Data Modeling and Databases II - The Relational Data Model and SQL

Luiz Araujo

Innopolis University

Week 3 - Tutorial

In this tutorial

- More Complex SQL Retrieval Queries
- Specifying Constraints as Assertions and Actions as Triggers
- Views (Virtual Tables) in SQL
- Schema Change Statements in SQL

More Complex SQL Retrieval Queries

- Comparisons Involving NULL and Three-Valued Logic:

SQL uses a three-valued logic with values TRUE, FALSE, and UNKNOWN instead of the standard two-valued (Boolean) logic with values TRUE or FALSE

In select-project-join queries, only those combinations of tuples that evaluate the logical expression in the WHERE clause of the query to TRUE are selected

```
Q18:    SELECT    Fname, Lname
        FROM      EMPLOYEE
        WHERE     Super_ssn IS NULL;
```

Table 7.1 Logical Connectives in Three-Valued Logic

(a)	AND	TRUE	FALSE	UNKNOWN
	TRUE	TRUE	FALSE	UNKNOWN
	FALSE	FALSE	FALSE	FALSE
	UNKNOWN	UNKNOWN	FALSE	UNKNOWN
(b)	OR	TRUE	FALSE	UNKNOWN
	TRUE	TRUE	TRUE	TRUE
	FALSE	TRUE	FALSE	UNKNOWN
	UNKNOWN	TRUE	UNKNOWN	UNKNOWN
(c)	NOT			
	TRUE	FALSE		
	FALSE	TRUE		
	UNKNOWN	UNKNOWN		

More Complex SQL Retrieval Queries

- Nested Queries, Tuples, and Set/Multiset Comparisons:

```
Q4A:  SELECT   DISTINCT Pnumber
      FROM     PROJECT
      WHERE    Pnumber IN
               ( SELECT   Pnumber
                 FROM     PROJECT, DEPARTMENT, EMPLOYEE
                 WHERE    Dnum = Dnumber AND
                          Mgr_ssn = Ssn AND Lname = 'Smith' )
               OR
               Pnumber IN
               ( SELECT   Pno
                 FROM     WORKS_ON, EMPLOYEE
                 WHERE    Essn = Ssn AND Lname = 'Smith' );
```

```
SELECT   DISTINCT Essn
FROM     WORKS_ON
WHERE    (Pno, Hours) IN   ( SELECT   Pno, Hours
                             FROM     WORKS_ON
                             WHERE    Essn = '123456789' );
```

```
SELECT   Lname, Fname
FROM     EMPLOYEE
WHERE    Salary > ALL   ( SELECT   Salary
                          FROM     EMPLOYEE
                          WHERE    Dno = 5 );
```

Notice that this query can also be specified using the MAX aggregate function

Do it with your colleague by your side!

More Complex SQL Retrieval Queries

- How to kill the DBMS performance? You can use correlated subqueries, i.e. when the inner query is filtered by values from the outer query)

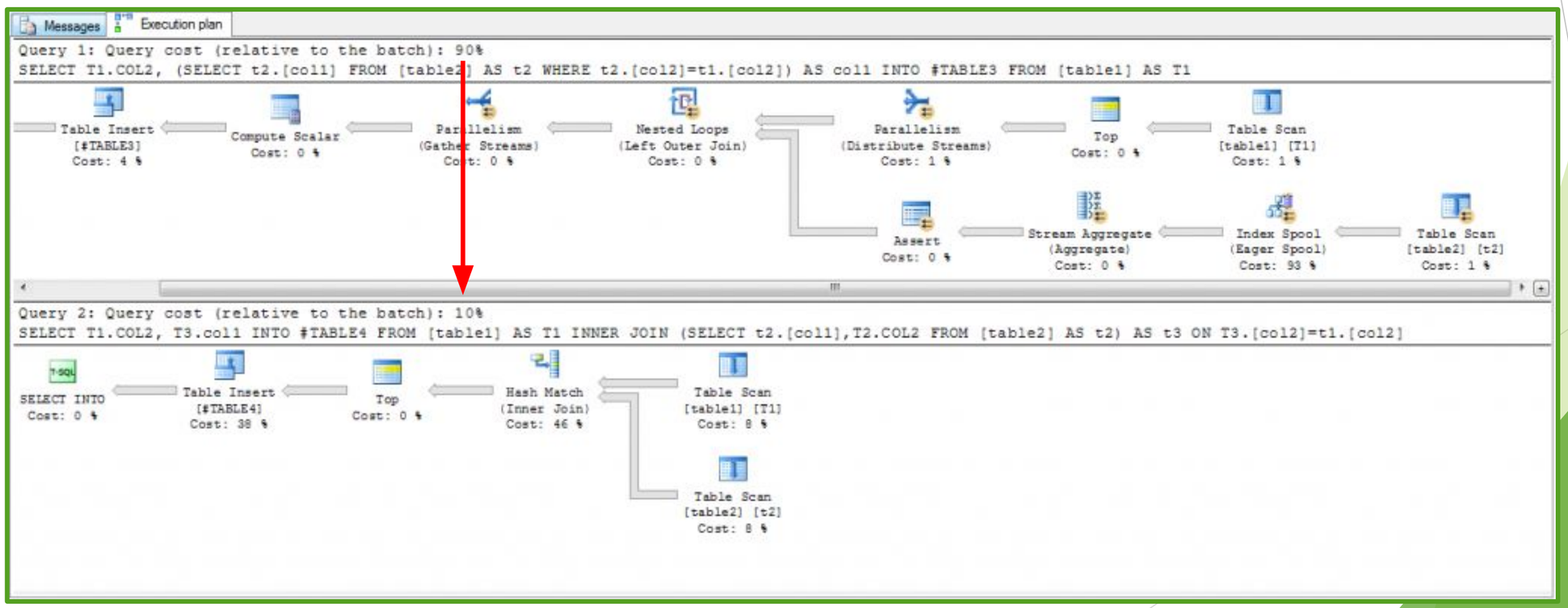
```
SELECT   T1.COL2 ,
    col1 = (SELECT    t2.[col1]
            FROM [table2] AS t2
            WHERE   t2.[col2] = t1.[col2]
            )  --The inner query that is correlated with the outer query
INTO #TABLE3
FROM [table1] AS T1
```

The nested query is evaluated once for each tuple (or combination of tuples) in the outer query! $O(n^2)$

More Complex SQL Retrieval Queries

- How to kill the DBMS performance? You can use correlated subqueries, i.e. when the inner query is filtered by values from the outer query)

Nested query vs Inner Join



When you have a nested loop you are going to execute the bottom branch for every record in the top branch.
With the hash match join you get a result from both branches and match them together.

More Complex SQL Retrieval Queries

The EXISTS and UNIQUE Functions in SQL

- EXISTS and UNIQUE are Boolean functions that return TRUE or FALSE; hence,
- they can be used in a WHERE clause condition.

```
Q16B:   SELECT   E.Fname, E.Lname
        FROM     EMPLOYEE AS E
        WHERE    EXISTS ( SELECT   *
                          FROM     DEPENDENT AS D
                          WHERE    E.Ssn = D.Essn AND E.Sex = D.Sex
                                   AND E.Fname = D.Dependent_name);
```

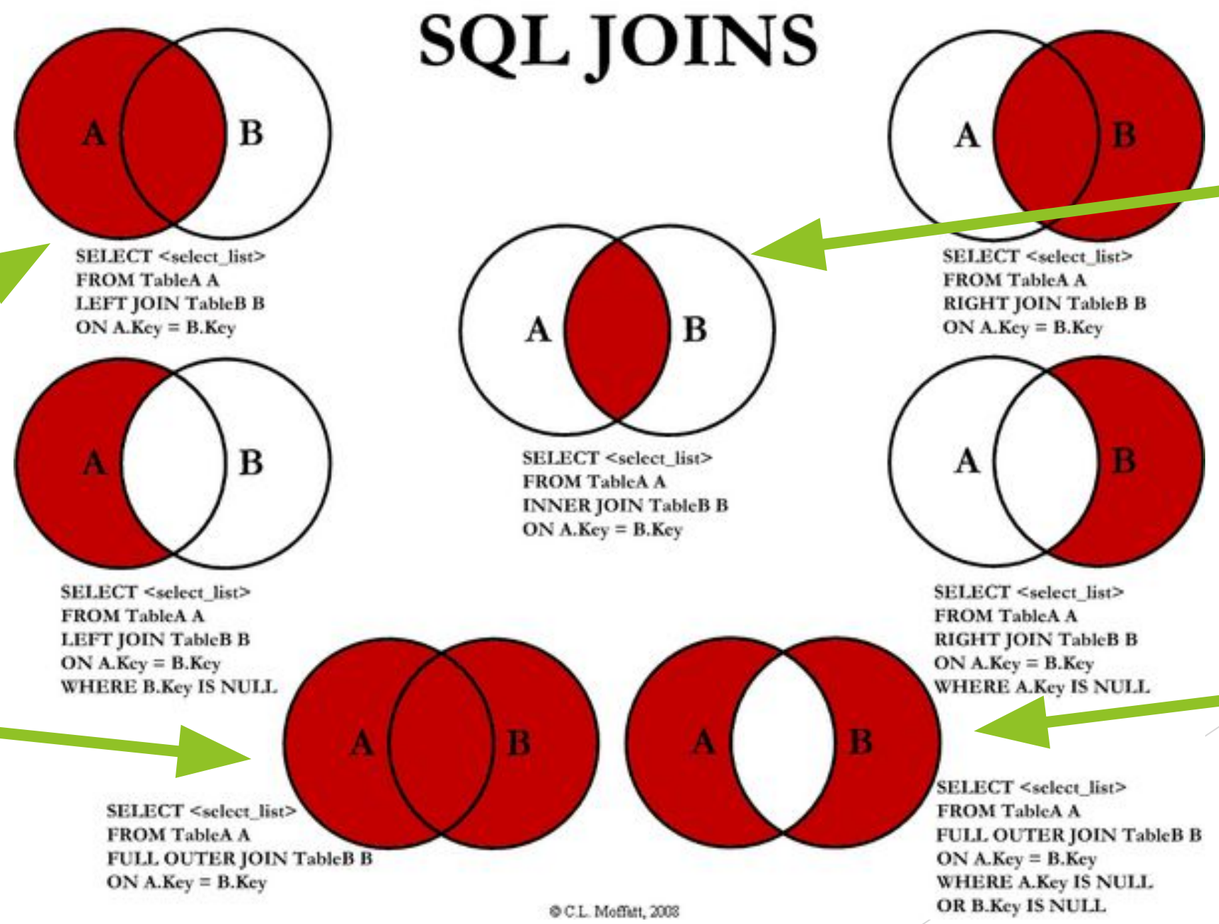
There is another SQL function, UNIQUE(Q), which returns TRUE if there are no duplicate tuples in the result of query Q;
otherwise, it returns FALSE.
This can be used to test whether the result of a nested query is a set (no duplicates) or a multiset (duplicates exist).

More Complex SQL Retrieval Queries

NATURAL JOINS, INNER JOIN, OUTER JOIN, LEFT {OUTER} JOIN, RIGHT {OUTER} JOIN

This query will return all of the records in the left table (table A) regardless if any of those records have a match in the right table (table B). It will also return any matching records from the right table.

This query will return all of the records from both tables, joining records from the left table (table A) that match records from the right table (table B).



This query will return all of the records in the left table (table A) that have a matching record in the right table (table B).

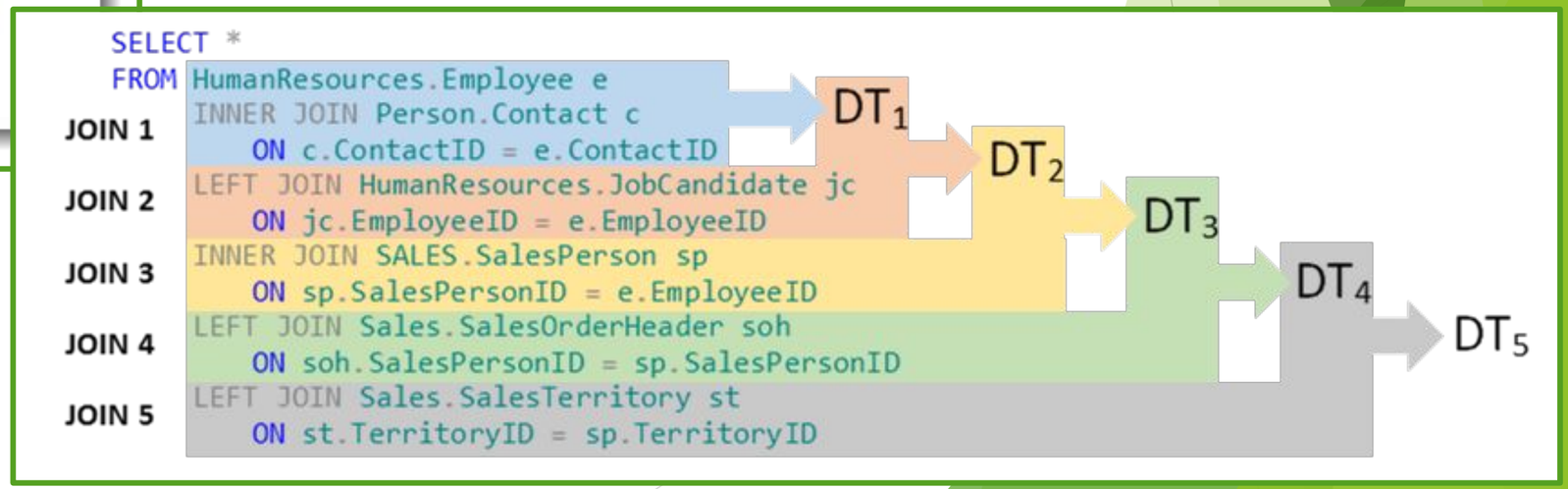
This query will return all of the records in the left table (table A) and all of the records in the right table (table B) that do not match.

More Complex SQL Retrieval Queries

- What aliens on what planet wrote such a thing?

```
SELECT *
FROM HumanResources.Employee e
INNER JOIN Person.Contact c
      ON c.ContactID = e.ContactID
LEFT JOIN HumanResources.JobCandidate jc
      ON jc.EmployeeID = e.EmployeeID
INNER JOIN SALES.SalesPerson sp
      ON sp.SalesPersonID = e.EmployeeID
LEFT JOIN Sales.SalesOrderHeader soh
      ON soh.SalesPersonID = sp.SalesPersonID
LEFT JOIN Sales.SalesTerritory st
      ON st.TerritoryID = sp.TerritoryID
```

It performs a series of incremental, single joins between two tables at a time (while this article refers only to tables for simplicity sake, joins can be between tables, views, table valued functions, CTEs, and derived table subqueries). Each single join produces a single derived table (DT) that is then joined to the next table and so on.



More Complex SQL Retrieval Queries

- Aggregate Functions

```
Q20:  SELECT  SUM (Salary), MAX (Salary), MIN (Salary), AVG (Salary)
      FROM    (EMPLOYEE JOIN DEPARTMENT ON Dno = Dnumber)
      WHERE   Dname = 'Research';
```

```
Q21:  SELECT  COUNT (*)
      FROM    EMPLOYEE;
```

- The GROUP BY and HAVING Clauses

```
Q26:  SELECT  Pnumber, Pname, COUNT (*)
      FROM    PROJECT, WORKS_ON
      WHERE   Pnumber = Pno
      GROUP BY Pnumber, Pname
      HAVING  COUNT (*) > 2;
```

What is this query selecting?

```
Q28:  SELECT  Dno, COUNT (*)
      FROM    EMPLOYEE
      WHERE   Salary>40000 AND Dno IN
              ( SELECT  Dno
                FROM    EMPLOYEE
      GROUP BY Dno
                HAVING  COUNT (*) > 5)
      GROUP BY Dno;
```

More Complex SQL Retrieval Queries

- Aggregate Functions

```
Q20:  SELECT  SUM (Salary), MAX (Salary), MIN (Salary), AVG (Salary)
      FROM    (EMPLOYEE JOIN DEPARTMENT ON Dno = Dnumber)
      WHERE   Dname = 'Research';
```

```
Q21:  SELECT  COUNT (*)
      FROM    EMPLOYEE;
```

- The GROUP BY and HAVING Clauses

```
Q26:  SELECT  Pnumber, Pname, COUNT (*)
      FROM    PROJECT, WORKS_ON
      WHERE   Pnumber = Pno
      GROUP BY  Pnumber, Pname
      HAVING  COUNT (*) > 2;
```

For each department that has more than five employees, retrieve the department number and the number of its employees who are making more than $40,000.

```
Q28:  SELECT  Dno, COUNT (*)
      FROM    EMPLOYEE
      WHERE   Salary>40000 AND Dno IN
              ( SELECT  Dno
                FROM    EMPLOYEE
      GROUP BY  Dno
                HAVING  COUNT (*) > 5)
      GROUP BY  Dno;
```

More Complex SQL Retrieval Queries

- Challenge: Write a query that displays the rank associated with each row without using the RANK function provided by some DBMSs
- EXAMPLE:

Table *Total_Sales*

Name	Sales
John	10
Jennifer	15
Stella	20
Sophia	40
Greg	50
Jeff	20

Result:

Name	Sales	Sales_Rank
Greg	50	1
Sophia	40	2
Stella	20	3
Jeff	20	3
Jennifer	15	5
John	10	6

Who will find the correct SQL statement first? You or the colleague by your side?

More Complex SQL Retrieval Queries

- Challenge: Write a query that displays the rank associated with each row without using the RANK function provided by some DBMSs
- EXAMPLE:

Table *Total_Sales*

Name	Sales
John	10
Jennifer	15
Stella	20
Sophia	40
Greg	50
Jeff	20

Result:

Name	Sales	Sales_Rank
Greg	50	1
Sophia	40	2
Stella	20	3
Jeff	20	3
Jennifer	15	5
John	10	6

Solution

```
SELECT
   a1.Name, a1.Sales, COUNT (a2.Sales) Sales_Rank

FROM Total_Sales a1, Total_Sales a2

WHERE
   a1.Sales < a2.Sales OR
   (a1.Sales=a2.Sales AND a1.Name = a2.Name)

GROUP BY a1.Name, a1.Sales
ORDER BY a1.Sales DESC, a1.Name DESC;
```

More Complex SQL Retrieval Queries

Recursive Queries

- An example of a recursive relationship between tuples of the same type is the relationship between an employee and a supervisor
- EXAMPLE: Retrieve all supervisees of a supervisory employee e at all levels—that is, all employees e′ directly supervised by e, all employees e′ directly supervised by each employee e′, all employees e‴ directly supervised by each employee e″, and so on:

```
Q29:    WITH RECURSIVE    SUP_EMP (SupSsn, EmpSsn) AS
            ( SELECT      SupervisorSsn, Ssn
              FROM        EMPLOYEE
              UNION
              SELECT      E.Ssn, S.SupSsn
              FROM        EMPLOYEE AS E, SUP_EMP AS S
              WHERE       E.SupervisorSsn = S.EmpSsn)
        SELECT*
        FROM              SUP_EMP;
```

View SUP_EMP that will hold the result of the recursive query. Initially empty.

Loading the first level (base query)

Successive level of supervisees, where the view contents are joined again with the base values to get the second level combinations, which are UNIONed with the first level. This is repeated with successive levels until no more tuples are added to the view.

In this tutorial

- ~~More Complex SQL Retrieval Queries~~
- Specifying Constraints as Assertions and Actions as Triggers
- Views (Virtual Tables) in SQL
- Schema Change Statements in SQL

Specifying Constraints as Assertions and Actions as Triggers

Specifying General Constraints as Assertions in SQL

- Additional types of constraints that are outside the scope of the built-in relational model constraints (primary and unique keys, entity integrity, and referential integrity)
- EXAMPLE: “Constraint that the salary of an employee must not be greater than the salary of the manager of the department that the employee works”

Condition (in parentheses) that must hold true on every database state
In other words, this is a query that selects any tuples that violate the desired condition.

```
CREATE ASSERTION SALARY_CONSTRAINT
CHECK ( NOT EXISTS ( SELECT     *
                     FROM       EMPLOYEE E, EMPLOYEE M,
                                DEPARTMENT D
                     WHERE      E.Salary>M.Salary
                                AND     E.Dno = D.Dnumber
                                AND     D.Mgr_ssn = M.Ssn ) );
```

Specifying Constraints as Assertions and Actions as Triggers

Specifying General Constraints as Assertions in SQL

- EXAMPLE II: “Boston based departments do not employ trainers”

```
create assertion NO_TRAINERS_IN_BOSTON as CHECK
(not exists
        (select 'trainer in Boston’
        from EMP e, DEPT d
        where e.DEPTNO = d.DEPTNO
           and e.JOB    = 'TRAINER’
           and d.LOC    = 'BOSTON’)
)
```

Specifying Constraints as Assertions and Actions as Triggers

Specifying General Constraints as Assertions in SQL

- CHECK applies to a single row
- ASSERTION stops action being taken on a database object. It is a predicate expressing a condition we wish the database to always satisfy
- Only when a transaction changes involved data in such a manner that it could potentially violate the SQL assertion, would the RDBMS perform a re-validation
- If the assertion is valid, any further modification to the database is allowed only if it does not cause that assertion to be violated

Specifying Constraints as Assertions and Actions as Triggers

Introduction to Triggers

- Used to specify automatic actions that the database system will perform when certain events and conditions occur
- This type of functionality is generally referred to as active databases
- EXAMPLE: We want to check whenever an employee's salary is greater than the salary of his or her direct supervisor
- Events that can trigger this rule: inserting a new employee record, changing an employee's salary, or changing an employee's supervisor

Specifying Constraints as Assertions and Actions as Triggers

Introduction to Triggers

- EXAMPLE: We want to check whenever an employee's salary is greater than the salary of his or her direct supervisor

```
R5: CREATE TRIGGER SALARY_VIOLATION
    BEFORE INSERT OR UPDATE OF SALARY, SUPERVISOR_SSN
      ON EMPLOYEE
    FOR EACH ROW
      WHEN ( NEW.SALARY > ( SELECT SALARY FROM EMPLOYEE
                            WHERE SSN = NEW.SUPERVISOR_SSN ) )
                            INFORM_SUPERVISOR(NEW.Supervisor_ssn,
                            NEW.Ssn );
```

Event (BEFORE/AFTER type OF columns ON table)

Condition (WHEN…)

Action (sequence of SQL statements). In this example, execute the store procedure INFORM_SUPERVISOR

- Used for maintaining database consistency, monitoring database updates, and updating derived data automatically

In this tutorial

- ~~More Complex SQL Retrieval Queries~~
- ~~Specifying Constraints as Assertions and Actions as Triggers~~
- Views (Virtual Tables)
- Schema Change Statements

Views (Virtual Tables)

- A view is supposed to be always up-to-date; if we modify the tuples in the base tables on which the view is defined, the view must automatically reflect these changes

```
V1:   CREATE VIEW    WORKS_ON1
      AS SELECT      Fname, Lname, Pname, Hours
         FROM        EMPLOYEE, PROJECT, WORKS_ON
         WHERE       Ssn = Essn AND Pno = Pnumber;

V2:   CREATE VIEW    DEPT_INFO(Dept_name, No_of_emps, Total_sal)
      AS SELECT      Dname, COUNT (*), SUM (Salary)
         FROM        DEPARTMENT, EMPLOYEE
         WHERE       Dnumber = Dno
         GROUP BY    Dname;
```

- View materialization: physically creating a temporary or permanent view table when the view is first queried or created and keeping that table on the assumption that other queries on the view will follow. It can be immediate, lazy (on demand) or periodic

Views (Virtual Tables)

INSERT, DELETE, or UPDATE on a view table is in many cases not possible:

- A view with a single defining table is updatable if the view attributes contain the primary key of the base relation, as well as all attributes with the NOT NULL constraint that do not have default values specified
- Views defined on multiple tables using joins are generally not updatable
- Views defined using grouping and aggregate functions are not updatable

"Some researchers have suggested that the DBMS have a certain procedure for choosing one of the possible updates as the most likely one. Some researchers have developed methods for choosing the most likely update, whereas other researchers prefer to have the user choose the desired update mapping during view definition. But these options are generally not available in most commercial DBMSs."

Views (Virtual Tables)

Views as Authorization Mechanisms:

- We can grant the user the privilege to query the view but not the base table itself
- A view can restrict a user to only see certain columns

GRANT INSERT, DELETE **ON** EMPLOYEE, DEPARTMENT **TO** A2;

Permissions — DB objects — Account

REVOKE SELECT **ON** EMPLOYEE **FROM** A3;

In this tutorial

- ~~More Complex SQL Retrieval Queries~~
- ~~Specifying Constraints as Assertions and Actions as Triggers~~
- ~~Views (Virtual Tables)~~
- Schema Change Statements

Schema Change Statements

- DROP and ALTER commands
- Behaviour options:
  - CASCADE: drop all the elements in the object or that refer to the object
  - RESTRICT: if the object has elements in it, the command will not be executed
- SQL Server does not allow you to delete a table that is referenced by a foreign constraint
- Or, if you are certain:

```
DROP TABLE T1, T2
```

with the referencing table listed first

In this tutorial

- ~~More Complex SQL Retrieval Queries~~
- ~~Specifying Constraints as Assertions and Actions as Triggers~~
- ~~Views (Virtual Tables)~~
- ~~Schema Change Statements~~

What people are researching on this domain?

- Automatic database management system tuning through large-scale machine learning - D Van Aken, A Pavlo, GJ Gordon, B Zhang - Proceedings of the 2017
- Query hint learning in a database management system - SJ Baranczyk, RP Konik, RA Mittelstadt… - US Patent App. 10 …, 2018 - Google Patents
- Performance prediction and adaptation for database management system workload using case-based reasoning approach - B Raza, YJ Kumar, AK Malik, A Anjum, M Faheem - Information Systems, 2018 – Elsevier
- A Hindi question answering system using machine learning approach - G Nanda, M Dua, K Singla - Computational Techniques in …, 2016 - ieeexplore.ieee.org
- The results above explore just the use of machine learning to DB.
  But there's much more!