Edinburgh Napier UNIVERSITY

SQL - Subqueries and Schema

Chapter 3.4

V4.0

Subqueries

Edinburgh Napier UNIVERSITY

- Subquery – one SELECT statement inside another
- Used in the WHERE clause
- Subqueries can return many rows.
- Subqueries can only return 1 column i.e. SELECT X
- 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?

1. SELECT dob FROM driver WHERE name = ‘Jim Smith’;

Dob
11 Jan 1980

2. SELECT name FROM driver WHERE dob < ’11 Jan 1980’;

name
Bob Smith
Bob Jones

- Combined together:

=, >, >=, <, <= only work if there is just one record returned by the subquery

```
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.

ANY and ALL

Edinburgh Napier UNIVERSITY

- 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 row 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.

Example 1

- What cars are the same colour 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 that are 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

Edinburgh Napier UNIVERSITY

- 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? [Subquery returns (‘RED’, ‘BLUE’)]

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

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')
;
```

EXISTS

Edinburgh Napier UNIVERSITY

- 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 contain the subquery. This is exceptionally useful, but can be slow to execute and confusing to look at.

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

```
SELECT DISTINCT a.colour
FROM car a, car b                    -- unique name for each table
WHERE a.colour = b.colour            -- Same colour
AND a.regno != b.regno -- Different cars
```

colour
BLUE

Edinburgh Napier UNIVERSITY

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

```
SELECT a.colour
FROM car a
WHERE NOT EXISTS (

    SELECT b.colour
    FROM car b                          -- unique name for table
    WHERE a.colour = b.colour       -- Same colour as table a
    AND a.regno != b.regno -- Different car from table a
                        );
```

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 combine different SELECT statements together.
- UNION automatically removes duplicate rows.
- UNION-ed columns must match and be of the same type

- For the next example, assume a new 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

Edinburgh Napier UNIVERSITY

```
SELECT name, count(*)
FROM driver JOIN car on (name = owner) GROUP BY
    name
```

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

- Link (UNION) 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

Changing Data

Edinburgh Napier UNIVERSITY

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

INSERT

Edinburgh Napier UNIVERSITY

```
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 the columns to be assigned values.

The value_list is a set of values for 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');
```

DELETE

Edinburgh Napier UNIVERSITY

DELETE FROM table_name [WHERE condition];

the rows of table_name which satisfy the condition are deleted.

- Delete Examples:

  DELETE FROM car; -- Deletes **all** rows from CAR

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

UPDATE

Edinburgh Napier UNIVERSITY

```
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
```

View Manipulation

Edinburgh Napier UNIVERSITY

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

- When it is defined or
- when it is accessed

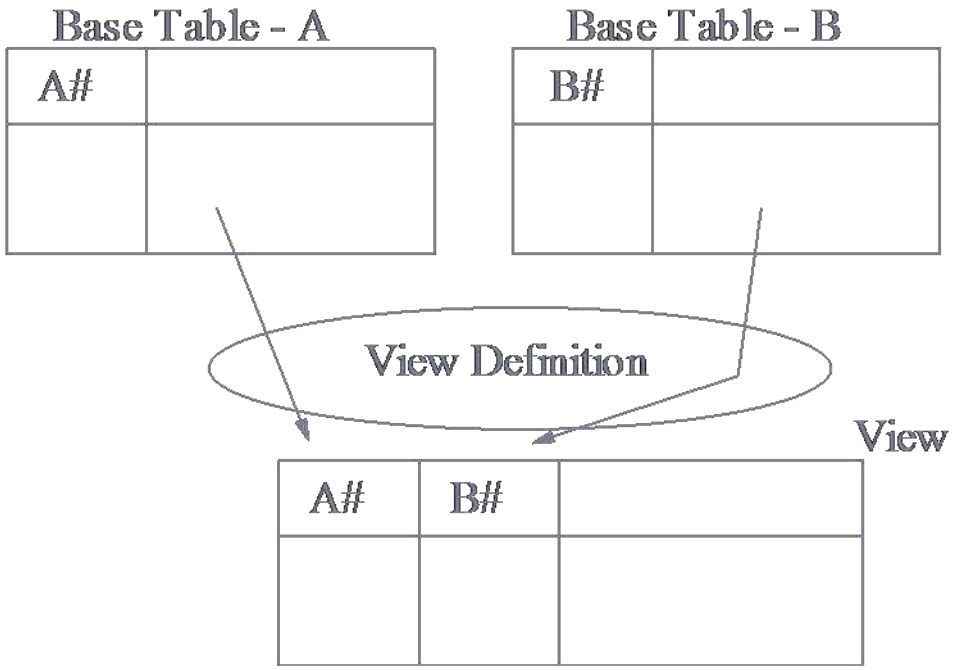
If it is the former then subsequent inserts, deletes and updates would not be visible. If the latter then changes will be seen.

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

VIEW update, insert and delete

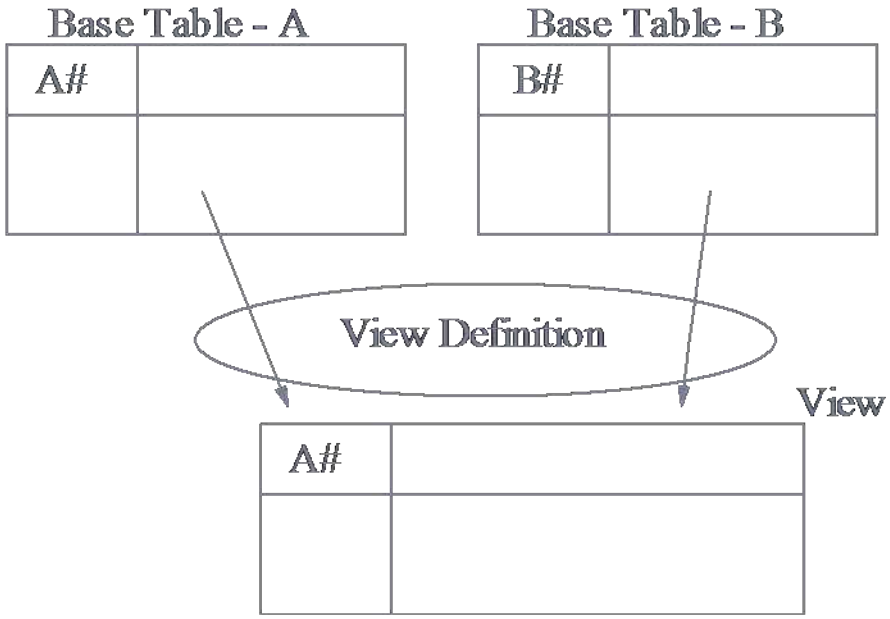
Can we change data in views?

- Yes, provided the **primary keys** of all the base tables which make up the view **are present** in the view.



VIEW cont...

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



Controlling the Schema

Edinburgh Napier UNIVERSITY

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

CREATE TABLE

Edinburgh Napier UNIVERSITY

- 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 – variable character length
  - CHAR -- Pads out strings with spaces
  - DATE
  - BLOB –maps, picture, videos etc

SYNTAX

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

- Optionalinfo could be things like
  - Col_B INTEGER REFERENCES TableB (Col_C)
  - Col_A INTEGER PRIMARY KEY
  - Col_A INTEGER NOT NULL

- The other optional info at the end of the definition (i.e. “other optional Info”) tend to be rules which impact on more than one attribute:
  - PRIMARY KEY (col1, col2,…)
  - FOREIGN KEY (col1, col2,…)
    REFERENCES sometable (col3)

CAR + DRIVER

Edinburgh Napier UNIVERSITY

```
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)
);
```

Entity Integrity

Referential Integrity

Foreign key

DRIVER
Using Additional Info

Edinburgh Napier UNIVERSITY

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

PRIMARY KEY (name)
);
```

CAR
Using Additional Info

Edinburgh Napier UNIVERSITY

```
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)
);
```

DROP TABLE

Edinburgh Napier UNIVERSITY

- 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 (owner references name), 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;
```

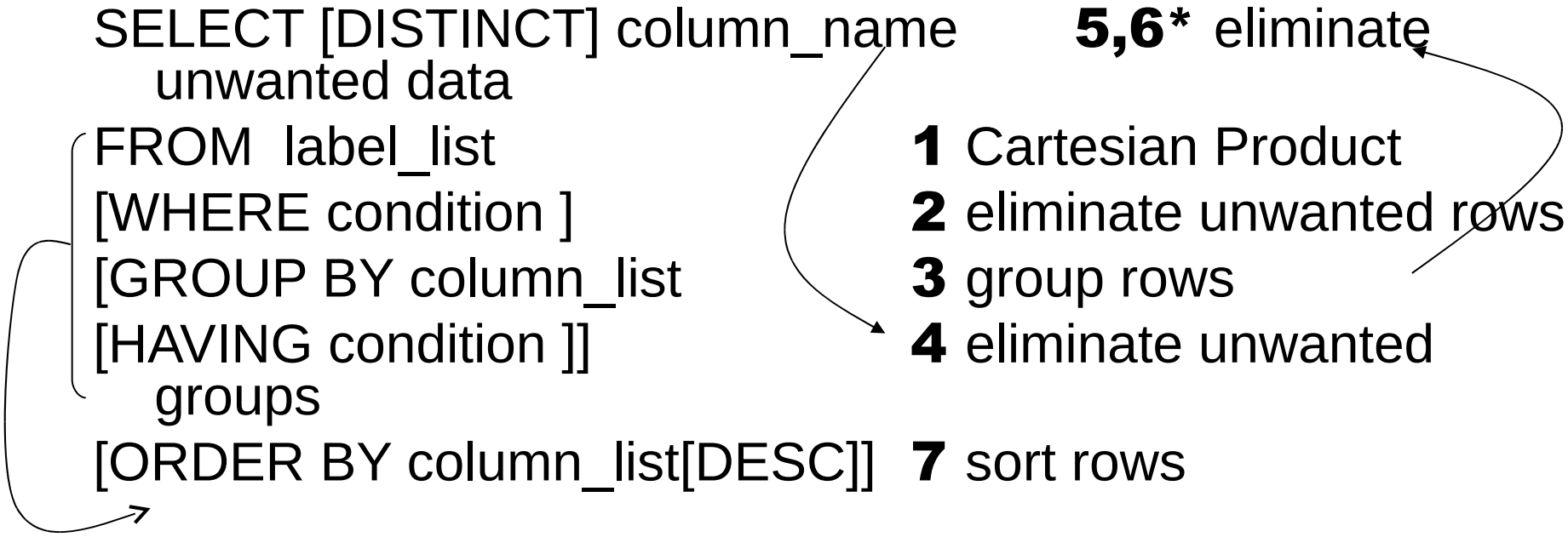
ALTER TABLE (DDL)

Edinburgh Napier UNIVERSITY

- To change a table which already exists you could use ALTER TABLE.
- It is a complex command with many different options [MODIFY, ADD, DROP, RENAME].
- 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



The last four components are optional.

* **5** eliminate duplicates **6** specifies columns