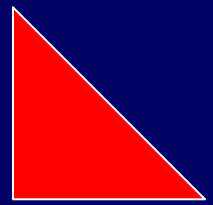
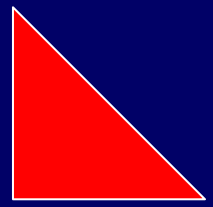


SQL 3

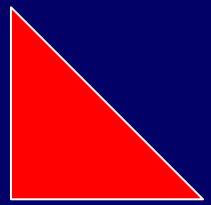
Unit 1.4

Subqueries



- A SELECT query can be used within another SELECT condition and is then known as a subquery
- A subquery can return only one attribute having zero or more values
- The use of a view may provide a simpler query format than using techniques such as self-joins

Simple Example

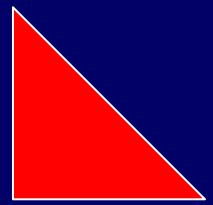


Name employees younger than Liza Brunell:

```
SELECT surname,forenames
FROM employee
WHERE dob >
   ( SELECT dob FROM employee – subquery
     WHERE forenames = 'Liza'
        AND surname = 'Brunell');
```

Note - there is no need to use aliases for employee table references since these are unambiguous within the scopes of the main query and the subquery.

Subqueries with ANY, ALL

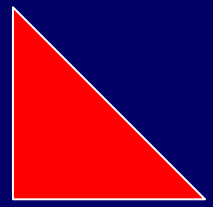


- ANY or ALL can be used to qualify tests carried out on the values in the set returned by a subquery.

List employees currently earning less than anyone now in programming:

```
SELECT      empno FROM jobhistory
WHERE       salary < ALL
            (SELECT    salary – subquery
             WHERE     position LIKE '%Programmer%'
                AND    enddate IS NULL)
  AND       enddate IS NULL;
```

Subqueries with IN, NOT IN

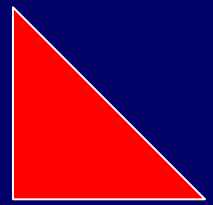


- IN and NOT IN can be used to test if a value is or is not present in the set of values returned by a subquery

List the names and employee numbers of all those who have never been on atraining course:

```
SELECT      empno,forenames,surname
FROM        employee
WHERE       empno NOT IN
            (SELECT DISTINCT empno
            FROM empcourse);
```

Subqueries with EXISTS



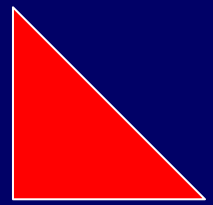
- EXISTS tests if a set returned by a subquery is empty

List the employee number and job title of all those doing a unique job:

```
SELECT      empno
FROM        jobhistory mainjh
WHERE       enddate IS NULL
    AND NOT EXISTS (
                SELECT empno
                FROM   jobhistory subjh
                WHERE enddate IS NULL
                AND    mainjh.position = subjh.position
                AND    mainjh.empno = subjh.empno );
```

Note that aliases are needed to enable references from subquery to main query

UNION of Subqueries

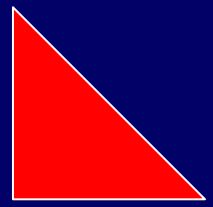


- A query included two or more subqueries connected by a set operation such as UNION (MINUS or INTERSECT).
- UNION returns all the distinct rows returned by two subqueries

List the number of each employee in departments 2 or 4, plus employees who know about administration:

```
(SELECT     empno FROM employee
 WHERE      depno IN (2,4))
 UNION
 (SELECT    empno FROM course,empcourse
 WHERE      course.courseno = empcourse.courseno
 AND        cname LIKE '%Administration%');
```

Views

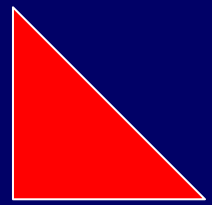


- A view is a way of collecting information from parts of one or more tables to enable users to more easily access the database.

CREATE VIEW view_name [(column_list)]
AS query;

Attributes can be renamed in column_list if required.

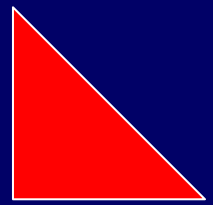
Views cont...



- Suppose a user needs to regularly manipulate details about employee, name, and current position. It might be simpler to create a view limited to this information only, rather than always extracting it from two tables:

```
CREATE VIEW empjob AS
   SELECT    employee.empno,surname,forenames,position
   FROM      employee,jobhistory
   WHERE     employee.empno = jobhistory.empno
   AND       enddate IS NULL;
```

Views cont...



- A view can be accessed like any other table

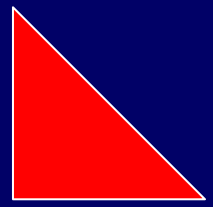
List those currently in Programming type jobs:

```
SELECT      empno,surname,forenames
FROM        empjob
WHERE       position LIKE '%Program%';
```

- A view can (should) be dropped when no longer required:

```
DROP VIEW view_name
```

Views cont...



- The use of a view may provide a simpler query format than using techniques such as self-joins or subqueries

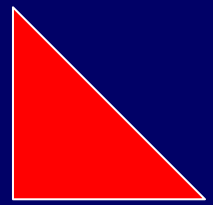
Name employees younger than Liza Brunell:

```
CREATE VIEW liza AS
        SELECT dob FROM employee
        WHERE forenames = 'Liza'
        AND surname = 'Brunell';

SELECT surname,forenames
FROM employee,liza
WHERE employee.dob > liza.dob;

DROP VIEW liza;
```

View Manipulation



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

- When it is defined or
- when it is accessed

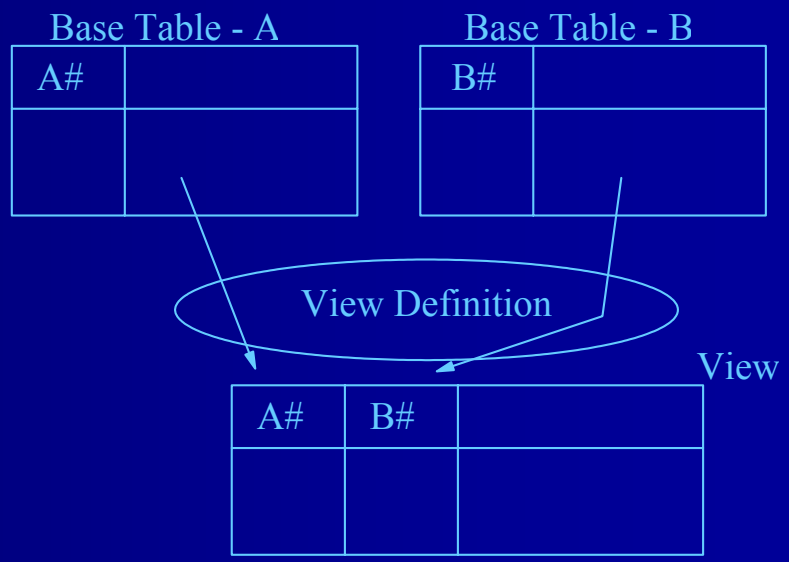
If it is the former then subsequent inserts, deletes and updates would not be visible. If the latter then changes will be seen.

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.

VIEW update, insert and delete

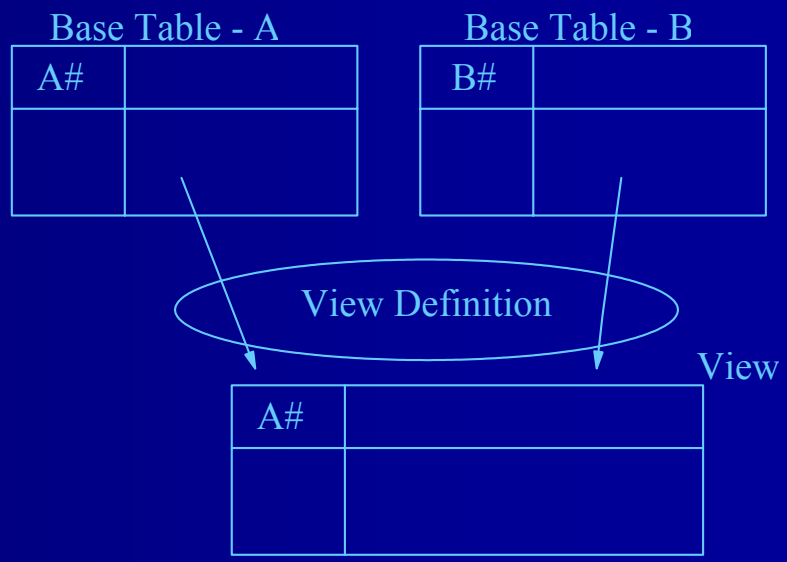
Can we change views?

- Yes, provided the primary key of all the base tables which make up the vieware present in the view.

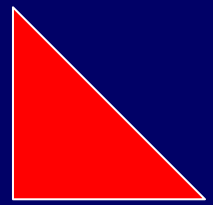


VIEW cont...

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

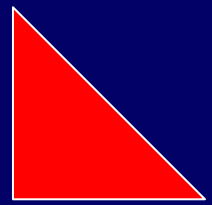


Other SQL Statements



- So far we have just looked at SELECT but we need to be able to do other operations as follows:
  - INSERT - which writes new rows into a database
  - DELETE - which deletes rows from a database
  - UPDATE - which changes values in existing rows
- We also need to be able to control access to out tables by other users (see the later SECURITY lecture).
- We may need to provide special views of tables to make queries easier to write. These views can also be made available to other users so that they can easily see our data but not change it in any way.

INSERT

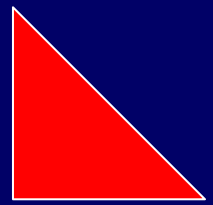


INSERT INTO table_name
[(column_list)] VALUES (value_list)

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 course
        values (11,'Advanced Accounting',10-jan-2000);
insert into course (courseno,cname)
        values(13,'Advanced Administration');
```

DELETE



DELETE FROM table_name [WHERE condition];

the rows of table_name which satify the condition are deleted.

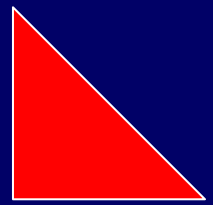
- Delete Examples:

  DELETE FROM jobhistory -- remote current posts from jobhistoryWHERE enddate IS NULL;

  DELETE FROM jobhistory -- Remove all posts from jobhistory,
  ; -- leaving an empty table

  DROP jobhistory; -- Remove jobhistory table completely

UPDATE



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

The expression can be

- NULL
- a literal value
- an expression based upon the current column value

Give a salary rise of 10% to all accountants:

```
UPDATE jobhistory
    SET salary = salary * 1.10
    WHERE position LIKE '%Accountant%'
    AND enddate IS NULL;
```