Views in SQL

SQL-based systems as described heretofore suffer from two limitations:

(i) Security cannot be provided at sub-table level. It is not possible, for example, to hide certain columns and/or certain rows from individual users.

(ii) Sometimes, an apparently straightforward data request requires a complex SQL query. This is because the structure of the database may be at variance with the nature of the data request.

Views are SQL artefacts which aim to partially address these problems. A view - also called a *derived table* is a definable *window* into one or more tables of the database. The term *derived table* relates to the fact that data is not stored on disk; rather, it can be materialized when required. Specifically, a view can address the two limitations mentioned above:

(i) It is possible to grant permissions on a view, but not on the base tables from which it derives.

(ii) Views can be constructed which apparently structure the database in a manner suitable to one or more users of the system, thereby simplifying their access to data.

1. View Definition

e.g. Create a view containing information on Houston-based employees of Department 5

```
CREATE VIEW HoustonEmployeesD5  AS
SELECT  Ssn, Fname, Lname, Sex, Salary
FROM  EMPLOYEE
WHERE  Address LIKE ‘%Houston, TX’
    AND  Dno = 5
```

e.g. Create a view containing certain information on Houston-based employees of the Research department

```
CREATE  VIEW  HER_1  ( StaffId, Name, Sex, Salary )  AS
SELECT  Ssn, CONCAT (Lname, ‘, ‘, Fname), Sex, Salary
FROM  EMPLOYEE, DEPARTMENT
WHERE  Dno = DNumber
    AND  Address LIKE ‘%Houston, TX’
    AND  Dname = ‘Research’
```

or

```
CREATE  VIEW  HER_2  ( StaffId, Name, Sex, Salary )  AS
SELECT  Ssn, CONCAT (Lname, ‘, ‘, Fname), Sex, Salary
FROM  EMPLOYEE
WHERE  Address LIKE ‘%Houston, TX’
   AND  Dno IN
        (  SELECT  Dnumber
           FROM   DEPARTMENT
             AND   Dname = ‘Research’  )
```

e.g. Create a view that links departments and projects, quantified by work undertaken

```
CREATE  VIEW  DeptProjWork ( DeptName, ProjName, WorkDone )  AS
SELECT  Dname,  Pname, SUM (Hours)
FROM  WORKS_ON, DEPARTMENT, PROJECT
WHERE  Pno = Pnumber
    AND  Dnum = Dnumber
```

e.g. Create a view containing information on work carried out by Houston-based employees of the Research Department

```
CREATE  VIEW  HERWORK  AS
SELECT  Pno, Pname, Plocation, Hours
FROM  WORKS_ON, PROJECT
WHERE  Pno  =  Pnumber
   AND  Essn IN
        (  SELECT  StaffId
           FROM  HER_1  )
```

or

```
CREATE  VIEW  HERWORK  AS
SELECT  Pno, Pname, Plocation, Hours
FROM  WORKS_ON, PROJECT, HER_1
WHERE  Pno  =  Pnumber
   AND  Essn =  StaffId
```

e.g. Create a view which gives access to department payrolls

```
CREATE  VIEW  DeptPayroll  ( Dno, Outlay )  AS
SELECT  Dno, SUM (Salary)
FROM  EMPLOYEE
GROUP  BY  Dno
```

e.g. Create a view which gives average monthly salary values across departments

```
CREATE  VIEW  DeptAvgSal ( DeptId, AvgSal )  AS
SELECT  Dno, AVG (Salary/12)
FROM  EMPLOYEE
GROUP  BY  Dno
```

2. View Manipulation

For the purpose of writing queries, views can be treated in exactly the same way as base tables, as depicted in the examples below. In fact, it is possible that a user constructing a query is unaware of the fact that a table reference is to a view rather than a base table.

e.g. ***Find the name and salary of Houston-based employees of department 5***

```
SELECT  Fname, Lname, Salary
FROM  HoustonEmployeesD5
```

e.g. ***Find the number of females from Houston working for department 5***

```
SELECT  COUNT (*)
FROM  HoustonEmployeesD5
WHERE  Sex  =  'F'
```

e.g. ***Find the name of projects on which more than 1000 hours of work has been completed***

```
SELECT  ProjName
FROM  DeptProjWork
WHERE  WorkDone > 1000
```

e.g. ***Find the identifier and name of Houston-based members of the Research department, together with details of work he/she has carried out on the "Computerization" project***

```
SELECT  StaffId, Name, Pno, Hours
FROM  Her_1,  WORKS_ON, PROJECT
WHERE  StaffId  =  Sno
    AND  Pno = Pnumber
    AND  Pname = ‘Computerization’
```

[Could also use HERWORK view for this data request]

e.g. ***Find the salary outlay for department 5***

```
SELECT  AvgSal
FROM  DeptAvgSal
WHERE  DeptId = 5
```

3. View Implementation

Views are implemented in SQL using a technique known as *query modification*: the text of the view creation statement and the text of the user query are merged into a single *final query* that refers only to base tables [and is the query that the user would have issued if no views existed]. The *query modifier* algorithm uses pre-defined rules for text merging, as outlined in the examples below.

e.g. View Definition:

```
CREATE  VIEW  MaleStaff  AS

SELECT  Ssn, Fname, Lname, Bdate, Salary
FROM  EMPLOYEE
WHERE  Sex  =  'M'
```

e.g. User Query:

```
SELECT  Fname, Lname
FROM  MaleStaff
WHERE  Salary > 50000
```

Final Query:

```
SELECT  Fname, Lname
FROM  EMPLOYEE
WHERE  Sex  =  'M'
       and  Salary > 50000
```

e.g. User Query:

```
SELECT  Fname, Lname, Salary
FROM  MaleStaff
WHERE  Bdate <= '1990-01-01'
     AND  Salary between  12000  AND  15000
ORDER  BY  Lname, Fname
```

Final Query:

```
SELECT  Sname, Lname, Salary
FROM  EMPLOYEE
WHERE  Sex  =  'M'
        and  Bdate < '1990-01-01'
       AND  Salary between  12000  AND  15000
ORDER  BY  Lname, Fname
```

e.g. ***User Query:***

```
SELECT  *
FROM  MaleStaff
WHERE  Lname  =  'White'
```

Final Query:

```
SELECT  Sno, Fname, Lname, Bdate, Salary
FROM  Staff
WHERE  Sex  =  'M'
       and   Lname  =  'White'
```

e.g. ***User Query:***

```
SELECT  Ssn,  Fname,  Lname,  Pno,  Hours
FROM  MaleStaff,  WORKS_ON
WHERE  Ssn  =  Essn
     AND  Lname = ‘Smith’
     AND  Hours  >  20.0
```

Final Query:

```
SELECT  Ssn,  Fname,  Lname,  Pno,  Hours
FROM  Staff , WORKS_ON
WHERE  Sex  =  'M'
      and  Ssn  =  Essn
     AND  Lname = ‘Smith’
     AND  Hours > 20.0
```

e.g. ***View Definition:***

```
CREATE  VIEW  HER_1  ( StaffId, Name, Sex, Salary )  AS

SELECT  Ssn, CONCAT (Lname, ‘, ‘, Fname), Sex, Salary
FROM  EMPLOYEE, DEPARTMENT
WHERE  Dno = DNumber
     AND  Address LIKE ‘%Houston, TX’
     AND  Dname = ‘Research’
```

e.g. ***User Query:***

```
SELECT  StaffId,  Name, Salary
FROM  HER_1
WHERE  Sex = ‘M’
     AND  Salary <= 25000
```

Final Query:

```
SELECT  Ssn,  CONCAT (Lname, ‘, ‘, Fname), Salary
FROM  EMPLOYEE, DEPARTMENT
WHERE  Dno = DNumber
     AND  Address LIKE ‘%Houston, TX’
     AND  Dname = ‘Research’
      and    Sex = ‘M’
     AND  Salary <= 25000
```

e.g. View Definition:

```
CREATE  VIEW  HERWORK  AS
SELECT  Pno, Pname, Plocation, Hours
FROM  WORKS_ON, PROJECT, HER_1
WHERE  Pno  =  Pnumber
    AND  Essn =  StaffId
```

```
CREATE  VIEW  HER_1  ( StaffId, Name, Sex, Salary )  AS
SELECT  Ssn, CONCAT (Lname, ‘, ‘, Fname), Sex, Salary
FROM  EMPLOYEE, DEPARTMENT
WHERE  Dno = DNumber
     AND  Address LIKE ‘%Houston, TX’
     AND  Dname = ‘Research’
```

e.g. User Query:

```
SELECT  Pno, Pname
FROM  HERWORK
WHERE Plocation = ‘Sugarland
   AND  Hours > 100
```

Final Query:

(i)

```
SELECT  Pno, Pname
FROM  WORKS_ON, PROJECT, HER_1
WHERE  Pno  =  Pnumber
     AND  Essn =  StaffId
      and   Plocation = ‘Sugarland
     AND  Hours > 100
```

(ii)

```
SELECT  Pno, Pname
FROM  EMPLOYEE, DEPARTMENT, WORKS_ON, PROJECT
WHERE  Dno = DNumber
     AND  Address LIKE ‘%Houston, TX’
     AND  Dname = ‘Research’
      and    Pno  =  Pnumber
     AND  Essn =  Ssn
      and   Plocation = ‘Sugarland
     AND  Hours > 100
```

4. View Updatability

For retrieval purposes, there is no real difference (to the user) between derived and base tables: in fact, the user may not even know what type of table he/she is retrieving from. For non-retrieval commands (INSERT, DELETE & UPDATE), however, the situation may be different.

It is conceivable that the query modification technique could be extended to commands other than SELECT. For example, the following conversion could take place:

```
UPDATE HoustonEmployeesD5              UPDATE  EMPLOYEE
SET Salary = Salary * 2           ➔    SET Salary = Salary * 2
WHERE  Salary < 10000                  WHERE Address  LIKE  '%Houston, TX'
                                           AND  Dno = 5
                                           AND  Salary < 10000
```

We could show examples to illustrate conversion of various UPDATE & DELETE statements on views into equivalent statements on the base tables (INSERT is a different case).

However, the following INSERT cannot be legitimately converted to an insert on the underlying base table [One reason is that NOT NULL columns of the base table do not appear in the view. Anotheris that, while the view contains details on Houston-based employees, it does not contain the actual address; so the insert is illogical]:

```
INSERT HoustonEmployeeD5
VALUES ('SL27', 'Joan', 'Crawford', 'F', 15000)
```

Some DBMS take the simplistic viewpoint that views can be used only for retrieval, and that no commands other than SELECT can be processed against them. Others take a more liberal approach, assuming that views are updatable under restricted circumstances. Typically, these would be:

1. View must be derived from a single base table
2. View must contain all of the NOT NULL columns of that base table
3. View cannot contain any function calls in its SELECT clause

Note that a simple syntactic analysis of a view definition would detect whether or not these conditions are met – and that the view could be marked as 'not updatable' upon creation.