Περιορισμοί Ακεραιότητας (Integrity Constraints)

- Αποτελούν μηχανισμό για τον έλεγχο της συνέπειας των δεδομένων.
- Χρησιμοποιούνται για να εξασφαλιστεί ότι μια βάση δεδομένων δεν θα βρεθεί ποτέ σε ασυνεπή κατάσταση.
- Περιορισμοί ακεραιότητας στο μοντέλο Οντοτήτων – Σχέσεων:
  - ορισμός κλειδιών: η δήλωση ενός πρωτεύοντος (ή υποψήφιου κλειδιού) για ένα σύνολο οντοτήτων περιορίζει τις αποδεκτές εισαγωγές και ενημερώσεις σε αυτές που δεν δημιουργούν οντότητες με ίδια τιμή στο κλειδί.
  - περιορισμοί πληθικότητας: περιορίζουν το σύνολο των αποδεκτών σχέσεων μεταξύ συνόλων οντοτήτων
- Εν γένει, οι περιορισμοί ακεραιότητας μπορούν να είναι αυθαίρετα κατηγορήματα που αναφέρονται σε μια βάση δεδομένων.

Περιορισμοί Ακεραιότητας (Integrity Constraints)

- Κατηγορήματα μεγάλης πολυπλοκότητας θα είναι δύσκολο να ελέγχονται.
- Περιοριζόμαστε σε τύπους περιορισμών οι οποίοι είναι εύκολο να ελεγχθούν μετά από κάθε αλλαγή που συμβαίνει στη βάση δεδομένων.
- Τύποι περιορισμών ακεραιότητας:
  1. Περιορισμοί Πεδίων Τιμών (Domain Constraints)
     - Στοιχειώδης τύπος περιορισμών ακεραιότητας
     - Χρησιμοποιείται για να ελεγχθεί αν τιμές γνωρισμάτων ανήκουν στα επιθυμητά πεδία τιμών

Περιορισμοί Ακεραιότητας (Integrity Constraints)

- Ορισμός περιορισμών πεδίων τιμών στην SQL:
  - Ορισμός πεδίου τιμών:
    e.g., **`create domain person-name char(20);`**
    ορίζει το **`person-name`** σαν ένα πεδίο τιμών αποτελούμενο από strings μήκους 20. Το **`person-name`** μπορεί ακόλουθα να χρησιμοποιηθεί σαν τύπος γνωρισμάτων.
  - Η κενή τιμή (**`null`**) ανήκει σε κάθε πεδίο τιμών.

Περιορισμοί Ακεραιότητας (Integrity Constraints)

- Παραδείγματα:

```
create domain hourly-wage numeric(5,2)
constraint wage-value-test
check(value>=4.00);

create domain account-number char(10)
constraint account-number-null-test
check(value not null);

create domain account-type char(10)
constraint account-type-test check(value in
(“Checking”, “Savings”));
```

Περιορισμοί Ακεραιότητας (Integrity Constraints)

2. Περιορισμοί Αναφοράς (Referential Integrity Constraints)

- Χρησιμοποιούνται για να διασφαλιστεί ότι τιμές γνωρισμάτων που εμφανίζονται σε μια σχέση εμφανίζονται και σε άλλες σχέσεις (όταν αυτές έχουν κοινά γνωρίσματα)
- Εκκρεμείς Πλειάδες (dangling tuples): πλειάδες σε σχέσεις στις οποίες μπορεί να εφαρμοστεί join οι οποίες όμως δεν συμμετέχουν στο αποτέλεσμα του join.

  Δηλαδή, αν για την πλειάδα $t_r \in r$, δεν υπάρχει πλειάδα $t_s \in s$ έτσι ώστε $t_r [r \cap s] = t_s [r \cap s]$, τότε η πλειάδα t_r λέγεται εκκρεμής.

Περιορισμοί Ακεραιότητας (Integrity Constraints)

- Παράδειγμα: έστω η σχέση account με σχήμα (account-number, branch-name, balance), η σχέση branch με σχήμα (branch-name, branch-city, assets). Έστω πλειάδα t στη σχέση account με t[branch name]="Atlantis". Αν στη σχέση branch δεν υπάρχει πλειάδα με αυτο το branch-name, τότε η πλειάδα t είναι εκκρεμής.
- Θέλομε να έχομε περιορισμούς οι οποίοι αποκλείουν τέτοιες καταστάσεις, ιδιαίτερα όταν το γνώρισμα στο οποίο εμφανίζονται «ανύπαρκτες» τιμές είναι ξένο κλειδί (foreign key).
- Ξένα κλειδιά: έστω r_1 και r_2 σχέσεις με κλειδιά K_1 και K_2 αντίστοιχα. Ένα υποσύνολο α του K_2 είναι ξένο κλειδί αναφερόμενο στο K_1, αν απαιτείται για κάθε πλειάδα t_2 στην r_2 να υπάρχει πλειάδα t_1 στην r_1 έτσι ώστε $t_1[K_1] = t_2[\alpha]$.

Περιορισμοί Ακεραιότητας (Integrity Constraints)

- Απαιτήσεις αυτής της μορφής ονομάζονται περιορισμοί αναφοράς.
- Περιορισμοί αναφοράς στο μοντέλο Οντοτήτων-Σχέσεων
  - Αν ένα σχεσιακό σχήμα προκύπτει από την παραγωγή πινάκων απο διαγράμματα Οντοτήτων-Σχέσεων, τότε κάθε πίνακας που αντιστοιχεί σε μια σχέση στο διάγραμμα έχει περιορισμούς αναφοράς:

    αν η σχέση R είναι βαθμού N μεταξύ των οντοτήτων E_1, E_2, ..., E_n και K_i είναι το πρωτεύον κλειδί της E_i, τότε το σχήμα για την R θα είναι $K_1 \cup K_2 \cup ... \cup K_n$ και κάθε K_i είναι ξένο κλειδί.
  - Το σχήμα κάθε σχέσης (πίνακα) που αναπαριστά μια ασθενή οντότητα περιλαμβάνει ως ξένο κλειδί το πρωτεύον κλειδί της αντίστοιχης ισχυρής οντότητας.

Περιορισμοί Ακεραιότητας (Integrity Constraints)

- Περιορισμοί αναφοράς και μεταβολή της ΒΔ.
  - Μεταβολές στην ΒΔ μπορούν να προκαλέσουν την παραβίαση περιορισμών αναφοράς.
  - Για κάθε είδος μεταβολής, υπάρχει μια συνθήκη η οποία πρέπει να ελεγχθεί για να διασφαλιστεί ότι δεν παραβιάζεται κανένας περιορισμός.
  - Έστω ότι ο περιορισμός αναφοράς περιγράφεται από την ακόλουθη συνθήκη:

$$\pi_{\alpha}(r_2) \subseteq \pi_{K}(r_1)$$

Περιορισμοί Ακεραιότητας (Integrity Constraints)

$$\pi_{\alpha}(r_2) \subseteq \pi_{K}(r_1)$$

a) Εισαγωγή: αν η πλειάδα t_2 εισαχθεί στη σχέση r_2, πρέπει να ελεγχθεί ότι υπάρχει πλειάδα t_1 στην r_1 έτσι ώστε $t_1[K] = t_2[\alpha]$. Άρα η συνθήκη που πρέπει να ελεγχθεί είναι: $t_2[\alpha] \in \pi_{K}(r_1)$

b) Διαγραφή: αν η πλειάδα t_1 διαγραφεί από την r_1 πρέπει να βρεθεί το σύνολο των πλειάδων της r_2 που αναφέρονται στην t_1: $\sigma_{\alpha = t1[K]}(r_2)$. Αν αυτό το σύνολο δεν είναι κενό, τότε είτε η διαγραφή δεν θα εκτελεστεί, είτε οι πλειάδες που αναφέρονται στην t_1 πρέπει να διαγραφούν επίσης.

   Η δεύτερη επιλογή μπορεί να οδηγήσει σε συνακόλουθες (cascaded) διαγραφές καθώς πλειάδες άλλων σχέσεων μπορεί να αναφέρονται σε πλειάδες που αναφέρονται στην t_1 κ.ο.κ.

Περιορισμοί Ακεραιότητας (Integrity Constraints)

$$\pi_{\alpha}(r_2) \subseteq \pi_K(r_1)$$

c) Ενημέρωση: υπάρχουν δύο περιπτώσεις

I. Ενημέρωση στην r_2: αν μια πλειάδα t_2 της r_2 ενημερώνεται και η ενημέρωση μεταβάλλει την τιμή του ξένου κλειδιού α, τότε αν t_2' είναι η νέα πλειάδα, πρέπει να ελεγχθεί αν $t_2'[\alpha] \in \pi_K(r_1)$

II. Ενημέρωση στην r_1: αν μια πλειάδα t_1 της r_1 ενημερώνεται και η ενημέρωση μεταβάλλει την τιμή του πρωτεύοντος κλειδιού K, τότε πρέπει να βρεθεί το σύνολο $\sigma_{\alpha= t1[K]}(r_2)$. Αν το σύνολο αυτό είναι μη-κενό, τότε είτε η ενημέρωση δεν εκτελείται είτε η ενημέρωση γίνεται με συνακόλουθες ενημερώσεις (όπως στην περίπτωση της διαγραφής).

Περιορισμοί Ακεραιότητας (Integrity Constraints)

- Ορισμός περιορισμών αναφοράς στην SQL:

```
create table customer (customer-name char(20)
not null, customer-street char(30), customer-
city char(30),
primary key (customer-name));

create table branch (branch-name char(15) not
null, branch-city char(30), assets integer,
primary key (branch-name),
check (assets>=0));
```

Περιορισμοί Ακεραιότητας (Integrity Constraints)

```
create table account (account-number char(10)
not null, branch-name char(15), balance integer,
primary key (account-number),
foreign key (branch-name) references branch,
check(balance >=0));

create table depositor (customer-name char(20)
not null, account-number char(10) not null,
primary key (customer-name, account-number),
foreign key (customer-name references customer,
foreign key (account-number) references
account);
```

Περιορισμοί Ακεραιότητας (Integrity Constraints)

- Καθορίζεται επίσης το πως αντιμετωπίζεται μια παραβίαση περιορισμού:

```
create table account (account-number char(10)
not null, branch-name char(15), balance integer,
primary key (account-number,
foreign key (branch-name) references branch,
on delete cascade,
on update cascade,
check(balance >=0));
```

Διαγραφή μιας πλειάδας από τη σχέση branch ακολουθείται από τη διαγραφή των πλειάδων της σχέσης account που την αναφέρουν. Παρόμοια, στην περίπτωση της ενημέρωσης, ενημερώνονται τα αντίστοιχα γνωρίσματα της σχέσης account.

Περιορισμοί Ακεραιότητας (Integrity Constraints)

- Άλλες ενέργειες για την αντιμετώπιση της παραβίασης περιορισμών περιλαμβάνουν την εισαγωγή κενής τιμής στο πεδίο το οποίο αναφέρεται σε πεδίο πλειάδας άλλης σχέσης η οποία διαγράφεται ή ενημερώνεται.
- Αν η παραβίαση ενός περιορισμού δεν μπορεί να διορθωθεί με διαγραφές / ενημερώσεις , τότε η πράξη που προκαλεί την παραβίαση διακόπτεται και αναιρούνται οι πράξεις που εκτελέστηκαν.
- Χειρισμός κενών τιμών:
  - Όλα τα γνωρίσματα του πρωτεύοντος κλειδιού θεωρούνται **`not null`**
  - Γνωρίσματα ξένων κλειδιών μπορούν να δέχονται κενές τιμές. Οι κενές τιμές δεν παραβιάζουν τους περιορισμούς αναφοράς.

Περιορισμοί Ακεραιότητας (Integrity Constraints)

- Δηλώσεις (assertions)
  - Κατηγορήματα τα οποία εκφράζουν συνθήκες οι οποίες πρέπει να ικανοποιούνται σε κάθε στιγμιότυπο μιας ΒΔ.
  - Οι περιορισμοί πεδίου και αναφοράς είναι ειδικές περιπτώσεις τέτοιων κατηγορημάτων.
  - Η SQL παρέχει την εντολή **`create assertion.`**
  - Παράδειγμα: the sum of all loan amounts for each branch must be less than the sum of all account balances at the branch

```
create assertion sum-constraint
check (not exists (select * from branch where
(select sum(amount) from loan where loan.branch-
name = branch.branch-name) >=
(select sum(balance) from account where
account.branch-name = branch.branch-name)));
```

Περιορισμοί Ακεραιότητας (Integrity Constraints)

- Δηλώσεις (assertions)
  - Όταν δημιουργείται μια δήλωση, ελέγχεται αν ισχύει. Αν ναι, τότε μεταβολές στη βάση δεδομένων επιτρέπονται μόνο αν δεν παραβιάζουν τη δήλωση.
  - Ο έλεγχος δηλώσεων έχει μεγάλο κόστος. Τα ΣΔΒΔ συνήθως δεν βελτιστοποιούν τον έλεγχο δηλώσεων.
- Ενεργοί κανόνες (active rules, triggers)
  - Ενεργοί κανόνες εκτελούνται αυτόματα σαν αποτέλεσμα μια μεταβολής στη βάση δεδομένων.
  - Ο ορισμός των κανόνων περιλαμβάνει τον προσδιορισμό των συνθηκών κάτω από τις οποίες ο κανόνας θα εκτελείται και τις ενέργειες που θα εκτελεστούν.

Περιορισμοί Ακεραιότητας (Integrity Constraints)

- Παράδειγμα: αν κάποιος αποσύρει περισσότερα χρήματα από όσα έχει στο λογαριασμό του, αυτόματα δημιουγείται ένα δάνειο με το ποσό που υπερβαίνει το υπόλοιπο του λογαριασμού του.

```
define trigger overdraft on update of account T
(if new T.balance < 0 then
(insert into loan values
   (T.branch-name, T.account-no, -new T.balance)
(insert into borrower
   (select customer-name, account-no from depositor
   where T.account-no = depositor.account-no)
update account S
set S.balance=0 where S.account-no=T.account-no));
```

  - Η λέξη **new** συμβολίζει την ενημερωμένη πλειάδα.

Σχεδίαση Β.Δ. (Database Design)

- Η σχεδίαση ενός σχήματος μιας Β.Δ. βασίζεται σε μεγάλο βαθμό στη διαίσθηση του σχεδιαστή σχετικά με τον κόσμο που θέλει να αναπαραστήσει.
- Η εννοιολογική σχεδίαση υπαρκτών κόσμων παράγει σχήματα υψηλού επιπέδου, π.χ. διαγράμματα Οντοτήτων – Σχέσεων.
- Ένα εννοιολογικό μοντέλο πρέπει να μετατραπεί σε ένα λογικό μοντέλο, π.χ. σχεσιακό.
- Προκειμένου να κρίνομε αν η σχεδίαση είναι ορθή πρέπει να χρησιμοποιήσομε κάποια τυπικά κριτήρια.

Θεωρία Σχεδίασης Σχεσιακών Β.Δ.

- Η σχεδίαση του σχήματος μιας σχεσιακής Β.Δ. μπορεί να τυποποιηθεί με χρήση της θεωρίας κανονικοποίησης (normalization theory)
- Με τον όρο «κανονικοποίηση» εννοούμε την εφαρμογή κανόνων σχεδίασης που αποκαλούνται κανονικές μορφές (normal forms) και οι οποίοι περιορίζουν τις δυνατές μορφές σχεσιακών σχημάτων.
- Αν ακολουθούνται οι κανόνες αυτοί αποφεύγεται ανώμαλη ή λανθασμένη συμπεριφορά του συστήματος.
- Επιβεβαιώνεται επίσης ότι το σχήμα έχει διάφορες επιθυμητές ιδιότητες.

Κανονικές Μορφές

- 1η Κανονική Μορφή (1NF)
  - Μια σχέση είναι σε 1η κανονική μορφή αν δεν έχει πλειότιμα γνωρίσματα
- 2η Κανονική Μορφή
- 3η Κανονική Μορφή
- Κανονική Μορφή Boyce-Codd

- Υπόθεση: όλες οι σχέσεις είναι σε 1NF

Κανονικές Μορφές

- Παράδειγμα: Employee Database
  - για κάθε υπάλληλο αποθηκεύεται η ακόλουθη πληροφορία:

    `emp_id, emp_name, emp_phone, dept_name, dept_phone, dept_mgrname, skill_id, skill_name, skill_date, skill_lvl`
  - Τα γνωρίσματα **`emp_id, dept_name`** και **`skill_id`** προσδιορίζουν μοναδικά υπαλλήλους, τμήματα και δεξιότητες αντίστοιχα.
  - Μια καθολική σχέση (universal relation) είναι μια σχέση η οποία περιέχει όλα τα γνωρίσματα που αντιστοιχούν σε πληροφορία σχετικά με τους υπαλλήλους.

Κανονικές Μορφές

- Προβλήματα με τη χρήση καθολικής σχέσης:
  - Επανάληψη πληροφορίας: για έναν υπάλληλο με πολλές δεξιότητες εμφανίζονται ισάριθμες πλειάδες στη σχέση
  - Αν ένας υπάλληλος αλλάξει τμήμα ή αριθμό τηλεφώνου, όλες οι πλειάδες του υπαλλήλου στη σχέση πρέπει να αλλαχθούν.
  - Αντίστοιχα, αν ένα τμήμα αποκτήσει καινούργιο διευθυντή.
  - Η επαναλαμβανόμενη πληροφορία δεν είναι μόνο περιττή, αλλά πρέπει επίσης να κρατείται ενημερωμένη
  - Αν ένας υπάλληλος ο οποίος έχει μια και μόνο δεξιότητα, την χάσει, τότε όλη η πληροφορία σχετικά με τον υπάλληλο πρέπει να χαθεί επίσης.

Προβλήματα Κακής Σχεδίασης

- Πρόβλημα ενημέρωσης:
  - Μια σχέση R πάσχει από πρόβλημα ενημέρωσης αν, όποτε αλλάζει η τιμή ενός γνωρίσματος για ένα στιγμιότυπο της οντότητας ή της σχέσης την οποία αναπαριστά η R, είναι απαραίτητη η ενημέρωση πολλαπλών πλειάδων της R.
- Πρόβλημα διαγραφής:
  - Μια σχέση R πάσχει από πρόβλημα διαγραφής αν η διαγραφή μιας πλειάδας της σχέσης έχει ως αποτέλεσμα την απώλεια πληροφορίας σχετικά με μια σχετιζόμενη οντότητα ή σχέση.
- Πρόβλημα εισαγωγής:
  - Μια σχέση R πάσχει από πρόβλημα εισαγωγής αν πληροφορία δεν μπορεί να αναπαρασταθεί παρά μόνο αν περιληφθεί πληροφορία σχετική με κάποια άλλη οντότητα ή σχέση (η οποία μπορεί να μην υπάρχει).

Προβλήματα Κακής Σχεδίασης

- Στο προηγούμενο παράδειγμα, κάποια από τα προβλήματα αυτά μπορούν να λυθούν αν η καθολική σχέση χωριστεί σε δύο πίνακες:

```
Employees (emp_id, emp_phone, dept_name,
                    dept_phone, dept_mgrname)
Skills (emp_id, skill_id, skill_name,
                          skill_date, skill_lvl)
```

- Ο πίνακας **`Employees`** περιέχει μια μοναδική πλειάδα για κάθε υπάλληλο.
- Ο πίνακας **`Skills`** περιέχει μια μοναδική πλειάδα για κάθε ζεύγος **`emp_id, skill_id.`**
- Η συνένωση των δύο σχέσεων δίνει την αρχική σχέση.
- Άρα έχομε την ίδια πληροφορία χωρίς περιττή επανάληψη.