *select count(*) from car;*

Select * from count1;

Total
3

Select * from count2;

Total
5

Total	Total
3	5

Combined:

Select count1.total,count2.total

from count1, count2;

CREATE VIEW count1 (total1) **AS** *select count(*) from driver;*

CREATE VIEW count2 (total2) **AS** *select count(*) from car;*

Select * from count1;

Total1
3

Select * from count2;

Total2
5

Combined:

Select total1, total2

from count1,count2;

Total1	Total2
3	5

Edinburgh Napier UNIVERSITY

Deleting a VIEW

- When you are finished with a VIEW you have created you can delete it.
- You do with with the **DROP VIEW** command.
- For our example…

```
DROP VIEW count1;

DROP VIEW count2;
```