# SQL - Subqueries and Schema

Chapter 3.4

V3.0

Dr Gordon Russell

INVESTOR IN PEOPLE

NAPIER UNIVERSITY

EDINBURGH

# Subqueries

- Subquery – one select statement inside another
- Used in the WHERE clause
- Subqueries can return many rows.
- Subqueries can only return 1 column.
- Used as a replacement for view or selfjoin.
- Some programmers see them as easier to understand than other options.
- The main drawback is that they can be much slower than selfjoin or view.

# Simple Example

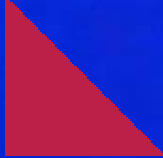- Who in the database is older than Jim Smith?

SELECT dob FROM driver WHERE name = 'Jim Smith';

| Dob |
| --- |
| 11 Jan 1980 |

SELECT name FROM driver WHERE dob > '11 Jan 1980';

| name |
| --- |
| Bob Smith |
| Bob Jones |

NAPIER UNIVERSITY

EDINBURGH

INVESTOR IN PEOPLE

- Combined together:

```
SELECT name
FROM driver
WHERE dob > (SELECT dob
                FROM driver
                WHERE name = 'Jim Smith')
;
```

- This query will only work if there is only 1 Jim Smith.

NAPIER UNIVERSITY
EDINBURGH

# ANY and ALL

- To support subqueries which return more than 1 row we need some additional operators… ANY and ALL.
- ANY – changes the rule so that it must be true for at least one of the rows returned from the subquery.
- ALL – changes the rule so that it must be true for each and every row returned from the subquery.

- The ANY or ALL operator goes immediately before the open bracket of the subquery.

NAPIER UNIVERSITY
EDINBURGH

# Example 1

- What cars are the same solour as a car owned by Jim Smith?
- Jim owns 2 cars, one is RED and the other BLUE. We are looking for cars which are either RED or BLUE.

SELECT regno FROM car
WHERE colour = ANY ( SELECT colour
                        FROM car
                        WHERE owner = 'Jim Smith'
            )

# Example 2

- List the drivers younger than all the people who own a blue car.
- We are looking for the age of drivers who own a BLUE car, and listing drivers who are younger than all of those ages.

```
SELECT name,dob FROM driver
WHERE dob < ALL (
                SELECT dob
                FROM car JOIN driver ON (owner=name)
                WHERE colour = 'BLUE'
        ) ;
```

# IN and NOT IN

- We earlier saw IN working for sets like ('A','B').
- A subquery itself returns its result as a set.
- Therefore we can use IN and NOT IN on subqueries.

- Question: Which cars are the same colour as one of Jim Smith's cars?

```
SELECT regno FROM car
WHERE colour IN (SELECT colour FROM car
                 WHERE owner = 'Jim Smith')
;
```

NAPIER UNIVERSITY
EDINBURGH

# Example of NOT IN

- Question: Which cars DO NOT have the same colour as one of Jim Smith's cars?

SELECT regno FROM car

WHERE colour NOT IN (SELECT colour FROM car

WHERE owner = 'Jim Smith')

;

NAPIER UNIVERSITY

EDINBURGH

# EXISTS

- If a question involves discovering uniqueness, then it can probably be easily solved using the operator EXISTS or NOT EXISTS.
- The EXISTS operator tests the result of running a subquery, and if any rows are returned it is TRUE, else it is FALSE.
- NOT EXISTS does the opposite of EXISTS.

- Note that subqueries can actually refer to tables defined outside the brackets which define the subquery. This is exceptionally useful, but can be slow to execute and confusing to look at.

NAPIER UNIVERSITY
EDINBURGH

INVESTOR IN PEOPLE

- Question: List the colours which are only used once in the database.

```
SELECT colour
FROM car a
WHERE exists (
    SELECT colour              -- The row does not matter
    FROM car b                 -- unique name from a
    WHERE a.colour = b.colour  -- Same colour as a
    AND a.regno != b.regno     -- Different car from a
);
```

NAPIER UNIVERSITY
EDINBURGH

# UNION

- Sometimes you might write two or more queries which produce the same types of answer, and you want to combine the rows of these answers into a single result.
- UNION does this, and basically allows you to join different SELECT statement together.
- UNION automatically removes duplicate rows.

- For the next example, assume a row has been added to the DRIVER table for David Davis, but that he owns no cars.
- Question: List all drivers in the DRIVER table, together with how many cars they own.

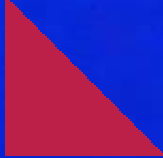SELECT name,count(*)

FROM driver JOIN car on (name = owner)

| NAME | Count(*) |
|-----------|----------|
| Jim Smith | 2 |
| Bob Smith | 1 |
| Bob Jones | 1 |

David Davis is missing, as he did not satisfy the JOIN condition.

- Write a query just for David Davis…

SELECT name,0

FROM driver

WHERE name NOT IN (select owner from car)

| NAME | |
|------|---|
| David Davis | 0 |

NAPIER UNIVERSITY

EDINBURGH

- Link the two queries together:

SELECT name,count(*)

FROM driver JOIN car on (name = owner)

UNION

SELECT name,0

FROM driver

WHERE name

  NOT IN (select owner

       from car)

| NAME | Count(*) |
|------|----------|
| Jim Smith | 2 |
| Bob Smith | 1 |
| Bob Jones | 1 |
| David Davis | 0 |

NAPIER UNIVERSITY

EDINBURGH

# Changing Data

- We have looked so far at just SELECT
- There are some other useful operators too:
  - INSERT
  - DELETE
  - UPDATE

NAPIER UNIVERSITY
EDINBURGH

# INSERT

INSERT INTO table_name
      [(column_list)] VALUES (value_list)

The column_list can be omitted if every column is to be assigned a value, otherwise it must list columns to be assigned values. The value_list is a set of literal values giving the value of each column in the same order as the column_list, if specified, or as the columns are defined in the original CREATE TABLE.

    insert into driver
        values ('Jessie James','31 Nov 1892');
    insert into driver (name,dob)
        values ('John Johnstone','1 Aug 1996');

NAPIER UNIVERSITY
EDINBURGH

# DELETE

DELETE FROM table_name [WHERE condition];

the rows of table_name which satify the condition are deleted.

- Delete Examples:
  DELETE FROM car; -- Delete all rows from CAR
  ;

  DELETE FROM car
  WHERE owner is null          -- Delete rows for cars without
  owners;

NAPIER UNIVERSITY
EDINBURGH

# UPDATE

UPDATE table_name
    SET column_name =
    expression,{column_name=expression}
    [WHERE condition]

Set all BLUE cars to GREEN
    UPDATE car SET colour = 'GREEN'
    WHERE colour = 'BLUE'

Add VAT/Purchase Tax at 17.5% to all prices
    UPDATE car SET price = price * 1.175

NAPIER UNIVERSITY
EDINBURGH

# View Manipulation

When is a view 'materialised' or populated with rows of data?

• When it is defined or
• when it is accessed

If it is the former then subsequent inserts, deletes and updates would not be visible. If the latter then changes will be seen.

Some systems allow you to chose when views are materialised, most do not and views are materialised whenever they are accessed thus all changes can be seen.
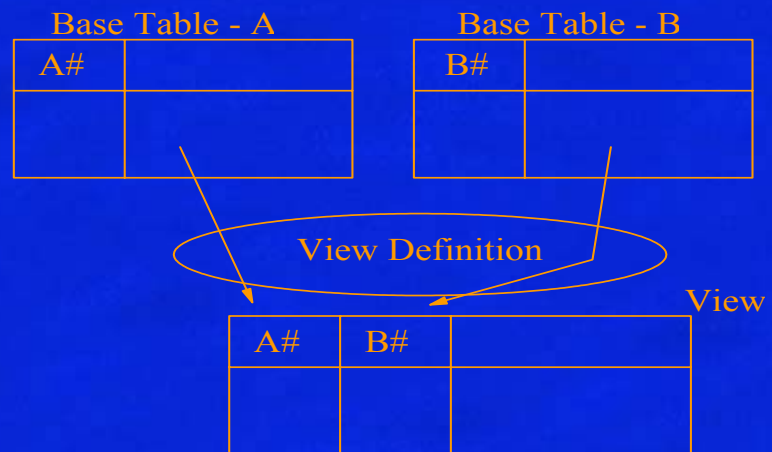
NAPIER UNIVERSITY
EDINBURGH

# VIEW update, insert and delete

Can we change views?
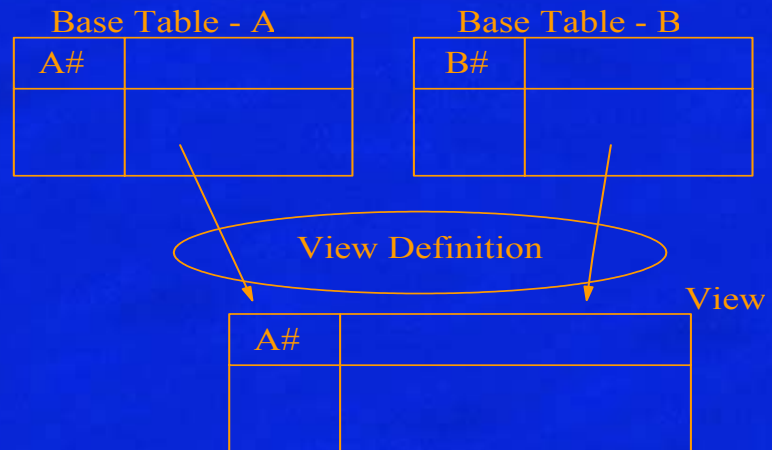
*   Yes, provided the primary key of all the base tables which make up the view are present in the view.

Base Table - A

| A# | |
|----|--|
| | |

Base Table - B

| B# | |
|----|--|
| | |

View Definition

View

| A# | B# | |
|----|----|--|
| | | |

NAPIER UNIVERSITY

EDINBURGH

# VIEW cont...

- This view cannot be changed because we have no means of knowing which row of B to modify

Base Table - A

| A# | |
|----|---|
| | |

Base Table - B

| B# | |
|----|---|
| | |

View Definition

View

| A# | |
|----|---|
| | |

NAPIER UNIVERSITY
EDINBURGH

INVESTOR IN PEOPLE

# Controlling Schema

- All the commands so far have allowed data to be looked at, changes, added to, or removed.
- We also need commands to build, change, and remove table definitions.
- We call this schema changes.
- The useful commands to do this include:
    - CREATE TABLE
    - DROP TABLE
    - ALTER TABLE

# CREATE TABLE

- Column types are needed to tell the DBMS what is allowed to be stored in each attribute column.
- A selection of types include:
  - INTEGER
  - REAL
  - DECIMAL : Including DECIMAL(5) and DECIMAL(4,2)
  - VARCHAR
  - CHAR : Pads out strings with spaces
  - DATE

# SYNTAX

CREATE TABLE tablename (
    colname type      optionalinfo
    ,colname type     optionalinfo
    ,other optional info
);

- Optionalinfo could be things like
  - a INTEGER REFERENCES b(c)
  - PRIMARY KEY
  - NOT NULL

- The other optional info at the end of the definition tend to be rules which impact on more than one attribute:
    - PRIMARY KEY (col1,col2,…)
    - FOREIGN KEY (col1,col2,…) REFERENCES othertable

NAPIER UNIVERSITY

EDINBURGH

# CAR + DRIVER

```
CREATE table driver (
    name         varchar(30)      PRIMARY KEY
    ,dob          date            NOT NULL
);
CREATE TABLE car (
    regno        varchar(8)       PRIMARY KEY
    ,make        varchar(20)
    ,colour      varchar(30)
    ,price       decimal(8,2)
    ,owner       varchar(30)      references driver(name)
);
```

NAPIER UNIVERSITY
EDINBURGH

# CAR + DRIVER
# Using Additional Info

```
CREATE table driver (
    name                    varchar(30)
    ,dob        date            NOT NULL
    ,PRIMARY KEY (name)
);
```

# CAR + DRIVER
# Using Additional Info

```
CREATE TABLE car (
    regno          varchar(8)
    ,make                    varchar(20)
    ,colour        varchar(30)
    ,price                   decimal(8,2)
    ,owner         varchar(30)
    ,FOREIGN KEY(owner)        references driver(name)
    ,PRIMARY KEY(regno)
);
```

NAPIER UNIVERSITY
EDINBURGH

# DROP TABLE

- If you want to delete a table you use DROP TABLE.
  - DROP TABLE tablename
- The main difficulty with dropping a table is referential integrity. As CAR refers to DRIVER, you must delete CAR first then DRIVER. If you try to delete DRIVER first, the system would report an error.

DROP TABLE car;
DROP TABLE driver;

NAPIER UNIVERSITY
EDINBURGH

# MODIFY TABLE

- To change a table which already exists you could use MODIFY TABLE.

- It is a complex command with many different options.

- A simple example of it would be adding an address field to the DRIVER table.

ALTER TABLE driver ADD address varchar(50)

# SELECT - Order of Evaluation

| | |
|---|---|
| SELECT [DISTINCT] column_name | **5,6** eliminate unwanted data |
| FROM      label_list | **1** Cartesian Product |
| [WHERE  condition ] | **2** eliminate unwanted rows |
| [GROUP BY column_list | **3** group rows |
| [HAVING condition ]] | **4** eliminate unwanted groups |
| [ORDER BY column_list[DESC]] | **7** sort rows |

The last four components are optional.

NAPIER UNIVERSITY

EDINBURGH