## Chapter 7 – Continue …

Constraints you have seen so far:

* Key constraints
* Referential integrity constraints
* Constraints on attribute domains and NULLs
* Constraints on individual tuples within a table with the CHECK clause.

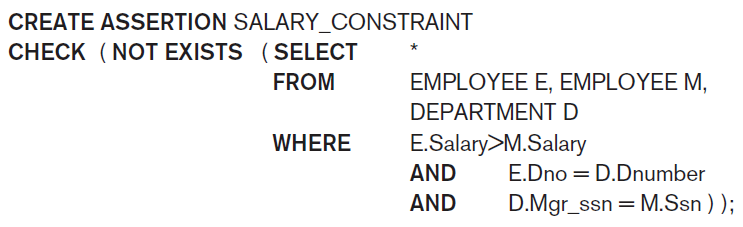
Example of CHECK clause on a tuple (and on more than one attribute):

CREATE TABLE People ( /\* From W3school.com \*/  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int,  
    City varchar(255),  
    CONSTRAINT CHK\_Person CHECK (Age>=18 AND City='Sandnes')  
);

## ASSERTION

There might be more complex constraints than the over mentioned ones in a system. For example *the salary of an employee must not be greater than the salary of the manager of the department that the employee works for.*

We cannot specify these kinds of constraints when creating a table. Verifying these constraints need several queries to be run. To define such complex constraints, SQL has ASSERTION.



### Creating assertions:

* Specify a query that selects any tuples that violate the desired condition, then put that inside NOT EXISTS ().
* The condition between CHECK parentheses must hold true on every valid database state. The DBMS is responsible for ensuring that the condition is not violated.
* Specifying constraint with CHECK clause is for one specific table, but ASSERTIONs could enforce conditions that involve more than one table.

## Introduction to TRIGGERs[[1]](#footnote-1)

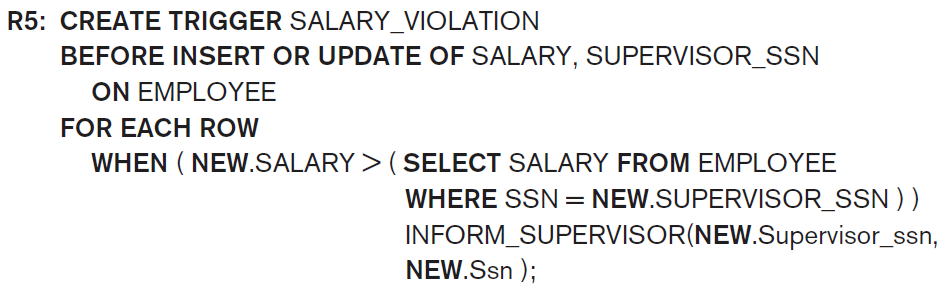
Triggers are used to specify what should be done if a constraint violated. On the other words, you specify what ACTION need to be taken in such cases.

Trigger statements composed of three components:

* Event
* Condition
* Action

i.e. ECA rule.

Lets see these components with an example:



1. Events are INSERT-ing a new record or UPDATE-ing salary or supervisor\_ssn.

Violation could happen either by inserting a new employee, changing salary to a new salary which is > that supervisor salary or by changing the employee supervisor to a person who has less salary than the employee.

1. Condition; Boolean formula between WHEN parentheses. It shows the violation case; salary of employee is > than his/her supervisor’s salary:

**NEW**.Salary > ( **SELECT** Salary **FROM** EMPLOYEE **WHERE** Ssn = **NEW**.Supervisor\_ssn )

1. Action could be a sequence of SQL statements or an external program that will be executed. In this example, the action is to execute the stored-procedure INFORM\_SUPERVISOR.

*What is Stored-Procedure?*

*A stored procedure is (like a programming method), i.e. a set of SQL statements with an assigned name, which are stored in a DBMS as a group, so it can be reused and shared by multiple programs. (chapter 10).*

## Views (Virtual Tables) in SQL

View is a table which its attribute values are derived from other tables.

Why Views?

* To avoid redundancy, we split and distribute data into many tables, but to get data back (in human readable format), we need to write queries to link tables, and that could be tedious!

To avoid the tedious work, you can define such queries as views.

* Another advantage is if you define a complex query as a view, since view results are stored in temporary files, reading from views is much faster than running the query.

Example:

**CREATE VIEW dept\_info(dept\_name, no\_of\_emps, total\_sal)**

**AS SELECT dname, count(\*), sum(salary)**

**FROM department, employee**

**WHERE department.dnumber = employee.dno**

**GROUP BY dname;**

How to use a view?

* Like normal tables (base tables), for example:

**Select \***

**From dept\_info**

**Where dept\_name = 'Research';**

View is always up-to-date and it is the responsibility of DBMS to keep it updated.

How does DBMS handle views? Does DBMS store a view data?

There are two strategies to maintain views up-to-date.

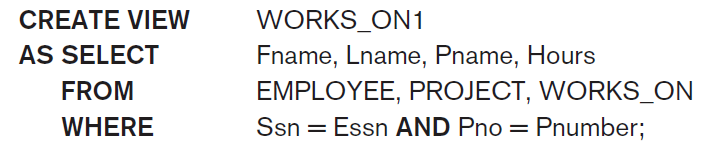
* **Strategy1: Query modification**
  1. Compute the view whenever there is a request for it. Do not store its results.
  2. Disadvantage: inefficient for views defined via complex queries that are time-consuming to execute, and for views queried very frequently.
* **Strategy 2: View materialization** 
  1. Create a temporary table when the view is first queried and store the results in.
  2. Requires efficient strategy to automatically update the view table when the base tables are updated.
  3. Multiple ways to handle materialization:
     + **immediate update strategy** updates a view as soon as the base tables are changed.
     + **lazy update strategy** updates the view whenever there is a request for the view.
     + **periodic update strategy** updates the view periodically, for example every midnight (a view query may get a result that is not up-to-date). This is commonly used in Banks, Retail store operations, etc.

**Question**: Can you write an UPDATE statement to modify a view?

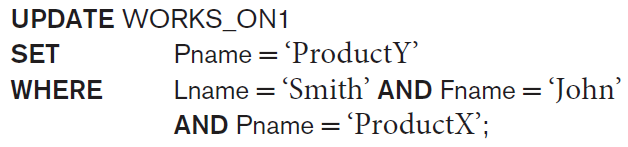
* Only if the view is based on **one** table, it does not have aggregate functions and the primary key is preserved in the view.
* For other cases it is hard to figure out which tables and under what conditions should be updated.

Example to show why updating views with more than one table is challenging:

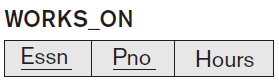
* Assume WORKS\_ON1 view is:

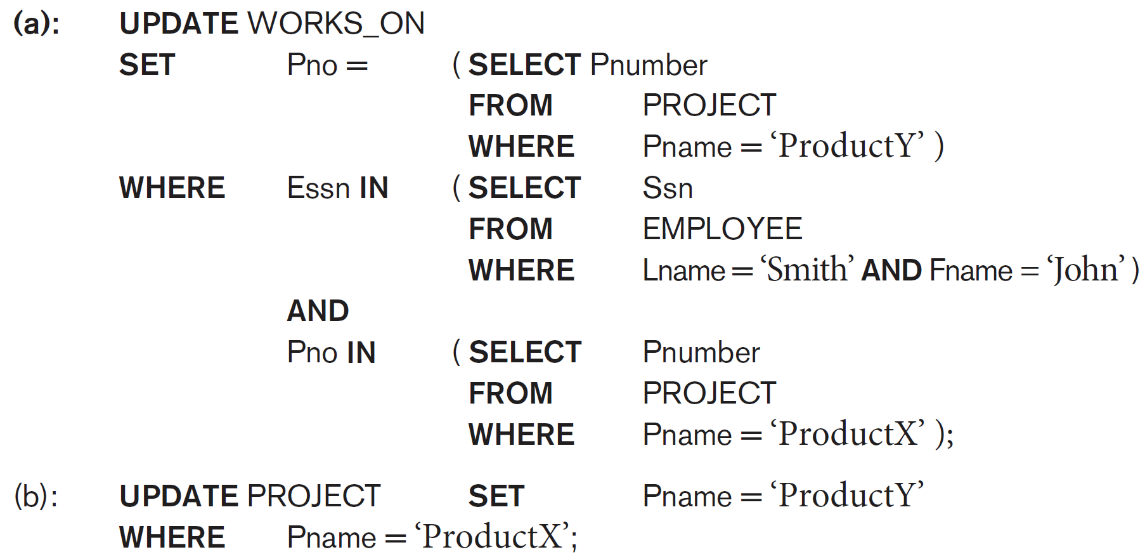


* Now imagine this update query:



* Two kinds of interpretation could happen:





Both (a) and (b) has the same results on ‘John Smith’ record in WORKS\_ON table, but (b) can have other side effects too. Moreover deriving (a) from UPDATE WORKS\_ON1 query is not straightforward for DBMS software.

### Access control via Views:

* Another application of using view is to create views for part of a database and give users access to those views. (for example to hide SSN primary key).

## Schema Change Statements in SQL

* + DROP command
  + ALTER command

#### The DROP Command

* + Used to drop named schema elements, such as tables, domains, or constraint.
  + Drop behavior options: CASCADE and RESTRICT

Example:

DROP SCHEMA COMPANY CASCADE;

This removes the schema and all its elements including tables, views, constraints, etc.

#### The ALTER table command

* **Alter table actions** include:
  1. Adding or dropping a column (attribute)
  2. Changing a column definition
  3. Adding or dropping table constraints
* Example:

ALTER TABLE COMPANY.EMPLOYEE **ADD** COLUMN Job\_title VARCHAR(12);

ALTER TABLE COMPANY.EMPLOYEE **MODIFY** COLUMN lname VARCHAR(50);

* To drop a column
  1. Choose either CASCADE or RESTRICT
  2. CASCADE would drop the column from views etc. RESTRICT is possible if no views refer to it.

Examples**:**

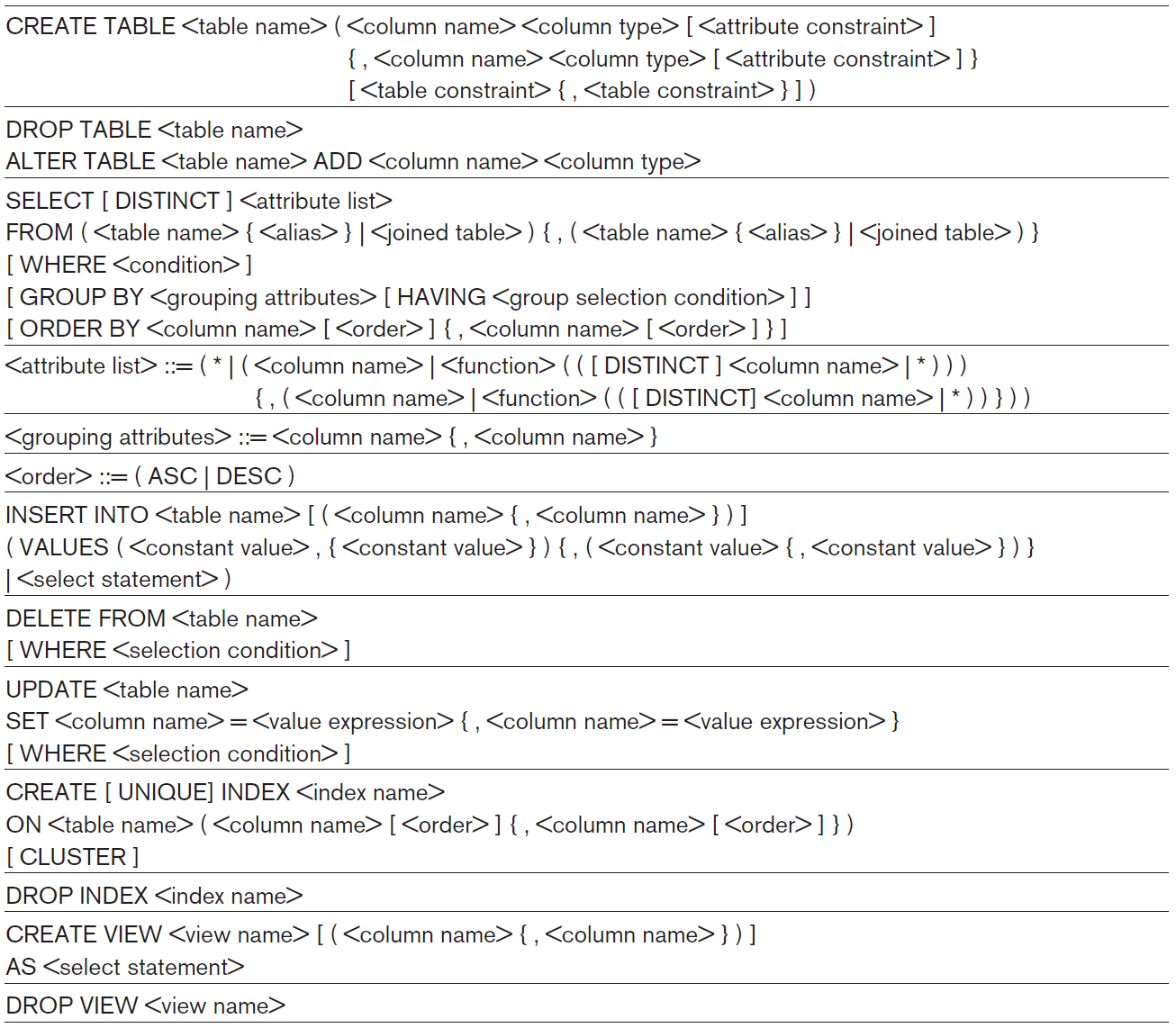
**ALTER TABLE** COMPANY.EMPLOYEE **DROP COLUMN** Address **CASCADE**;

Default values can be dropped and altered:

**ALTER TABLE** COMPANY.DEPARTMENT **ALTER COLUMN** Mgr\_ssn **DROP DEFAULT**;

**ALTER TABLE** COMPANY.DEPARTMENT **ALTER COLUMN** Mgr\_ssn **SET DEFAULT** ‘333445555’;

Summary of SQL Syntax



1. Triggers are covered in advanced databases and in chapter 26. Here we introduce the concept of triggers. [↑](#footnote-ref-1)