Handout 9
Views and other SQL features.

Views

View - a pseudo table where data is derived from one or more base tables.

- Do not actually store data. Just display it.
- Formed with a defining query and given a name. The resulting view-table can be used like any other table for querying.
- can also be used for insertion/update, but there are some subtleties involved.

Syntax:

1. CREATE VIEW <view name> [(colHeading1, ..., colHeadingN)] AS <query> - creates a new view.

2. CREATE OR REPLACE VIEW <view name> [(colHeading1, ..., colHeadingN)] AS <query> - replaces the view with the <view name> if it exists, with the new one.

Examples:

1. Create a phone directory showing each employee's name, department and phone number.

   CREATE OR REPLACE VIEW PhoneDirectory (Department, Employee, Phone) AS
   SELECT dept_name, RTRIM(emp_lname||','||emp_fname), emp_phone
   FROM htopi.dept, htopi.employee;

   View used in a single-table query: Display the directory ordered alphabetically by the department name, then employee name - uses the PhoneDirectory view.

   SELECT *
   FROM PhoneDirectory
   ORDER BY Department, Employee;

   View used in a multi-table query Display the part of the phone directory that lists only employees working in building 'H'.

   SELECT Department, Employee, Phone
   FROM PhoneDirectory, htopi.dept
   WHERE Department = dept_name and dept_building = 'H'
   ORDER BY Department, Employee;
Purposes of Views:

- A view can greatly simplify user perception of a complex database.
- A view can restrict access to data.
- A view can be used as a table storing intermediate results of some computation:

**Example:** Display the names and number of dependents of all employees over 25 years of age who have more than average number of dependents in their department.

Solution 1 - uses a correlated subquery:

```sql
SELECT e1.emp_lname, e1.emp_dept,
MONTHS_BETWEEN(SysDate, e1.emp_dateofbirth)/12 as Age,
e1.emp_nbr_of_dependents
FROM htopi.employee_s e1
WHERE MONTHS_BETWEEN(SysDate, e1.emp_dateofbirth)/12 < 25 and
  e1.emp_nbr_of_dependents > (SELECT AVG(e2.emp_nbr_of_dependents)
  FROM htopi.employee e2
  WHERE e2.emp_dept = e1.emp_dept);
```

Note, we cannot display the average number of dependents in the same table.

Solution 1 - uses a view to store intermediate results: list of all departments with the average number of dependents of employees in the department.

CREATE OR REPLACE VIEW AvgDependents (DepartmentNumber, AvgNumDependents) AS

```sql
SELECT emp_dept, avg(emp_nbr_of_dependents)
FROM htopi.employee_s
GROUP BY emp_dept;
```

```sql
SELECT e1.emp_lname, e1.emp_dept,
MONTHS_BETWEEN(SysDate, e1.emp_dateofbirth)/12 as Age,
e1.emp_nbr_of_dependents, AvgNumDependents
FROM htopi.employee_s e1, AvgDependents
WHERE MONTHS_BETWEEN(SysDate, e1.emp_dateofbirth)/12 < 25 and
  DepartmentNumber = e1.emp_dept and
  e1.emp_nbr_of_dependents > AvgNumDependents;
```

Storing intermediate results in a view simplifies the query by eliminating the correlated subquery. Also, the resulting table includes a row that displays the average number of dependents in the department.

**Transaction support**

**Transaction** - discrete unit of work that must be completely processed or not processed at all. May involve multiple updates If any update fails, then all other updates must be cancelled.

Example: money transfer from one account to another.

SQL commands for transactions:

- BEGIN TRANSACTION/END TRANSACTION - Marks boundaries of a transaction. At BEGIN - all updates ar logged. A END - they are permanently committed to the Database.
• COMMIT - Makes all updates permanent
• ROLLBACK - Cancels updates since the last COMMIT

Routines and Triggers

Both refer to procedural code that gets executed on a database. Both are part of DBMS code (not outside applications).

Routines - program modules that execute *on demand*. May combine a sequence of computations/updates to a database.

Example: Find the set of the least popular products and reduce their price by 10%.

Triggers - routines that execute *automatically in response to a database event* (*INSERT*, *UPDATE*, or *DELETE*)

Example: before inserting a new order for the customer, check that their debt does not exceed a certain amount.