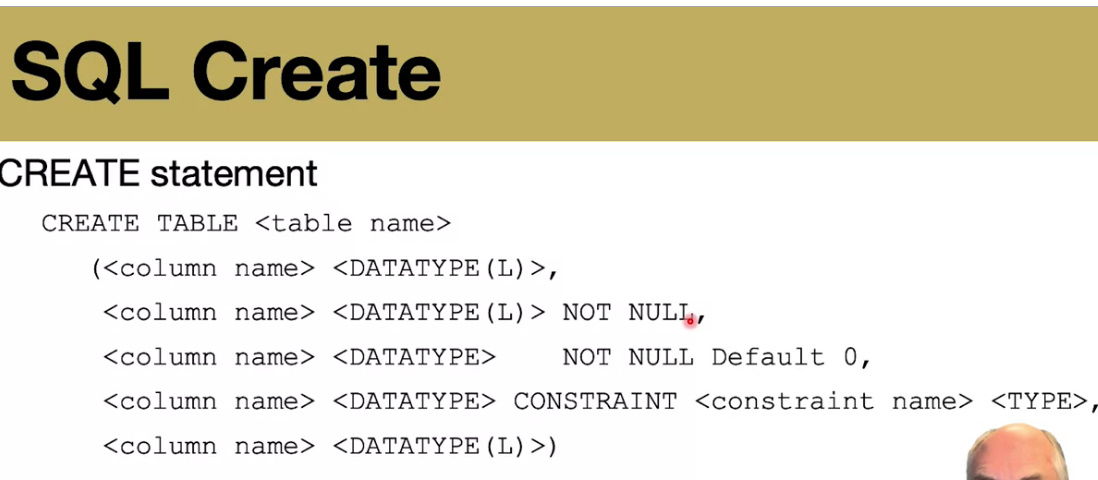
**DDL si DDM**

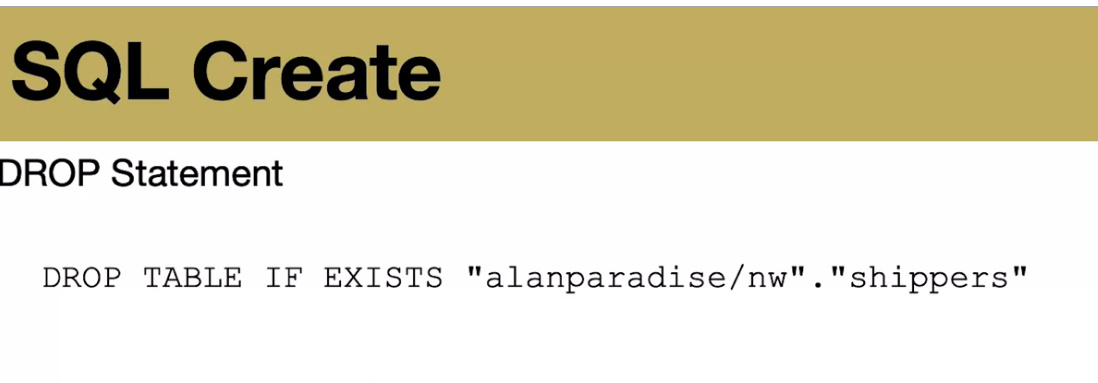
* **DDL** – Data Definition Language
* CREATE
* ALTER
* DROP
* **DML** – Data Manipulation Language
* INSERT
* UPDATE
* DELETE
* TRUNCATE

**Create**

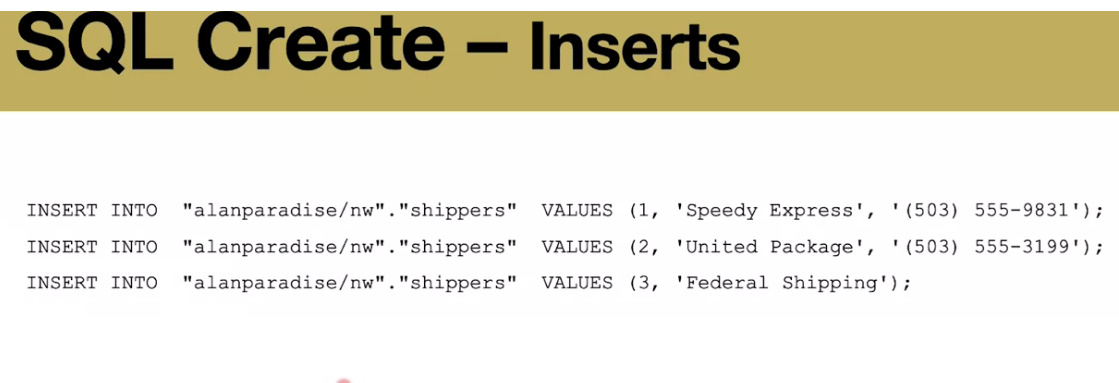


**DROP**

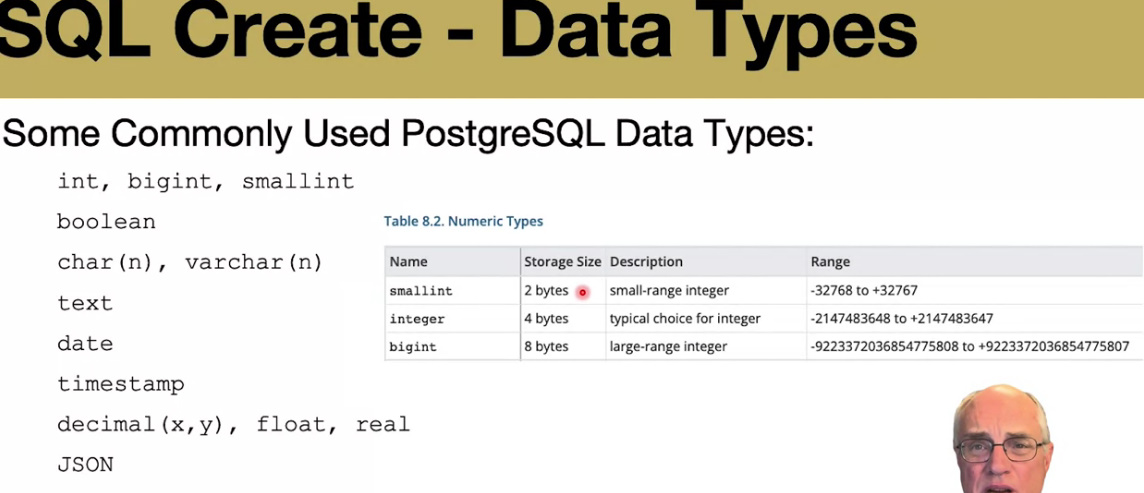
* Cand cream un tabel, e bine sa ne siguram intai ca acesta nu exista deja, iata de ce facem asa:



**Insert**

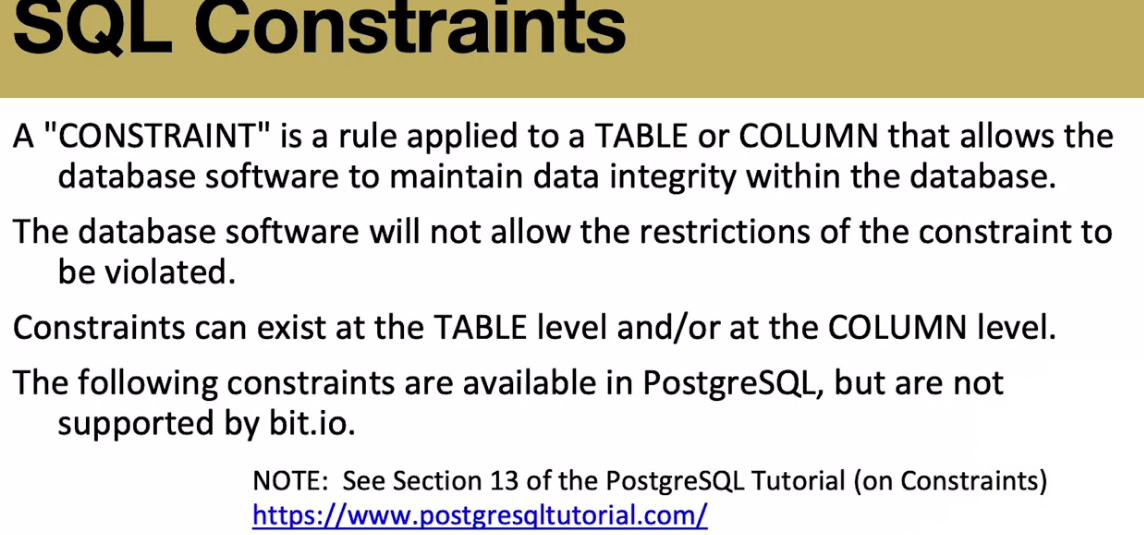


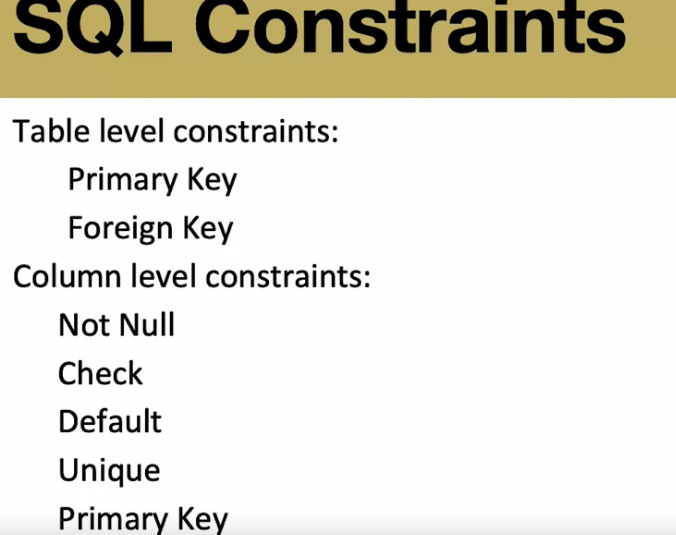
**Data Types**

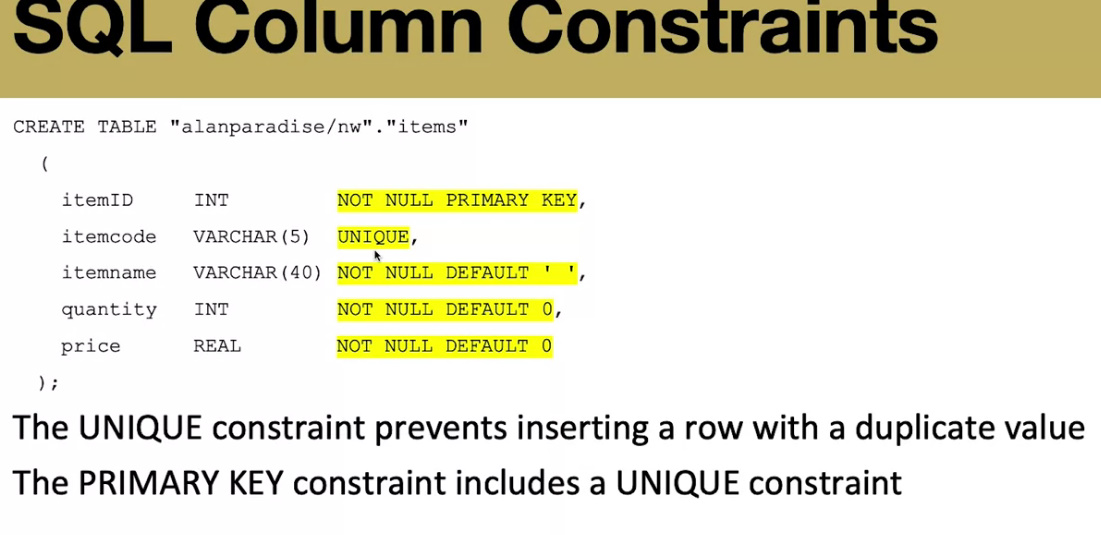


Unica diferenta dintre varchar si text e ca text nu are limita

**Constraints**

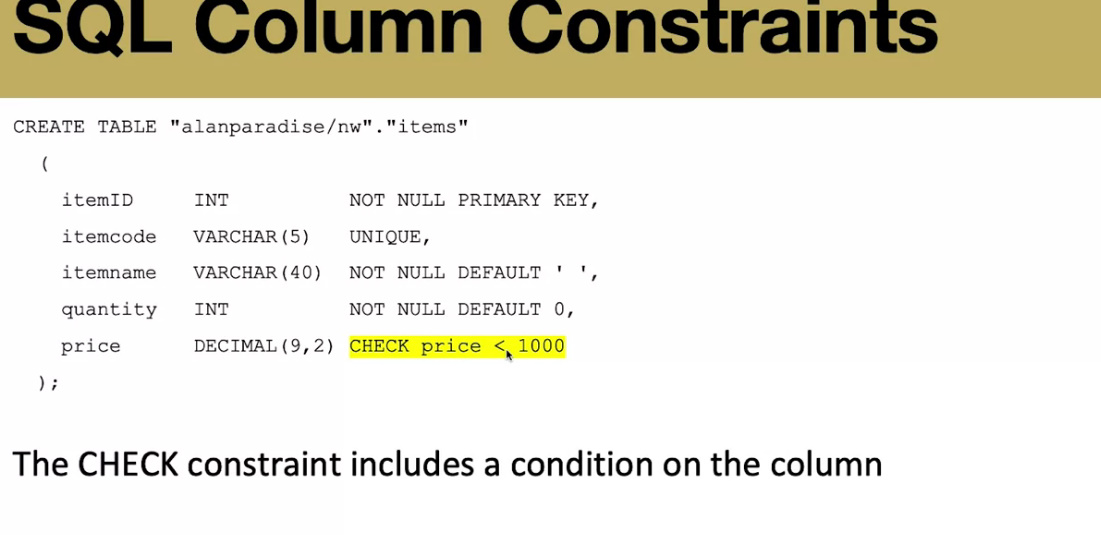


* Constranerile pot exista si la nivel de tabel, nu doar de coloana.
* 
* **Unique** – nu permite inserarea in coloana a unei valori deja existente. Primary key foloseste aceasta constrangere



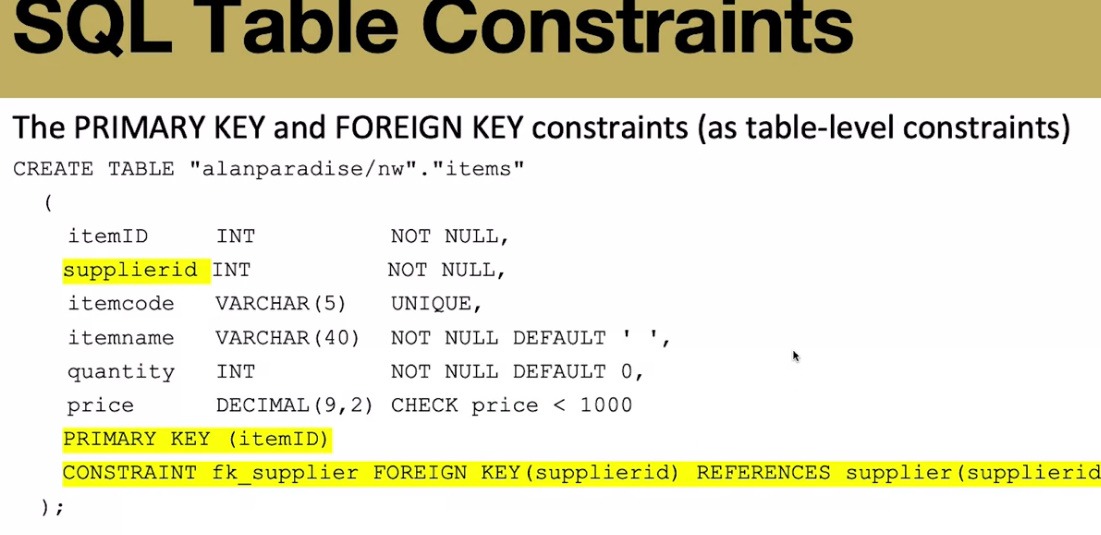


* **Check** – creaza o conditie pentru o valoare din coloana



**CHECK nu merge in MySql!!!!**

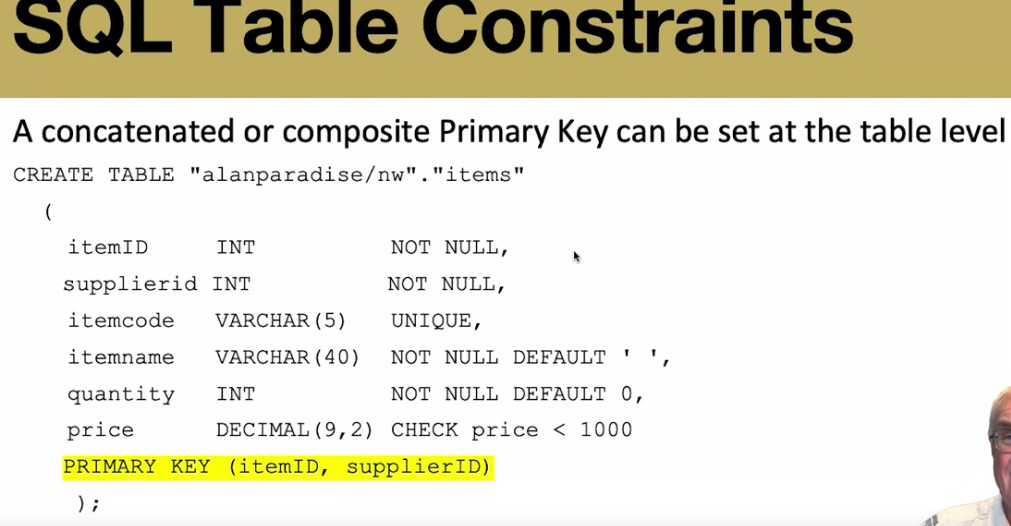
* **Primary/Foreign key**



O constrangere poate avea un nume, si asta o facem cu:

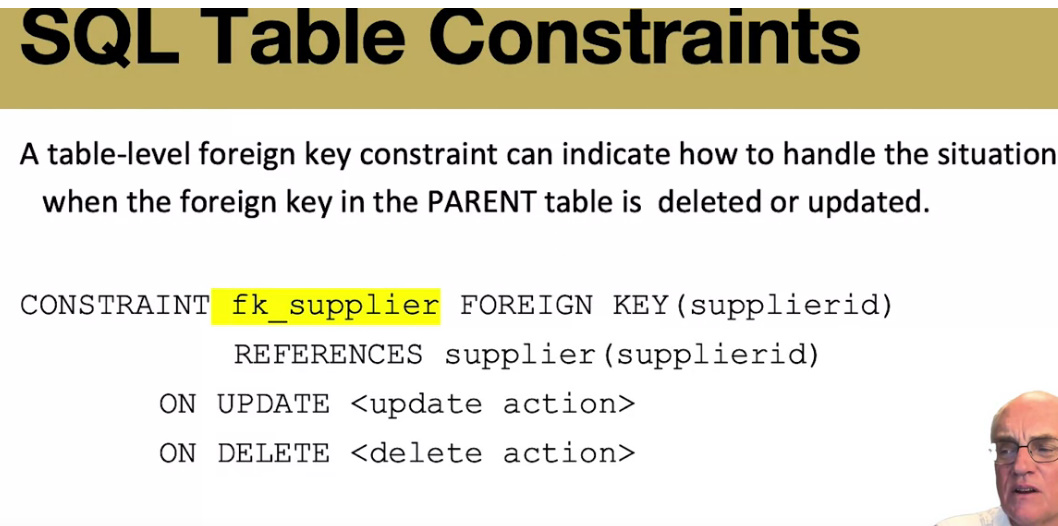
**CONSTRAINT nume**

* Nu putem sa definim mai multe coloane ca Primary Key la nivel de coloana, de asta putem sa le definim doar asa:



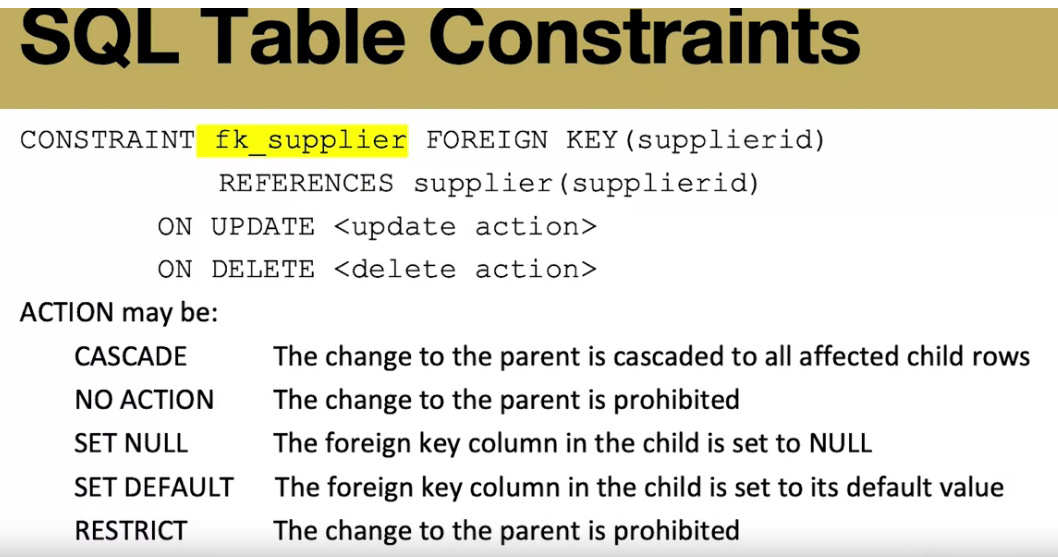
**ON UPDATE si ON DELETE**

* La definirea lui foreign key, putem decide ce sa se intample cand aceasta e modificata sau stearsa in tabelul parinte(cel in care foreign key e primary key acolo):

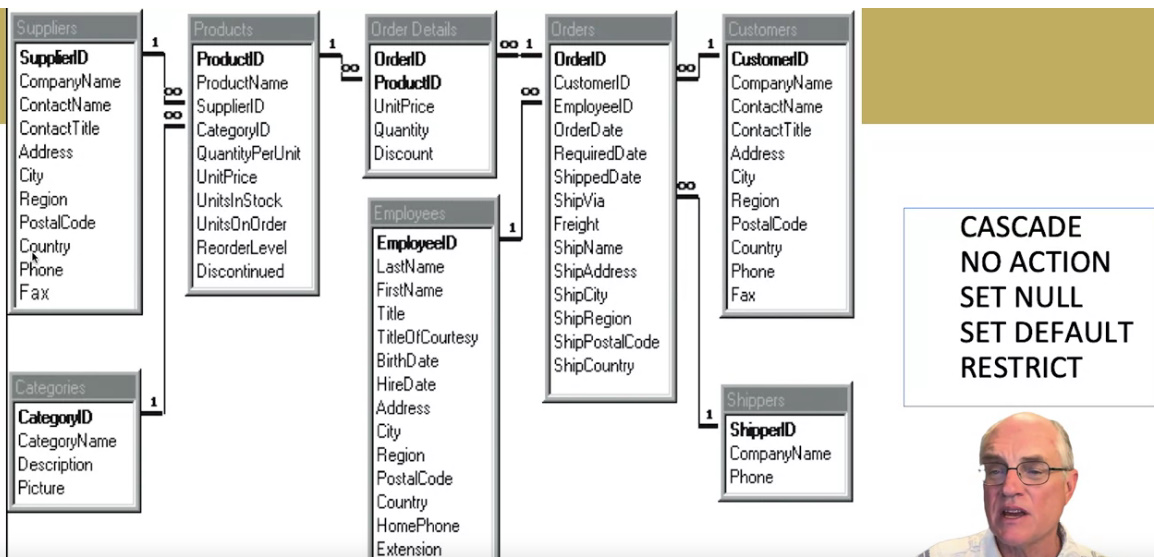


* ON arata ce se va intampla cu foreign key in tabelul in care ea se gaseste daca se modifica sau sterge valoarea ei in tabelul parinte.
* Deci, mai sus supplier e tabelul parinte, si acolo exista o coloana supplier id care e primary key, si in tabelul copil exista tot o coloana supplierid, dar care e foreign key. ON ne arata anume ce se va intampla cu foreign key cand primary key,spre care ea pointeaza in tabelul supplier, va fi modificata. Deci modificarea sau stergerea din ON se refera anume la tabelul supplier, nu la cel in care se gaseste foreign key. Daca modificam foreign key, e logic ca cu primary key din tabelul parinte nu se poate intampla nimic, dar invers poate.

**SQL Table Constraints**



* CASCADE – daca primary key se modifica in tabelul parinte, foreign key care pointeaza spre el se modifica(UPDATE). Daca punem la DELETE , **stergerea lui primary key va duce la stergerea tuturor randurilor din tabelul copil care avea acea valoare la foreign key**.
* NO ACTION – nu permite modificarea la primary key in tabelul parinte, spre care pointeaza foreign key.
* SET NULL – foreign key devine null
* SET DEFAULT – foreign key ia valoarea sa default
* RESTRICT – exact ca no action, nu permite stergerea la parent key

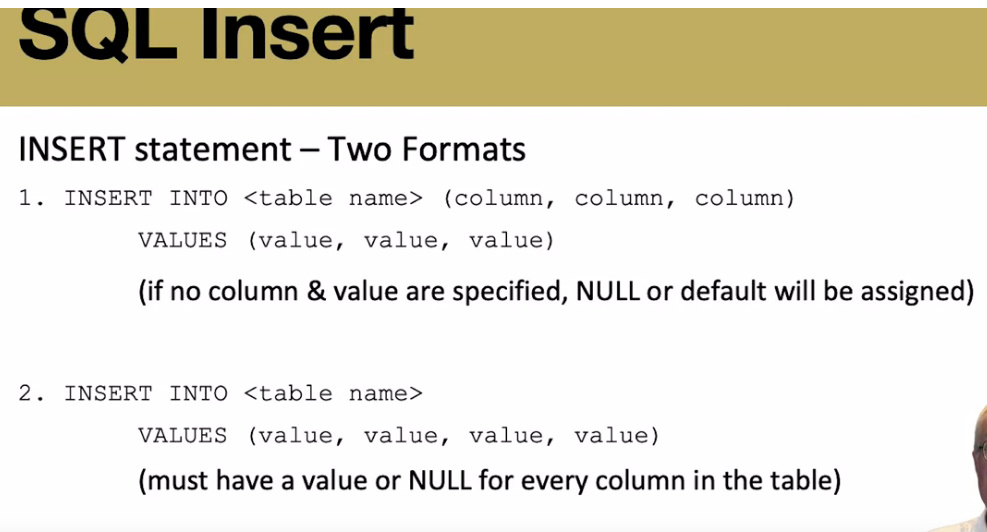


* Deci, cu cascade la update, daca modificam SupplierID din Suppliers,atunci in tabelul Products, SupplierID va primi noua valoare in toate randurile care au avut vechiul ID a lui SupplierID modificat in Suppliers.Asta daca am pus cascade la update, dar daca am pus la delete, atunci stergerea unui SupplierID din Suppliers va duce la stergerea tuturor randurilor din Products care aveau ca foreign key acel ID.
* no action nu va permite stergerea la un SupplierID din Suppliers care exista in Products

**Alter**

* ALTER TABLE Tabel RENAME TO „NumeNouTabel”
* ALTER TABLE Tabel RENAME COLUMN „Coloana” TO „NumeNouColoana”
* ALTER TABLE Tabel ADD COLUMN „Nume” Type cnstraints;
* ALTER TABLE Tabel ALTER COLUMN „Coloana ” SET DATA TYPE type;
* ALTER TABLE Tabel DROP COLUMN „Coloana”

**INSERT**



* Daca nu specificam nici-o coloana, trebuie sa introducem date pentru toate coloanele, caci SQL nu va sti care coloana anume lipseste daca scapam una
* Daca inseram numele a doar catorva coloane, restul nespecificte vor fi completate cu NULL sau defaul value

**Truncate**

* TRUNCATE TABLE „Tabel” – sterge toate liniile dintr-un tabel.
* DELETE fara WHERE e exact ca truncate

**Bulk Insert**

* Bulk insert pune in tabel multe linii deodata
* INSERT INTO „TABEL” (subquery)

