Department of Computer Science
University of Cyprus

EPL342 –Databases

Lecture 12: SQL DDL

SQL Data Definition Language

(Chapter 8.1-8.3, Elmasri-Navathe 5ED)

+ TransactSQL Reference Guide

http://msdn.microsoft.com/en-us/library/bb510741.aspx

Διδάσκων: Παναγιώτης Ανδρέου

http://www.cs.ucy.ac.cy/courses/EPL342

Περιεχόμενο Διάλεξης

- **Εισαγωγή στην SQL**
- **Τύποι Δεδομένων της SQL 1999**
  - Αριθμοί, Συμβολοσειρές, Δυαδικές Ακολουθίες, Λογικοί Τύποι, Ημερομηνίες & Ώρα, Χρονόσημα, κ.α.
  - TSQL: Συναρτήσεις Ημερομηνίας, Αυτόματη Αύξηση Τιμής, Υπολογιζόμενα Γνωρίσματα, Εξειδικευμένοι Τύποι
- **Ορισμός Δεδομένων (CREATE/DROP/ALTER)**
  - Δημιουργία Σχήματος και Πινάκων
- **Περιορισμοί (CONSTRAINTS)**
  - Οντότητας, Αναφορικής Ακεραιότητας, Ονομασία, Εντολές Ενεργοποίησης Αναφοράς, Προσωρινή Απενεργοποίηση Περιορισμών

Εισαγωγή στη SQL

- Στις προηγούμενες ενότητες μελετήσαμε μια **τυπική γλώσσα** ΒΔ, την **Σχεσιακή Άλγεβρα,** η οποία στηρίζεται πάνω στο **Σχεσιακό Μοντέλο.**
- Σε αυτή την ενότητα θα μελετήσουμε μια **πραγματική** γλώσσα βάσεων δεδομένων την **SQL (Structured Query Language)** η οποία στηρίζεται τόσο πάνω στη Σχεσιακή Άλγεβρα όσο και πάνω στον Λογισμό Πλειάδων.
- Η γλώσσα SQL χωρίζεται σε 4 **βασικές κατηγορίες (θα μελετήσουμε τις δύο πρώτες)**:
  - Γλώσσα Ορισμού Δεδομένων (Data Definition Language, **SQL-DDL**)
  - Γλώσσα Επεξεργασίας Δεδομένων (Data Manipulation Language, **SQL-DML**)
  - Γλώσσα Ορισμού Πρόσβασης (Data Control Language, **SQL-DCL**)
  - Γλώσσα Ορισμού Συναλλαγών (Transactional Control Language, **SQL-TCL**)

Εισαγωγή στην SQL (Παραδείγματα DDL/DML)

- Παράδειγμα *Γλώσσας Ορισμού Δεδομένων (Data Definition Language, **DDL**)*

```
CREATE TABLE DEPARTMENT (
    DNAME                VARCHAR(10)   NOT NULL,
    DNUMBER              INTEGER       NOT NULL,
    MGRSSN         CHAR(9),
    MGRSTARTDATE   CHAR(9)
);
```

- *Παράδειγμα Γλώσσας Επεξεργασίας Δεδομένων (Data Manipulation Language, **DML**)*

```
SELECT D.MGRSSN, D.MGRSTARTDATE,
FROM DEPARTMENT AS D
WHERE DNUMBER = 10;
```

Εισαγωγή στην SQL (Παραδείγματα DCL/TCL)

- Παράδειγμα *Γλώσσας Ορισμού Πρόσβασης (Data Control Language, **DCL**)*

```
GRANT SELECT, UPDATE
    ON My_table
    TO some_user, another_user;
```

Παράδειγμα Γλώσσας Ορισμού Συναλλαγών
Transactional Control Language, **SQL-TCL**)

Συναλλαγή - Ομαδοποίηση πολλών δηλώσεων

```
BEGIN TRANSACTION;
  UPDATE Account SET amount=amount-200 WHERE ac_no=1234;
  UPDATE Account SET amount=amount+200 WHERE ac_no=2345;

IF ERRORS=0 COMMIT;
IF ERRORS<>0 ROLLBACK;
```

Ιστορία της SQL

- ***1969:*** *Το Σχεσιακό Μοντέλο του Edgar F. Codd υλοποιείται από τη βάση* ***IBM System R***
- ***1970:*** *Ο* **Donald D. Chamberlin** και ο **Raymond F. Boyce,** επιστήμονες της IBM Almaden, δημιουργούν την πρώτη έκδοση της SQL η όποια ονομάζεται **SEQUEL**.
  - Το όνομα αλλάζει αργότερα σε SQL εφόσον το όνομα SEQUEL ήταν κατοχυρωμένο σε κάποια εταιρεία κατασκευής αεροπλάνων στην Αγγλία.
  - Την ίδια χρονιά η Relational Software, Inc., επηρεασμένο από τους Codd, Chamberlin, and Boyce, φτιάχνει την βάση δεδομένων Oracle για κρατικές υπηρεσίες των ΗΠΑ.
- ***1985:*** *Η* ***IBM*** *κάνει την* ***SQL*** *Πατέντα (US Pat. 4,506,326).*
- Οι πιο επιτυχημένες προσπάθειες **τυποποίησης**
  - SQL (ANSI 1986)
  - SQL1 (ANSI 1989)
  - SQL2 ή SQL92 (ANSI 1992)
  - **SQL3 ή SQL99 (ANSI 1999) (+OLAP, XML, object-relational etc.)**
  - Πρότυπα που έμειναν κυρίως στις συστάσεις: SQL 2003 και SQL 2008

Τύποι Δεδομένων της SQL:1999

- Σε αυτή την ενότητα θα μελετήσουμε τους **βασικούς τύπους** δεδομένων της ANSI SQL:1999.
  - Στα πλαίσια του εργαστηρίου έχετε μελετήσει ήδη αρκετούς **εξειδικευμένους τύπους** δεδομένων που ορίζονται στα πλαίσια της **TSQL-DDL** (π.χ., money, image=varbinary(max), text, κτλ.)
    - TSQL 2008 Reference: http://technet.microsoft.com/en-us/library/bb510741.aspx
  - Ο **SQL Server** έχει περισσότερους από **32** τύπους δεδομένων!
- **ANSI SQL:1999 Τύποι**
  - **Αριθμητικοί Τύποι:** Numeric, Decimal, INT, FLOAT, REAL, κτλ.
  - **Αλφαριθμητικοί (Συμβολοσειρές) Τύποι:** CHAR, VARCHAR, CLOB, NCHAR, NVARCHAR, NCLOB, κτλ.
  - **Δυαδικές Ακολουθίες:** BIT, BLOB.
  - **Λογικοί Τύποι:** BOOLEAN.
  - **Ημερομηνίες & Ώρα και Χρονόσημα:** DATE, TIME, TIMESTAMP.
  - **Δημιουργία Εξειδικευμένων Τύπων:** CREATE DOMAIN.

Τύποι Δεδομένων της SQL (**Αριθμητικοί Τύποι**)

- **Ακέραιοι αριθμοί**:
  - Δύο τύποι δεκαδικών ακεραίων αριθμών με διαφορετικό εύρος: **INTEGER (ή INT) , SMALLINT και TINYINT**
  - Το εύρος των αριθμητικών τύπων εξαρτάται από την υλοποίηση εάν και συνήθως είναι **32 bits**, **16 bits και 8 bits** αντίστοιχα.
  - Ο **SQLServer** υποστηρίζει το **BIGINT** το οποίο είναι **64 bits**.
- **Πραγματικοί Αριθμοί (Real Numbers)**:
  - 3 τύποι κινητής υποδιαστολής: **FLOAT[(precision)], REAL (➔ Float(24)), και DOUBLE PRECISION (➔ Float(53))**
  - Οι χρήστες μπορούν να καθορίσουν την ακρίβεια του **FLOAT** όχι όμως αυτή του **REAL** ή **DOUBLE PRECISION**
  - Αναπαράσταση δεκαδική ή επιστημονική (π.χ., **0.1 ή 1.0E-1)**
- **Δεκαδικοί Αριθμοί**
  - 3 τύποι δεκαδικής μορφής (όλοι υποδηλώνουν το ίδιο τύπο στη πράξη):

    DECIMAL(i,j), DEC(i,j) ή NUMERIC(i,j) (π.χ., i=3,j=2 ➔ 111.20)
    - **i** = ακρίβεια (# όλων των ψηφίων εκτός της υποδιαστολής i=18 default)
    - **j** = κλίμακα (# των δεκαδικών ψηφίων. j=0 το default)
  - Στον SQL Server δεν υπάρχει η έννοια των **unsigned αριθμών.**

Τύποι Δεδομένων της SQL (**Συμβολοσειρές**)

- Μια **Συμβολοσειρά (Αλφαριθμητική Ακολουθία)** είναι μια σειρά από *εκτυπώσιμους (printable)* χαρακτήρες οι οποίοι δηλώνονται με **μονά εισαγωγικά**: *'Hello SQL'*

- **Τύποι Συμβολοσειρών:** → *προαιρετικό*
  - ***Σταθερού μήκους n*: CHAR[(n)] ή CHARACTER[(n)]**
    - Η προκαθορισμένη τιμή του n είναι 1, αντιπροσωπεύοντας ένα μονό χαρακτήρα, δηλ., **CHAR ή CHARACTER** αντιπροσωπεύει ένα χαρακτήρα
    - *Οι χαρακτήρες που δεν αξιοποιούνται μένουν κενοί (space padding) ενώ οι κενοί χαρακτήρες δεν λαμβάνουν μέρος σε συγκρίσεις (δεν ισχύει για εξ. εφαρμογές)*
  - ***Μεταβλητού μήκους, μέγιστου μεγέθους n*: VARCHAR[(n)] ή CHAR VARYING[(n)]**
  - **Μεγάλου Μήκους (Character Large OBject – CLOB): TEXT (deprecated – θα αφαιρεθεί από μελλοντική έκδοση της MSSQL)**
    - Προσδιορίζει μεγάλες ακολουθίες (π.χ., κείμενα ή μεγάλες περιγραφές) μέχρι 2 ή 4 GB (τα αντικείμενα αυτά αποθηκεύονται συνήθως εκτός βάσης δεδομένων)
    - Πάνω σε αυτά τα πεδία δεν είναι συνήθως δυνατό να γίνουν συγκρίσεις.
    - Στον SQL Server να ορίζεται ως **VARCHAR(MAX) => δεν είναι deprecated**
  - Χαρακτήρες **UNICODE** για αναπαράσταση Ξένων Γλωσσών:
    - **NATIONAL CHARACTER[(n] ή NCHAR(N)**
    - **NATIONAL VARYING CHARACTER[(n)] ή NVARCHAR(N)**
    - **NATIONAL LARGE CHARACTER OBJECT[(n)] ή NCLOB(N)**

Τύποι Δεδομένων της SQL **(Δυαδικές Ακολουθίες)**

- Τα Bit strings είναι σειρές από **δυαδικά ψηφία** (0 ή 1) ή **NULL.** (TRUE=1, FALSE=0 ή NULL)
- **Τύποι Bit Strings**
  - ***Σταθερού μήκους n*:** BIT[(n)]
  - ***Μεταβλητού μήκους, μέγιστου μεγέθους n*:** **VARBIT[(n)] ή BIT VARYING[(n)]**
  - **Μεγάλου Μήκους (Binary Large OBject – BLOB):** Αντίστοιχα με το CLOB αλλά χωρίς ρητή κωδικοποίηση
- Η προκαθορισμένη τιμή του **n** είναι **1**.
- Ο SQL server αποθηκεύει τα bits σε byte (π.χ., 3bit = 1Byte, 9b=2B, …)
- **Μειονεκτήματα**
  - Κάποιες υλοποιήσεις (π.χ., Oracle) **δεν υποστηρίζουν** bit-strings οπόταν συνίσταται η χρήση Char(1)=Y|N
  - Δεν επιτρέπεται η **χρήση ευρετηρίων** πάνω από bit strings
    - π.χ., ευρετήριο κατακερματισμού που να βρίσκει γρήγορα τους EMPLOYEE βάσει του δυαδικού γνωρίσματος sex="M|F".

Τύποι Δεδομένων της SQL (**Ημερομηνία και Ώρα**)

- Οι τύποι δεδομένων **Ημερομηνίας** και **Ώρας** προστεθήκαν στην **SQL2**
- Τρεις τύποι χρονολογικών δεδομένων
  - **Ημερομηνία** (**DATE**)
  - **Ώρα** (**TIME**)
  - Χρονοσήμανση (**TIMESTAMP)**
- Πέρα από αυτούς τους **τύπους** η έννοια της **Ώρας/Ημερομηνίας** στις περισσότερες υλοποιήσεις **υποστηρίζεται** και υπό την μορφή **συναρτήσεων** όπως θα δούμε στη συνέχεια.

Τύποι Δεδομένων της SQL (**Ημερομηνία και Ώρα**)

- **Τύπος Δεδομένων DATE**
  - **Εξ'ορισμού μορφή** (Default format): **YYYY-MM-DD**
  - **Εύρος:** 0001-01-01 έως 9999-12-31
  - **Ακρίβεια:** Ημέρας
  - **Ημερολόγιο:** Γρηγοριανό
  - **DEFAULT τιμή:** 1900-01-01
  - **Μέγεθος:** 3 bytes στον SQL Server)
    - 3 bytes (16M days = 45K years)
    - Εάν ήταν 2bytes (65K days = 180 years)
- Παράδειγμα

```
CREATE TABLE DEPARTMENT (
    ...
    MGRSTARTDATE  DATE  NOT NULL DEFAULT 1900-01-01
);
```

Τύποι Δεδομένων της SQL (**Ημερομηνία και Ώρα**)

- **Τύπος Δεδομένων TIME [(precision)]**
  - **Εξ'ορισμού μορφή: hh:mm:ss[.nnnnnnn]**
  - **Εύρος:** 00:00:00.0000000 εως 23:59:59.9999999
  - **Ακρίβεια:** 100 nanoseconds (almost microsec.)
  - **DEFAULT τιμή:** 00:00:00
  - **Μέγεθος:** 5 bytes (στον SQL Server)
- **Τύπος Δεδομένων TIMESTAMP[(precision)]**
  - **Συνδυάζει την ώρα μαζί μαζί την ημερομηνία**
  - Στον SQL Server 2008 ονομάζεται **datetime2** (YYYY-MM-DD hh:mm:ss[.nnnnnnn])
  - Υπάρχουν και τα **datetime** (YYYY-MM-DD hh:mm:ss[.nnn] - millisecond) και **smalldatetime** (YYYY-MM-DD hh:mm:ss - second)

Τύποι Δεδομένων της SQL (**Ημερομηνία και Ώρα**)

Εξειδικευμένοι Τύποι SQL Server 2008
http://msdn.microsoft.com/en-us/library/ms187752.aspx

Data type	Format	Range	Accuracy	Storage size (bytes)
time	**hh:mm:ss[.nnnnnnn]**	00:00:00.0000000 through 23:59:59.9999999	100 nanoseconds	3 to 5
date	**YYYY-MM-DD**	0001-01-01 through 9999-12-31	1 day	3
smalldatetime	**YYYY-MM-DD hh:mm:ss**	1900-01-01 through 2079-06-06	1 minute	4
datetime	**YYYY-MM-DD hh:mm:ss[.nnn]**	1753-01-01 through 9999-12-31	0.00333 second	8
datetime2	**YYYY-MM-DD hh:mm:ss[.nnnnnnn]**	0001-01-01 00:00:00.0000000 through 9999-12-31 23:59:59.9999999	100 nanoseconds	6 to 8
datetimeoffset	**YYYY-MM-DD hh:mm:ss[.nnnnnnn] [+|-]hh:mm**	0001-01-01 00:00:00.0000000 through 9999-12-31 23:59:59.9999999 (in UTC)	100 nanoseconds	8 to 10

Τύποι Δεδομένων της SQL (**Συναρτήσεις** Ημερομηνίας και Ώρας)

- Πέρα από τους τύπους για ημερομηνίες, όλα τα συστήματα παρέχουν και συναρτήσεις:
  - **Επεξεργασίας** της ώρας/ημερομηνίας
  - **Πρόσθεση** Ημερομηνιών / Ωρών, Εξαγωγή μήνα, ημέρας, ή χρονολογίας από μια ημερομηνία,
  - **Εμφάνιση** ημερομηνίας με διάφορους τρόπους, κτλ.
  - Αυτές θα αξιοποιηθούν αργότερα από την SQL-DML

- **Παραδείγματα Συναρτήσεων από τον SQL Server:**
  - **DATEDIFF (datepart , startdate , enddate)**
    - SELECT DATEDIFF(year, '2005-12-31' '2006-01-01'); ➔ Returns 1
  - **DATEADD (datepart , number, date)**
    - SELECT DATEADD(month, 1, '2006-08-30'); ➔ Returns 2006-09-30
  - **DATEPART(datepart , date)**
    - SELECT DATEPART(year, 0); ➔ Returns: 1900
    - **ISDATE (expression)**
    - SELECT ISDATE('04/15/2008'); ➔ Returns 1

Datepart: year, month, dayofyear, day,week, weekday,hour,second,millisecond

Τύποι Δεδομένων της SQL (**Αυτόματη Αύξηση Τιμής** - Autoincrement)

- Σε πολλές περιπτώσεις είναι χρήσιμο **να δημιουργούνται** αυτόματα κάποιοι **μοναδικοί αριθμοί** από την βάση δεδομένων (π.χ., για να χρησιμοποιηθούν ως πρωτεύων κλειδιά)
  - Π.χ., Κατά την **εισαγωγή προϊόντων** (products) σε μια βάση θέλουμε να παράγεται αυτόματα το κλειδί **PID**.
- Εάν και η **SQL:99** δεν έχει πρόνοια για ένα τέτοιο τύπο, όλες οι βάσεις τον υλοποιούν με κάποιο τρόπο.
- Συγκεκριμένα, το **Auto-increment** υλοποιείται ως **ειδικό τύπο ακεραίου** ο οποίος αυξάνεται αυτόματα
  - **PostgreSQL:**
    - **Serial**: Από 1 μέχρι 2147483647
    - **Bigserial**: Από 1 μέχρι 9223372036854775807
  - **Microsoft Access: Autonumber**
  - **Oracle: CREATE SEQUENCE command**
  - **SQL Server 2008: IDENTITY**

Τύποι Δεδομένων της SQL

(**Αυτόματη Αύξηση Τιμής** - Autoincrement)

- Στο SQL Server 2008, η αυτόματη αύξηση γίνεται μέσω του **IDENTITY**, το οποίο δίνεται ως γνώρισμα στην εντολή CREATE TABLE, δηλ:
- **IDENTITY [(*seed* , *increment*)]**
  - **Seed:** Αρχή Αρίθμησης, **Increment:** Αύξηση Μετρητή
  - Η εξ' ορισμού τιμή είναι (1,1) **ή** πρέπει να ορίζονται και τα δυο.
- **Παράδειγμα Χρήσης**

  CREATE TABLE Products (PID int **IDENTITY(1,1) PRIMARY KEY, .**)
- **Επισημάνσεις:**
  - Το IDENTITY χρησιμοποιείται σε συνδυασμό με κάποια **αριθμητική τιμή** (tinyint, smallint, bigint, dec(p,0)) πράξη μόνο με **int** ή **bigint.**
  - Εάν κάνουμε **DROP ή TRUNCATE** ένα πίνακα τότε ο **counter** του **IDENTITY** μηδενίζεται (για αυτό να χρησιμοποιείται το **DELETE**)
  - Εάν χρησιμοποιείται ως **PRIMARY KEY** χρειάζεται προσοχή ειδικά εάν τα δεδομένα πρόκειται να φορτωθούν στη βάση δεδομένων.
  - Όμοια, ο τύπος δεδομένων **UniqueIdentifier (GUID)** είναι 16byte συμβολοσειρά, π.χ., '6F9619FF-8B86-D011-B42D-00C04FC964FF' η οποία ανατίθεται αυτόματα με **DEFAULT NEWID()**

Τύποι Δεδομένων της SQL

(**Υπολογιζόμενα Γνωρίσματα** - Computed Columns)

- Σε κάποιες περιπτώσεις είναι χρήσιμο **να δημιουργούνται** αυτόματα κάποια γνωρίσματα βάσει άλλων γνωρισμάτων
- Π.χ., ListPrice = Price * 1.2;
- Ορισμένες βάσεις δεδομένων, όπως ο SQL Server υποστηρίζουν την έννοια των **Υπολογιζόμενων Γνωρισμάτων (Computed Columns)** τα οποία γνωρισματα δεν υπάρχουν αλλά υπολογίζονται κατά την αναφορά τους.

```
CREATE TABLE Grades (
    Midterm float, Final float, Exercise float,
    FinalGrade AS 0.25*Midterm + 0.5*Final + 0.25*Exercises)
```

- **Μειονεκτήματα**
  - Δεν είναι συμβατό με την SQL:99 οπόταν υπάρχουν προβλήματα μεταφερσιμότητας (portability) του κώδικα
  - Σε αρκετές περιπτώσεις, πολλές λειτουργίες δεν υποστηρίζονται (π.χ., ευρετήρια, περιορισμοί, κτλ.)

Τύποι Δεδομένων της SQL
(Δημιουργία **Εξειδικευμένων Τύπων** Δεδομένων)

- Η SQL:99 παρέχει την δυνατότητα δημιουργίας **εξειδικευμένων τύπων δεδομένων** μέσω της εντολής **CREATE DOMAIN**.
- Η αντίστοιχη εντολή στην TSQL είναι η **CREATE TYPE**
- **Παραδείγματα σε SQL:99:**

```
CREATE DOMAIN name_dom AS CHAR(20);
CREATE DOMAIN sectno_dom AS SMALLINT;
CREATE DOMAIN section_dom VARCHAR(20) DEFAULT 'none';
CREATE DOMAIN address_dom CHAR(50) DEFAULT NULL;
CREATE DOMAIN qpa_dom DECIMAL (3,2) DEFAULT 0.00;
```

- **Παραδείγματα σε TSQL:**

```
CREATE TYPE SSN FROM varchar(11) NOT NULL ;
```

http://msdn.microsoft.com/en-us/library/ms175007.aspx

Τύποι Δεδομένων της SQL
(Δημιουργία **Εξειδικευμένων Τύπων** Δεδομένων)

- Εάν και θα δούμε την έννοια των περιορισμών αργότερα, αναφέρουμε ότι η SQL99 **υποστηρίζει περιορισμούς CHECK** και κατά την **δημιουργία εξειδικευμένων τύπων.**
- **Παραδείγματα σε SQL:99**
  - CREATE DOMAIN gender_dom AS CHAR(1)
    CHECK (VALUE IN ('F', 'f', 'M', 'm'));
  - CREATE DOMAIN gender_dom AS CHAR(1)
    CHECK (VALUE IN ('F', 'f', 'M', 'm') OR (VALUE IS NULL));
  - CREATE DOMAIN ssn_dom CHAR(11)
    CHECK ((VALUE BETWEEN '000-00-0000' AND '999-99-9999'));
  - CREATE DOMAIN hour_dom AS INTEGER DEFAULT 0
    CHECK (VALUE >= 0);

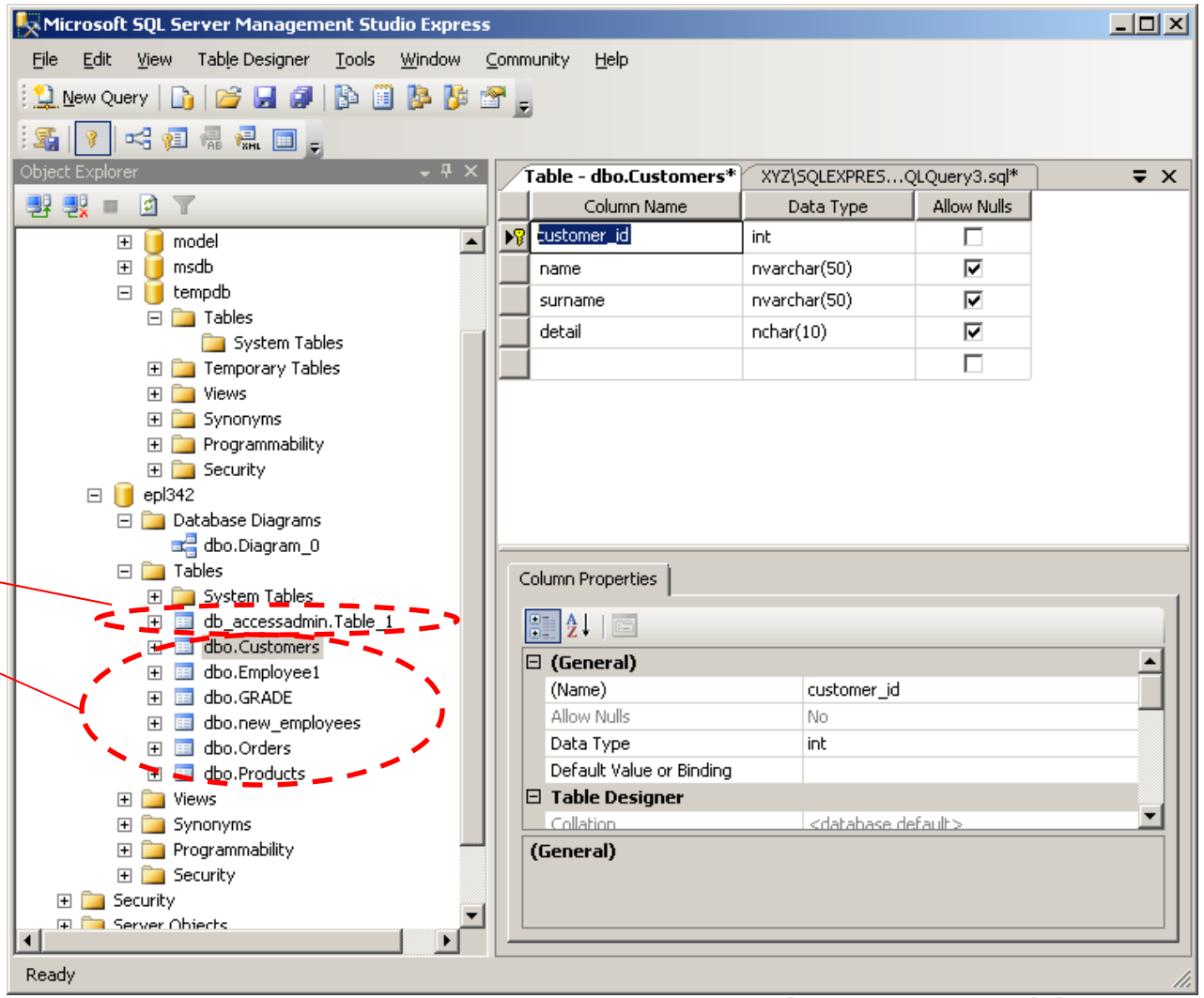
Ορισμός Δεδομένων στην SQL (Δημιουργία Σχήματος)

- Σε νεότερες εκδόσεις του SQL Server μπορούμε να **αναθέσουμε επί μέρους** αντικείμενα της βάσης σε επί **μέρους Ομάδες** (διαφορετικά Σχήματα Βασης), π.χ.,
- **Command: CREATE SCHEMA Person**

  SQL Server 2008 Schema => Container of Objects
- Το παράδειγμα στα δεξιά δείχνει

  δυο ομάδες Person και HumanResources.
  - Η default ομάδα είναι ο **dbo (database owner)**
- Συνεπώς μια αναφορά σε ένα αντικείμενο χρησι
  έξης σημειογραφία:
- **[ServerName.[DatabaseName.[SchemaName.]]].Objec**

Ορισμός Δεδομένων στην SQL
(Δημιουργία Σχήματος)



Διαφορετικά Σχήματα

Ορισμός Δεδομένων στην SQL (Data Definition Language - DDL)

- Μια ***Γλώσσα Ορισμού Δεδομένων (Data Definition Language, SQL-DDL)*** μας επιτρέπει να **δημιουργήσουμε** και να διαχειριστούμε τις δομές μιας βάσης δεδομένων.
- Αυτό επιτυγχάνεται μέσω κάποιων εντολών της SQL, π.χ.,:
  - **CREATE** δημιουργεί ένα αντικείμενο βάσης
    - π.χ., σχήμα βάσης, πίνακα, ευρετήριο, σκανδάλη, τύπο δεδ., κτλ.
  - **DELETE/DROP/TRUNCATE**
    - **DELETE** διαγράφει στοιχεία από το αντικείμενο βάσει συνθήκης (χωρίς να αποδεσμεύεται ο χώρος).
    - **TRUNCATE/DROP:** διαγράφει το αντικείμενο / διαγράφει το αντικείμενο και την περιγραφή του αποδεσμεύοντας και τον χώρο.
  - **ALTER** μεταβάλει την δομή ενός αντικειμένου
    - Π.χ., προσθήκη στήλης, επιπλέον περιορισμού, κτλ.

Ορισμός Δεδομένων στην SQL (CREATE TABLE)

- Από την SQL2, μπορούμε με την χρήση της εντολής CREATE TABLE να ορίσουμε και τους ακόλουθους περιορισμούς:
  - **Γνωρίσματα**
    - Ορίζεται ο τύπος δεδομένων κάθε γνωρίσματος, π.χ., age int,
  - **Πρωτεύοντα** κλειδιά (**PRIMARY KEY**)
    - Ορίζεται το Κλειδί κάθε Σχέσης
    - Δεν μπορεί να είναι NULL
    - Δεν είναι απαραίτητο να υπάρχει εάν και συνίσταται να υπάρχει
  - **Δευτερεύοντα** κλειδιά (**UNIQUE**)
    - Ορίζονται τα μοναδικά γνωρίσματα μιας Σχέσης (π.χ., Dname στο Department)
    - Μπορεί να είναι NULL (βασική διαφορά από τα Primary Keys)
  - **Κανόνες Αναφορικής Ακεραιότητας** (**FOREIGN KEY**).
    - Για ένα ξένο κλειδί, αναφέρει σε ποιο **πρωτεύων κλειδί** αναφέρεται καθώς και εντολής ενεργοποίησης αναφοράς. (πράξεις αντιδράσεις σε διαγραφές και ενημερώσεις)

Ορισμός Δεδομένων στην SQL (CREATE TABLE)

- Παράδειγμα δημιουργίας πίνακα με βασικούς περιορισμούς και **χωρίς δράσεις αντίδρασης** σε **διαγραφές** και **ενημερώσεις**

Το γνώρισμα ΔΕΝ μπορεί να είναι NULL. Default: NULL (δηλαδή επιτρέπεται να είναι NULL)

```
CREATE TABLE DEPT (
  DNAME     VARCHAR(10)  NOT NULL [UNIQUE],
  DNUMBER   INTEGER NOT NULL [PRIMARY KEY]
  MGRSSN    CHAR(9) [REFERENCES EMP],
  MGRSTARTDATE CHAR(9),
   PRIMARY KEY (DNUMBER),
   UNIQUE (DNAME),
   FOREIGN KEY (MGRSSN) REFERENCES EMP
);
```

Για **σύνθετα κλειδιά** επιβάλλεται να ορίζεται στο τέλος και όχι δίπλα από τον ορισμό του γνωρίσματος.

Ορισμός Δεδομένων στην SQL (CREATE TABLE - Παράδειγμα)

```
CREATE TABLE EMPLOYEE
    ( Fname          VARCHAR(15)      NOT NULL,
      Minit          CHAR,
      Lname          VARCHAR(15)      NOT NULL,
      Ssn            CHAR(9)          NOT NULL,
      Bdate          DATE,
      Address        VARCHAR(30),
      Sex            CHAR,
      Salary         DECIMAL(10,2),
      Super_ssn      CHAR(9),
      Dno            INT              NOT NULL,
    PRIMARY KEY (Ssn),
    FOREIGN KEY(Super_ssn) REFERENCES EMPLOYEE(Ssn),
    FOREIGN KEY(Dno) REFERENCES DEPARTMENT(Dnumber) );
```

Σημειώστε ότι ακόμη δεν ορίσαμε την αντίδραση σε διαγραφές και ενημερώσεις)

Μπορούσαν να οριστούν δίπλα από το γνωρίσματα

```
CREATE TABLE DEPARTMENT
    ( Dname          VARCHAR(15)      NOT NULL,
      Dnumber        INT              NOT NULL,
      Mgr_ssn        CHAR(9)          NOT NULL,
      Mgr_start_date DATE,
    PRIMARY KEY(Dnumber),
    UNIQUE(Dname),
    FOREIGN KEY(Mgr_ssn) REFERENCES EMPLOYEE(Ssn) );
```

```
CREATE TABLE DEPENDENT
    ( Essn           CHAR(9)          NOT NULL,
      Dependent_name VARCHAR(15)      NOT NULL,
      Sex            CHAR,
      Bdate          DATE,
      Relationship   VARCHAR(8),
    PRIMARY KEY(Essn, Dependent_name),
    FOREIGN KEY(Essn) REFERENCES EMPLOYEE(Ssn) );
```

Σύνθετα κλειδιά ορίζονται πάντα στο τέλος

Ορισμός Δεδομένων στην SQL (CREATE TABLE - Παράδειγμα)

```
CREATE TABLE DEPT_LOCATIONS
        ( Dnumber                   INT                 NOT NULL,
          Dlocation                 VARCHAR(15)         NOT NULL,
        PRIMARY KEY(Dnumber, Dlocation),
        FOREIGN KEY(Dnumber) REFERENCES DEPARTMENT(Dnumber) );
```

```
CREATE TABLE PROJECT
        ( Pname                     VARCHAR(15)         NOT NULL,
          Pnumber                   INT                 NOT NULL,
          Plocation                 VARCHAR(15),
          Dnum                      INT                 NOT NULL,
        PRIMARY KEY(Pnumber),
        UNIQUE(Pname),
        FOREIGN KEY(Dnum) REFERENCES DEPARTMENT(Dnumber) );
```

```
CREATE TABLE WORKS_ON
        ( Essn                      CHAR(9)             NOT NULL,
          Pno                       INT                 NOT NULL,
          Hours                     DECIMAL(3,1)        NOT NULL,
        PRIMARY KEY(Essn, Pno),
        FOREIGN KEY(Essn) REFERENCES EMPLOYEE(Ssn),
        FOREIGN KEY(Pno)  REFERENCES PROJECT(Pnumber) );
```

Ορισμός Δεδομένων στην SQL (CREATE TABLE – Constraint Ordering)

- Οι **αναφορικοί περιορισμοί** μπορούν να ορίζονται σε επίπεδο i) **γνωρίσματος**, ii) **σχέσης** ή iii) **με μεταβολή σχήματος** (με ALTER TABLE που θα δούμε σε λίγο)
- Ο **βολικότερος** τρόπος είναι με **μεταβολή σχήματος** εφόσον μπορεί να υπάρχουν προβλήματα **προτεραιότητας** όπως το πιο κάτω και να **μην μπορούμε** να προχωρήσουμε.

Πρόβλημα Προσέγγισης (i) και (ii)

```
CREATE TABLE EMPLOYEE(
    SSN INTEGER PRIMARY KEY,
    DNO INTEGER NOT NULL,
      FOREIGN KEY (DNO) REFERENCES
                    DEPARTMENT(DNO));

CREATE TABLE DEPARTMENT(
    DNO INTEGER NOT NULL PRIMARY KEY,
    MGRSSN INTEGER NOT NULL,
      FOREIGN KEY (MGRSSN) REFERENCES EMPLOYEE(SSN));
```

Μήνυμα Λάθους

Foreign key 'FK__EMPLOYEE__DNO__4CA06362' references invalid table 'DEPARTMENT'.

Ορισμός Δεδομένων στην SQL (CREATE TABLE – Constraint Ordering)

- Επίλυση **με μεταβολή σχήματος** κάνοντας χρήση του ALTER TABLE

/* Δημιουργία Πινάκων χωρίς αναφορικούς περιορισμούς */

```
CREATE TABLE EMPLOYEE(
    SSN INTEGER PRIMARY KEY,
    DNO INTEGER NOT NULL);

CREATE TABLE DEPARTMENT(
    DNO INTEGER NOT NULL PRIMARY KEY,
    MGRSSN INTEGER NOT NULL);
```

Η ALTER θα μελετηθεί εκτενεστέρα σε λίγο

/* Δημιουργία Αναφορικών Περιορισμών με ALTER TABLE */

```
ALTER TABLE EMPLOYEE ADD CONSTRAINT
      FOREIGN KEY (DNO) REFERENCES  DEPARTMENT(DNO));
ALTER TABLE DEPARTMENT ADD CONSTRAINT
      FOREIGN KEY (MGRSSN) REFERENCES EMPLOYEE(SSN));
```

Διαγραφή Δεδομένων στην SQL (DROP TABLE)

- Η **Drop** χρησιμοποιείται για να **αφαιρέσει μια σχέση** και τον **ορισμό της** (σχήμα) από τον **κατάλογο** της βάσης:

  Π.χ., DROP TABLE EMPLOYEE, DEPARTMENT,…,n;

A. Δεν επιτρέπεται να κάνουμε DROP πίνακα που **αναφέρεται από κάποιο άλλο πίνακα (πρώτα DROP τα constraint)**
   - `ALTER TABLE dbo.Employee DROP CONSTRAINT fk_dno`

B. Μπορούμε να κάνουμε DROP **πολλαπλούς πίνακες** δεδομένου του ότι τους κάνουμε drop με ακολουθία που δεν επηρεάζει το σημείο A

- **DELETE** διαγράφει εγγραφές βάσει συνθήκης και καταγράφοντας τις αλλαγές σε ένα κατάστιχο (LOG)

  Π.χ., DELETE [*] FROM DEPENDENT [WHERE condition];

- **TRUNCATE:** διαγράφει ΟΛΕΣ τις εγγραφές μιας σχέσης και δεν καταγράφει τις αλλαγές σε κατάστιχο (LOG)

  Π.χ., TRUNCATE TABLE DEPENDENT;

Εξέλιξη Σχήματος Πινάκων (Η εντολή ALTER)

- Η εντολή **ALTER** μας επιτρέπει την μετεξέλιξη του σχήματος μιας βάσης (π.χ., προσθήκη ή αλλαγή πεδίου, περιορισμών, κτλ.)
- Ας δούμε κάποια **παραδείγματα** (δείτε και τα παραδείγματα του εργαστηρίου)

Α) Προσθήκη Γνωρίσματος

```
ALTER TABLE Company.Employee
    ADD COLUMN LastRaise date;
```

- ***Παρατηρήσεις***
  - *Η στήλη δημιουργείται αλλά φέρει την τιμή **NULL**.*
  - *Μπορούμε να κάνουμε τα ακόλουθα:*
    - **Προσθήκη Σταθερής Τιμής**
      ADD COLUMN LastRaise date NOT NULL **DEFAULT '2009-01-01'**
    - **Προσθήκη Μεταβαλλόμενης Τιμής (μέσω συνάρτησης TSQL)**
      ADD COLUMN LastRaise date NOT NULL **DEFAULT GETDATE()**
    - **Το DEFAULT θα μελετηθεί εκτενεστέρα στη συνέχεια**

Εξέλιξη Σχήματος Πινάκων (Η εντολή ALTER)

Β) Διαγραφή Γνωρίσματος

```
ALTER TABLE Company.Employee
    DROP COLUMN LastRaise;
```

- ***Παρατηρήσεις***
  - Εαν η στήλη **υπό διαγραφή αναφέρεται** από κάποια άλλη σχέση τότε δεν είναι εφικτή η διαγραφή (εφόσον παραβιάζεται ο κανόνας αναφορικής ακεραιότητας).
    - Στην SQL:99 (όχι TSQL) υπάρχει η έννοια του CASCADE
      ALTER TABLE tname DROP COLUMN cname CASCADE;
      το οποίο ακυρώνει οποιεσδήποτε αναφορές στο column_name (constraints, views, κτλ) προτού γίνει η διαγραφή.
- *Η ALTER έχει πολλές επιλογές, π.χ.,:*
  - **ALTER TABLE tname ALTER COLUMN cname DROP DEFAULT;**
  - **ALTER TABLE tname ALTER COLUMN cname SET DEFAULT 1;**
- Για την πλήρη σύνταξη της ALTER σε TSQL
  - http://msdn.microsoft.com/en-us/library/ms190273.aspx

Επιπλέον Περιορισμοί στην SQL

- Σε αυτή την ενότητα θα μελετήσουμε τους **περιορισμούς** που μπορούμε να διατυπώσουμε με την **SQL-DDL.**
  - Θα κάνουμε ιδιαίτερη αναφορά στους περιορισμούς στα πλαίσια της TSQL-DDL (SQL Server 2008).
- Πέρα από τους περιορισμούς που είδαμε ήδη **NULL/NOT NULL, PRIMARY KEY, UNIQUE και FOREIGN KEY θα** μελετήσουμε και τα ακόλουθα:
  - **Ονομασία Περιορισμών** (Naming)
  - **Περιορισμοί DEFAULT**
  - **Περιορισμοί CHECK**
  - **Εντολές Ενεργοποίησης Αναφοράς**
    - **ON DELETE | ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }**
  - **Προσωρινή Απενεργοποίηση Περιορισμών**
    - **WITH CHECK / WITH NO CHECK**

Περιορισμοί στην SQL (**Ονομασία Περιορισμών**)

- Στα προηγούμενα παραδείγματα δεν δώσαμε κάποια **ονόματα** στους **περιορισμούς** ,π.χ.,

```
CREATE TABLE DEPARTMENT
        ( Dname                 VARCHAR(15)             NOT NULL,
          Dnumber               INT                     NOT NULL,
          Mgr_ssn               CHAR(9)                 NOT NULL,
          Mgr_start_date        DATE,
        PRIMARY KEY(Dnumber),
        UNIQUE(Dname),
        FOREIGN KEY(Mgr_ssn) REFERENCES EMPLOYEE(Ssn) );
```

- Αυτό που γίνεται είναι ότι η DBMS φτιάχνει τα δικά της ονόματα (π.χ., **PK_Department_22AE2995**) για να αναπαρασταθεί το PRIMARY KEY(Dnumber).
- Το όνομα θα χρειαστεί εάν θέλουμε να αλλάξουμε ένα περιορισμό αργότερα με την χρήση της εντολής ALTER.
- **CREATE TABLE DEPARTMENT (…**

  Dnumber INT NOT NULL, ….

  CONSTRAINT PKDnumber PRIMARY KEY(Dnumber)

)

Περιορισμοί στην SQL (**Περιορισμός DEFAULT**)

- **DEFAULT <value>: Ορίζει ένα value για ένα γνώρισμα** δεδομένου ότι αυτό **δεν ορίζεται ρητά** κατά την εισαγωγή μιας πλειάδας σε μια σχέση.
- Παράδειγμα:

```
CREATE TABLE  LIBRARIAN                    /* or Micro_db.LIBRARIAN */
(    Name     nvarchar(50) NOT NULL,
     SSN      int                NOT NULL PRIMARY KEY,
     startdate DATE              NOT NULL      DEFAULT GETDATE(),
     Salary    DEC(8,2)          DEFAULT 0.0
);
```

- Πληροφορίες για TSQL
  - Το **value** μπορεί να είναι **NULL** ή **String** ή **scalar value**
  - Το **DEFAULT** δεν εφαρμόζεται πάνω σε IDENTITY, TIMESTAMP

Περιορισμοί στην SQL (**Περιορισμός CHECK**)

- **CHECK (logical expression): Περιορισμός** που επιβάλει την **ακεραιότητα οντότητας** περιορίζοντας τις δυνατές τιμές ενός γνωρίσματος βάσει λογικής συνθήκης.
  - Η συνθήκη εφαρμόζεται σε κάθε αλλαγή (insert, update)
  - ΔΕΝ επιτρέπεται αναφορά σε άλλο πίνακα μέσω της CHECK
- Παράδειγμα:

```
CREATE TABLE  LIBRARIAN (              /* or Micro_db.LIBRARIAN */
     Name     nvarchar(50) NOT NULL,
     SSN      int                NOT NULL PRIMARY KEY,
     Salary   DEC(8,2)     DEFAULT 0.0      CHECK(Salary >= 0)
);
```

- Πληροφορίες για TSQL

Δεν μπορεί να οριστεί για **text**, **ntext**, or **image**. Σημειώστε ότι αυτοί οι τύποι είναι επίσης deprecated. Για αυτό μπορεί να οριστούν ως **varchar(max), nvarchar(max) και varbinary(max) αντίστοιχα.**

Περιορισμοί στην SQL (**Περιορισμός CHECK**)

- Παραδείγματα Λειτουργιών της **CHECK** σε **TSQL**

Λειτουργία	Σύνταξη σε TSQL
Εξειδίκευση Πεδίου Ορισμού	Month **BETWEEN** 1 and 12
Εξειδίκευση Πεδίου Ορισμού	Shippers **IN** ('Ups', 'Fed Ex', 'USPS')
Σύγκριση Πεδίου με Σταθερά	Price **>=** 0
Σύγκριση Πεδίων	**DeliveryDate >= OrderDate**
Σύγκριση Πεδίου με Συνάρτηση	StartDate <= **GETDATE ( )**
Έλεγχος Μορφής (e.g., 111-11-1111) *'[ ]': Οποιοσδήποτε χαρακτήρας μέσα σε προσδιορισμένο **εύρος** (π.χ. [a-f]) ή **σύνολο** ([abcdef]).*	SSN **LIKE** '[0-9] [0-9] [0-9] -[0-9] [0-9] – [0-9] [0-9] [0-9] [0-9] '
Έλεγχος Μορφής *'%': **0** ή περισσότεροι χαρακτήρες* *'_': **1** οποιοσδήποτε χαρακτήρας.*	email LIKE '%@cs.ucy.ac.cy' '%@%.cy'

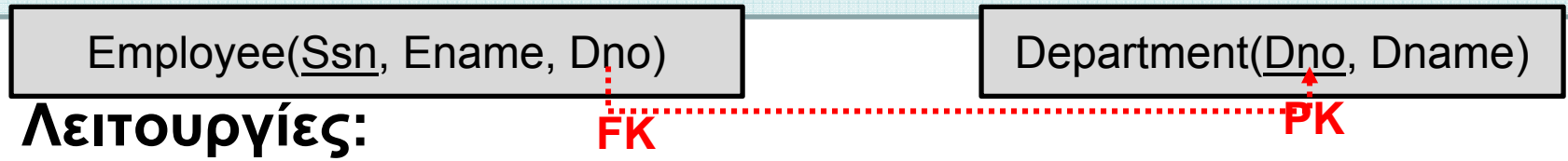
* Το LIKE θα μελετηθεί εκτενεστέρα στα πλαίσια της SQL-DML

Περιορισμοί στην SQL
(Εντολές Ενεργοποίησης Αναφοράς)

- Οι **CHECK**, **NOT NULL**, **DEFAULT** είναι περιορισμοί που χρησιμοποιούνται στο επίπεδο **ενός γνωρίσματος** ή μιας **σχέσης**.
- Η **FOREIGN KEY** από την άλλη αναφέρεται σε περιορισμούς μεταξύ δυο σχέσεων.
  - π.χ., Π.χ., FOREIGN KEY (MGRSSN) REFERENCES EMP(SSN)
- Εάν παραβιαστεί ο αναφορικός περιορισμός τότε η βάση απορρίπτει την πράξη (**REJECT**)
  - Π.χ., δεν μπορούμε να σβήσουμε τον Manager κάποιου υπαλλήλου στην βάση.
- Για να εξειδικεύσουμε την αντίδραση της βάσης σε τέτοιες τροποποιήσεις χρησιμοποιούνται κάποιες επιπλέον εντολές στην δήλωση ενός πίνακα.
  - Δες επόμενη διαφάνεια

Περιορισμοί στην SQL
(Εντολές Ενεργοποίησης Αναφοράς)

- Οι **Εντολές Ενεργοποίησης Αναφοράς** δηλώνουν *τι θα γίνει στο **FK (Foreign Key)** μιας σχέσης από αλλαγή στο **PK (Primary Key)** αναφερόμενης σχέσης.*
  - Ενεργοποιούνται αν παραβιαστεί ένας αναφορικός περιορισμός κατά το **ON DELETE** ή **ON UPDATE**



- **Λειτουργίες:**
  - **NO ACTION: Απαγορεύει την Αλλαγή** (Default).
    - Γιατί να ορίζεται; Για να ξεκαθαρίσει την πρόθεση του σχεδιαστή
  - **SET NULL: Θέτει το FK=NULL.**
    - Προϋποθέτει ότι η στήλη FK μπορεί να είναι NULLABLE.
  - **SET DEFAULT: Θέτει το FK=DEFAULT.**
    - Προϋποθέτει ότι το Default έχει οριστεί για το FK.
  - **CASCADE: Διάδοση Αλλαγής**
    - **ON DELETE CASCADE:** Διαγράφει την πλειάδα του FK
    - **ON UPDATE CASCADE:** Ενημερώνει το FK με το νέο PK

Περιορισμοί στην SQL
(Εντολές Ενεργοποίησης Αναφοράς)

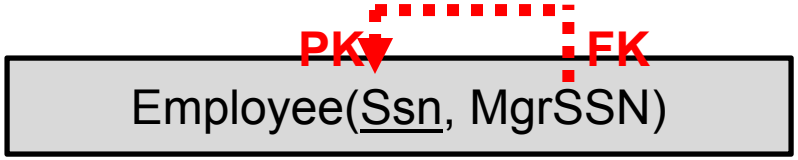
```
CREATE TABLE EMPLOYEE
      ( . . . ,
        Dno             INT          NOT NULL          DEFAULT 1,
      CONSTRAINT EMPPK
        PRIMARY KEY(Ssn),
      CONSTRAINT EMPSUPERFK
        FOREIGN KEY(Super_ssn) REFERENCES EMPLOYEE(Ssn)
                        ON DELETE SET NULL          ON UPDATE CASCADE,
      CONSTRAINT EMPDEPTFK
        FOREIGN KEY(Dno) REFERENCES DEPARTMENT(Dnumber)
                        ON DELETE SET DEFAULT       ON UPDATE CASCADE );
CREATE TABLE DEPARTMENT
      ( . . . ,
        Mgr_ssn         CHAR(9)      NOT NULL          DEFAULT '888665555',
        . . . ,
      CONSTRAINT DEPTPK
        PRIMARY KEY(Dnumber),
      CONSTRAINT DEPTSK
        UNIQUE(Dname),
      CONSTRAINT DEPTMGRFK
        FOREIGN KEY(Mgr_ssn) REFERENCES EMPLOYEE(Ssn)
                        ON DELETE SET DEFAULT       ON UPDATE CASCADE );
CREATE TABLE DEPT_LOCATIONS
      ( . . . ,
      PRIMARY KEY(Dnumber, Dlocation),
      FOREIGN KEY(Dnumber) REFERENCES DEPARTMENT(Dnumber)
                        ON DELETE CASCADE           ON UPDATE CASCADE );
```

Εάν σβηστεί ο Supervisor του υπαλλήλου X τότε X.Super_ssn=NULL

Εάν αλλάξει SSN ο Supervisor του υπαλλήλου X τότε X.Super_ssn=NewSSN

Περιορισμοί στην SQL
(Εντολές Ενεργοποίησης Αναφοράς)

- Οι Εντολές Ενεργοποίησης Αναφοράς, ειδικότερα το **ON DELETE CASCADE**, χρειάζονται μεγάλη προσοχή διότι η ενέργεια δύναται να προκαλέσει **αλυσίδωτες αλλαγές (Chaining)** που μπορεί να σβήσει ολόκληρο πίνακα!



- Π.χ., Θεωρήστε την δήλωση

```
CREATE TABLE Employee (
  SSN int NOT NULL PRIMARY KEY,   MgrSSN int,
  FOREIGN KEY(MgrSSN) REFERENCES Employee(SSN)
     ON DELETE CASCADE)
```

Και το στιγμιότυπο Employee = {(1,1), (2,1), (3,2), (4,3)}

- Σβήνοντας τον Employee.SSN=1 θα προκαλούσε διαγραφή όλων των εγγραφών του πίνακα!
- Για αυτό ο SQL Server δεν επιτρέπει την πιο πάνω εντολή CREATE TABLE επιστρέφοντας το μήνυμα λάθους"
  - Introducing FOREIGN KEY constraint 'FK__Employee__SSN__1CF15040' on table 'Employee' may **cause cycles or multiple cascade paths**. Specify ON DELETE NO ACTION or ON UPDATE NO ACTION, or modify other FOREIGN KEY constraints.

Περιορισμοί στην SQL
(**Προσωρινή Απενεργοποίηση Περιορισμών**)

- Γνωρίζουμε ότι η βάση δεδομένων είναι συνεχώς σε μια **συνεπή (consistent)** κατάσταση.
  - Δηλαδή όλοι οι περιορισμοί (κλειδιού, αναφορικής ακεραιότητας, κτλ) τηρούνται με ακρίβεια.
- Κάποτε επιθυμούμε να απενεργοποιήσουμε **προσωρινά** κάποιους περιορισμούς, π.χ.,
  - **Σειρά Εισαγωγής Δεδομένων:** εάν φορτώσουμε πρώτα τους Employee μετά τα Departments δεν θέλουμε να μας δίνει μηνύματα λάθους (λόγω foreign key constraint)
  - **Εισαγωγή Παλιών Δεδομένων,** θέλουμε να φορτώσουμε κάποια ΠΑΛΙΑ δεδομένα που δεν ακολουθούν κάποιο περιορισμό (π.χ., μορφοποίηση). Μπορούμε να τα φορτώσουμε και μετά να ενεργοποιήσουμε τον περιορισμό για τα νέα δεδομένα.

➔ **ΔΕΝ ΣΥΝΙΣΤΑΤΑΙ ΤΟ ΣΕΝΑΡΙΟ 2 ΩΣΤΟΣΟ (ΑΣΥΝΕΠΕΙΑ)**

Περιορισμοί στην SQL
(**Προσωρινή Απενεργοποίηση Περιορισμών**)

- **WITH CHECK | WITH NOCHECK:** Δηλώνει κατά πόσο τα δεδομένα ενός **πίνακα θα επικυρωθούν** (validated) ή όχι έναντι κάποιου περιορισμού (π.χ., περιορισμό FOREIGN KEY ή CHECK)

- **Επισημάνσεις**
  - Οι περιορισμοί πρέπει να απενεργοποιούνται **ΠΡΟΣΩΡΙΝΑ**.
  - **Δεν μπορούμε** να απενεργοποιήσουμε το **PRIMARY KEY** και **UNIQUE**
  - **Ενεργοποίηση/Απενεργοποίηση** θα γίνει με χρήση της εντολής **ALTER.**
- **Παραδείγματα**
  - **Ενεργ./Απενεργ. Υφιστάμενου Περιορισμού**:
    - ALTER TABLE Employee **CHECK | NOCHECK** CONSTRAINT fk_something
  - **Εισαγωγή Νέου Περιορισμού FK** (χωρίς έλεγχο παλιών data): **:**

    ALTER TABLE Employee **WITH NOCHECK**
    ADD CONSTRAINT FK_Employee **FOREIGN KEY(SSN) REFERENCES** Employee(SSN)
  - **Εισαγωγή Νέου Περιορισμού CHECK** (χωρίς έλεγχο παλιών data):

```
ALTER TABLE Employee
  WITH NOCHECK
  ADD CONSTRAINT CN_SSNFormat
  CHECK (SSN LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9]')
```

WITH CHECK είναι default για νέους περιορισμούς