Database Programming with SQL
kurs 2017 – database design and programming with sql
students slajdovi

14-1 Intro to Constraints; NOT NULL and UNIQUE Constraints

- Ograničenja (constraints) se koriste za sprečavanje unosa pogrešnih podataka u tabele

Constraints in General

- Ograničenja su zapravo db pravila
- Sve definicije ograničenja s enalaze smeštena u DD
- Ograničenja sprečavaju brisanje tabele ako postoje zavisnosti od drugih tabela
- Ograničenja forsiraju pravila na podacima gde god je red unet, updejtovan ili obrisan iz tabele
- Pet tipova ograničenja su: NOT NULL, UNIQUE, PRIMERY KEY, FOREIGN KEY, CHECK
- Ograničenja su važna kao i njihovo imenovanje

Creating Constraints

- U CREATE TABLE iskazu na sledećem primeru, svaka kolona i njeni podaci su definisani
- Koristi se CREATE TABLE iskaz za uspostavljanje ograničenja za svaku kolonu u tabeli
- Postoje dva različita mesta u CREATE TABLE iskazu gde se specificiraju detalji ograničenja: na nivou kolone pored imena i tipa podataka, na nivou tabele posle svih imena izlistanih kolona

```
CREATE TABLE clients
(client_number NUMBER(4),
 first_name        VARCHAR2(14),
 last_name        VARCHAR2(13));
```

- Primetiti da nivo kolone samo se odnosi na prostor u CREATE TABLE iskazu gde su definisane kolone
- Nivo tabele se odnosi na poslednju liniju u iskazu ispod liste pojedinačnih imena kolona

Constraints at the Column Level

- Ograničenja na nivou kolone referenciraju jednu kolonu
- Za uspostavljanje ograničenja na nivou kolone, ograničenje mora biti definisano u CREATE TABLE iskazu kao deo definicije kolone
- Ispitati sledeći SQL iskaz koji uspostavlja ograničenje na nivou kolone:

```
CREATE TABLE clients
(client_number NUMBER(4) CONSTRAINT clients_client_num_pk PRIMARY KEY,
 first_name        VARCHAR2(14),
 last_name        VARCHAR2(13));
```

- CREATE TABLE klijente:

```
CREATE TABLE clients
(client_number NUMBER(4) CONSTRAINT clients_client_num_pk PRIMARY KEY,
 first_name        VARCHAR2(14),
 last_name        VARCHAR2(13));
```

- Ime ograničenja je clients_client_num_pk.
- Ono inforsira biznis pravila po kojem client_number je PK u clients tabeli

Naming Constraints

- Svako ograničenje u db ima ime. Kada ograničenje se napravi, može mu se dati ime, poput clients_client_num_pk, ili ostaviti bez imena, u kojem slučaju sistem daje ime kao SYS_C00585417
- Konvencija davanja imena može biti kombinacija abervacija imena tabele i abervacija imena kolone koju prati abervacija ograničenja: table-name_column-name_constraint-type
- Ako je rezervisna reč CONSTRAINT korišćena u CREATE TABLE definiciji, mora se dati ime ograničenju. Imena ograničenja su limitirana na 30 karaktera

Naming Constraints at the Column Level

- Najbolje je imenovati ograničenja samostalno pošto sistemski imenovana ograničenja se teško raspoznaju
- Pogledati sledeću definiciju tabele:

```
CREATE TABLE clients
(client_number    NUMBER(4),
 last_name        VARCHAR2(13),
 email            VARCHAR2(80));
```

- Prema našoj konvenciji imenovanja:
  - PK ograničenje na client_number će biti imenovano clients_client_number_pk
  - Not null ograničenje na last_name će biti imenovano clients_last_name_nn
  - Jedinstveno ograničenje na email adresi će biti imenovano clients_email_uk

```
CREATE TABLE clients
(client_number NUMBER(4) CONSTRAINT clients_cient_num_pk PRIMARY KEY,
 last_name     VARCHAR2(13) CONSTRAINT clients_last_name_nn NOT NULL,
 email         VARCHAR2(80) CONSTRAINT clients_emil_uk UNIQUE);
```

Constraint Naming example

- Ovaj primer pokazuje i user-named ograničenja i system-named ograničenja:

```
CREATE TABLE clients
 (client_number NUMBER(4) CONSTRAINT clients_client_num_pk PRIMARY KEY,
  last_name     VARCHAR2(13) NOT NULL,
  email         VARCHAR2(80));
```

- System-named ograničenja:

```
CREATE TABLE clients
 (client_number NUMBER(4) CONSTRAINT clients_client_num_pk PRIMARY KEY,
  last_name     VARCHAR2(13) NOT NULL,
  email         VARCHAR2(80));
```

- Dva ograničenja su kreirana:
  - user-named igraničenja clients_client_num_pk za enforsiranje pravila kojim client_number je PK
  - system_named ograničenje imenom SYS_Cn (gde je n jedinstven integer) za enforsiranje pravila kojim last_names ne mogu biti null

Constraints at the Table level

- Tabela-nivo ograničenja su izlistana odvojeno od definicija kolona u CREATE TABLE iskazu
- Ograničenja tabela-nivoa definicije su izlistane posle definicija svih kolona tabele

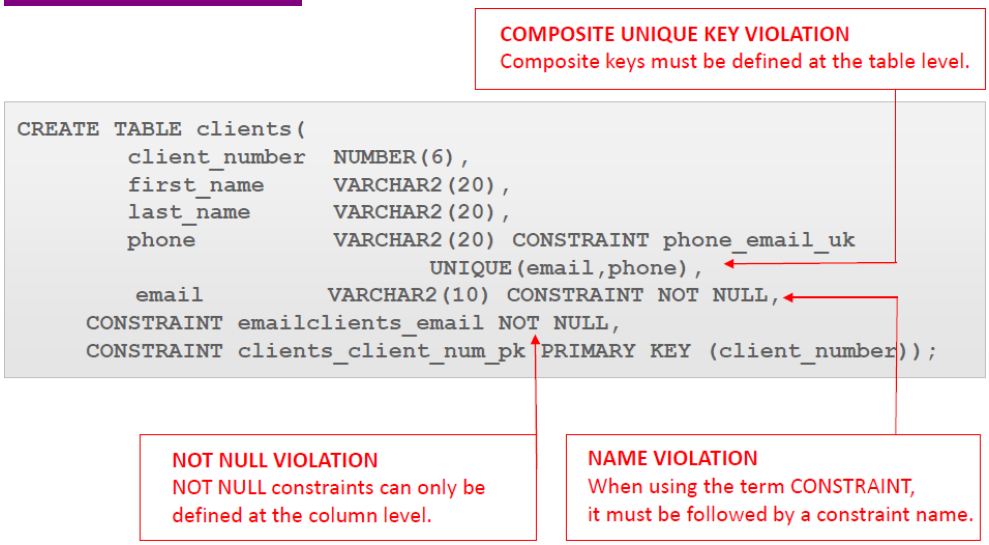
- U sledećem primeru, jedinstveno ograničenje je izlistano poslednje u CREATE TABLE iskazu

```
CREATE TABLE clients (
 client_number NUMBER(6) NOT NULL,
 first_name    VARCHAR2(20),
 last_name     VARCHAR2(20),
 phone         VARCHAR2(20),
 email         VARCHAR2(10) NOT NULL,
CONSTRAINT clients_phone_email_uk UNIQUE (email,phone));
```

Basic Rules for Constraints

- Ograničenja koja se referišu na više od jedne kolone (kompozitni ključ) morajubiti definisana na nivou tabele
- NOT NULL ograničenja se mogu specifirati samo na nivou kolone, ne na nivou tabele
- UNIQUE, PK, FK i CHECK ograničenja se mogu definisati ili na nivou kolone ili na nivou tabele
- Ako se koristi reč CONSTRAINT u CREATE TABLE iskazu, mora se dati ime za ograničenje
- Ograničenje koje se referiše na dve kolone ne može biti definisano kao deo definicije kolone. Koje od dve definicije kolone će biti postavljeno pored ? Ne mogu oba.
- Nivo tabele ograničenja mora biti user-named, pošto nivo tabele ograničenja mora uključiti službenu reč CONSTRAINT
- NOT NULL ograničenja se ne mogu definisati na nivou tabele pošto ANSI/ISO SQL standard to zabranjuje

Examine the Violations



Five Types of Constraints

- Pet tipova ograničenja postoji unutar Oracle db i svaki tip enforsira različito pravilo
- Tipovi ograničenja su: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK

NOT NULL Constraint

- Kolona definisana sa NOT NULL ograničenjem zahteva da za svaki red unešen u tabeli, vrednost mora postojati za tu kolonu
- Npr, ako email kolona u tabeli employees je bila definisana kao NOT NULL, svaki zaposleni unešen u tabelu mora imati vrednost u email koloni
- Pri definisanju NOT NULL kolona, obično se koristi sufiks _nn u imenu ograničenja

- Npr, ime ograničenja za NOT NULL email kolonu u tabeli zaposleni može biti emp_email_nn

UNIQUE Constraint

- UNIQUE ograničenje zahteva da svaka vrednost u koloni ili setu kolona (kompozitni ključ) bude jedinstven; tj nijednom dva reda u tabeli ne smeju imati duplicirane vrednosti
- Npr, može biti važno za biznis da se osigura da nijednom dva čoveka imaju istu email adresu
- Kolona email može biti definisana korišćenjem UNIQUE ograničenja
- Kolona ili set kolona koja je definisana kao UNIQUE se naziva unique key
- Ako kombinacija dve ili više kolona mora biti jedinstvena za svaki ulaz, ograničenje je kompozitni jedinstveni ključ
- Govoreći da sve kombinacije od emila i prezimena moraju biti UNIQUE je jedan primer kompozitnog jedinstvenog ključa
- Reč "key" se odnosi na kolone, ne na nazive ograničenja

Unique Constraint Example

- Ako emil kolona u tabeli je definisana sa UNIQUE ograničenjem, nijedan drugi klijentski unos ne sme imati identičan email
- Šta ako dva klijenta žive u isoj kući i dele email adresu ?

CLIENT_NUMBER	FIRST_NAME	LAST_NAME	PHONE	EMAIL
5922	Hiram	Peters	3715832249	hpeters@yahoo.com
5857	Serena	Jones	7035335900	serena.jones@jones.com
6133	Lauren	Vigil	4072220090	lbv@lbv.net

```
INSERT INTO clients (client_number, first_name, Last_name, phone,
email)
   VALUES (7234, 'Lonny', 'Vigil', 4072220091, 'lbv@lbv.net');
```

```
ORA-00001: unique constraint
(USWA_SKHS_SQL01_T01.CLIENT_EMAIL_UK) violated
```

- Naziv ograničenja koje je prekoračeno je prikazano kao poruka o grešci, koja govori korisniku gde je problem. Ako ograničenje nije imenovano, sistem generiše ime koji će se prikazati, što je teže za korisnika da razreši problem

Defining UNIQUE Constraints

- Pri definisanju UNIQUE ograničenja, često se koristi sufiks _uk u imenu ograničenja
- Npr, ime ograničenje za UNIQUE email kolonu u empoloyees tabeli može biti emp_email_uk
- Za definisanje kompozitnog jedinstvenog ključa, mora se definisati ograničenje na nivou tabele pre nego na nivou kolone
- Primer kompozitnog jedinstvenog ključa imena ograničenja je:

CONSTRAINT clients_phone_email_uk UNIQUE(email, phone)

Coimposite Unique Key

- UNIQUE ograničenja omogućavaju unos nulls osim ako kolona takođe ima NOT NULL definisano ograničenje

- Null u koloni (ili u svim kolonama kompozitnog jedinstvenog ključa) uvek zadovoljava UNIQUE ograničenje pošto nulls se ne smatraju identične sa bilo čime

CLIENT_NUMBER	FIRST_NAME	LAST_NAME	PHONE	EMAIL
5922	Hiram	Peters	3715832249	hpeters@yahoo.com
5857	Serena	Jones	7035335900	serena.jones@jones.com
6133	Lauren	Vigil	4072220090	lbv@lbv.net
7234	Lonny	Vigil	4072220091	lbv@lbv.net

This combination of columns must be **UNIQUE**

- Da bi se zadovoljilo ograničenje koje designates kompozitni jedinstveni ključ, nijedna dva reda u tabeli ne smeju imati istu kombinaciju vrednosti u kolonama ključa
- Takođe, bilo koji red koji sadrži nulls u svim kolonama ključa automatski zadovoljava ograničenje

Constraints Created at Table Creation

- Kada se doda NOT NULL ograničenje kao deo tabele creation statement, Oracle db će kreirati Check Constraint u db za enforsiranje vrednosti u NOT NULL koloni
- Ovo kreirano ograničenje može biti skoro nevidljivo za programera kada se kreira tabela – Oracle samo to uradi
- Na kraju table creation statement, poruka “Table created“ se pokaže, ali bez detalja o broju ili tipovima ograničenja koja su takođe kreirana

14-2 PRIMARY KEY, FOREIGN KEY and CHECK Constraints

- Pošto se može desiti preko nekakve greške da jedna jedinstvena identifikacija se doda drugoj osobi, to bi prekršilo integritet ograničenja

PRIMARY KEY Constraints

- PK ograničenje je pravilo po kojem vrednosti u jednoj koloni ili kombinacije kolona moraju jedinstveno identifikovati svaki red u tabeli
- Nijedna PK vrdenost se može pojaviti u više od jednog reda tabele
- Za zadovoljenje PK ograničenja, oba sledeća uslova moraju biti tačna: nijedna kolona koja je deo PK sme sadržavati null; tabela može imati samo jedan PK
- Iako tabela može imati samo jedan PK, on može biti napravljen od više kolona i to se naziva kompozitni PK
- PK ograničenja se mogu definisati u koloni na nivou tabele
- Ipak, ako postoji kompozitni PK, on mora biti definisan na nivou tabele
- Pri definisanju PK kolona, dobro je koristiti sufix _pk u imenu ograničenja
- Npr, ime ograničenja za PK kolonu imenom client_number u tabeli CLIENTS će biti clients_client_num_pk
- U CREATE TABLE iskazu, kolona nivo PK ograničenja ima sintaksu:

```
CREATE TABLE clients
(client_number NUMBER(4) CONSTRAINT clients_client_num_pk PRIMARY KEY,
first_name VARCHAR2(14),
last_name VARCHAR2(13));
```

- Primetiti da kolona-nivo samo se odnosi na deo CREATE TABLE iskaza gde su definisane kolone

- Nivo tabele se odnosi na poslednju liniju u iskazu ispod liste individualnih imena kolona
- Da bi se napravio PK ograničenje na nivou tabele:

```
CREATE TABLE clients
(client_number NUMBER(4),
 first_name VARCHAR2(14),
 last_name VARCHAR2(13),
CONSTRAINT clients_client_num_pk PRIMARY KEY (client_number));
```

- Primetiti da PK ime kolone prati tip ograničenja i zatvoreno je u zagradama
- Za definisanje kompozitnog PK, mora se definisati ograničenje na nivou tabele pre nego na nivou kolone
- Primer kompozitnog PK ograničenja:

```
CREATE TABLE copy_job_history
(employee_id NUMBER(6,0),
 start_date DATE,
 job_id VARCHAR2(10),
 department_id NUMBER(4,0),
CONSTRAINT copy_jhist_id_st_date_pk PRIMARY KEY(employee_id, start_date));
```

FOREIGN KEY (REFERENTAL INTEGRITY) Constraints

- FK ograničenja se takođe nazivaju “referntial integrity“ ograničenja
- FK ograničenja prekidaju označavanje (designate) kolone ili kombinacije kolona kao FK
- FK upućuju nazad na PK (ili jedinstveni ključ) u drugoj tabeli, i ovaj link je osnova relacija između tabela

Viewing a Foreign Key

- Tabela koja sadrži FK se naziva “child“ (dete) tabela a tabela koja sadrži referencirani ključ se naziva “parent“ (roditelj) tabela

DEPARTMENTS - Parent

DEPARTMENT_ID	DEPT_NAME	MANAGER_ID	LOCATION_ID
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting	-	1700

EMPLOYEE - Child

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID
100	Steven	King	90
101	Neena	Kochhar	90
102	Lex	De Haan	90
205	Shelley	Higgins	110
206	William	Gietz	110

- U prikazanimtabelama, PK od DEPARTMENTS tabele, department_id, se takođe pojavljuje u EMPLOYEES tabeli kao FK kolona

DEPARTMENTS - Parent

DEPARTMENT_ID	DEPT_NAME	MANAGER_ID	LOCATION_ID
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting	-	1700

EMPLOYEE - Child

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID
100	Steven	King	90
101	Neena	Kochhar	90
102	Lex	De Haan	90
205	Shelley	Higgins	110
206	William	Gietz	110

Referential-integrity Constraint

- Da bi zadovoljio referential integrity ograničenje, FK vrednost mora odgovarati postojećoj vrednosti u roditelj tabeli ili da bude NULL

DEPARTMENTS - Parent

DEPARTMENT_ID	DEPT_NAME	MANAGER_ID	LOCATION_ID
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting	-	1700

EMPLOYEE - Child

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID
100	Steven	King	90
101	Neena	Kochhar	90
102	Lex	De Haan	90
205	Shelley	Higgins	110
206	William	Gietz	110

- PK vrednost može postojati bez odgovarajućeg FK vrednosti; ipak FK mora imati odgovarajući PK

Referential-Integrity Constraint Rule

- Pravilo je: pre nego definišete referential integrity ograničenje u dete tabeli, referenciran UNIQUE ili PK ograničenje na roditelj tabeli mora već biti definisano
- Drugim rečima, prvo se mora imati roditelj PK definisan pre nego se napravi FK u dete tabeli

FOREIGN KEY Constraint

- Za definisanje FK ograničenja, dobra je praksa koristiti sufiks _fk u imenu ograničenja
- Npr, ime ograničenja za FK kolonu department_id u employees tabeli može biti imenovano emps_dept_id_fk

FOREIGN KEY Constraint Syntax

- Sintaksa za definisanje FK ograničenja zahteva referencu na tabelu i kolonu u roditelj tabeli
- FK ograničenje u CREATE TABLE iskazu na nivou kolone se može definisati kao:

```
CREATE TABLE copy_employees
(employee_id NUMBER(6,0) CONSTRAINT copy_emp_pk PRIMARY KEY,
 first_name VARCHAR2(20),
 last_name VARCHAR2(25),
 department_id NUMBER(4,0) CONSTRAINT c_emps_dept_id_fk
                           REFERENCES departments(department_id),
 email VARCHAR2(25));
```

- Prilikom definisanja FK na nivou kolone, reči “foreign key“ nisu uključena
- Sintaksa za definisanje FK ograničenja zahteva referenciranje na tabelu i kolonu u roditelj tabeli
- FK ograničenje u CREATE TABLE iskazu na nivou tabele može biti definisano na sledeći način:

```
CREATE TABLE copy_employees
(employee_id NUMBER(6,0) CONSTRAINT copy_emp_pk PRIMARY KEY,
 first_name VARCHAR2(20),
 last_name VARCHAR2(25),
 department_id NUMBER(4,0),
 email VARCHAR2(25),
 CONSTRAINT c_emps_dept_id_fk FOREIGN KEY (department_id)
                           REFERENCES departments(department_id));
```

- Prilikom definisanja FK na nivou kolone, reči “foreign key“ jesu uključene pre imena tabele i kolone koje ima to ograničenje

ON DELETE CASCADE – Maintaining Referential Integrity

- Korišćenje ON DELETE CASCADE opcije pri definisanju FK omogućava zavisne redove u dete tabeli da budu obrisani kada red u roditelj tabeli je obrisan
- Ako FK nema ON DELETE CASCADE opciju, referencirani redovi u roditelj tabeli ne mogu biti obrisani
- Drugim rečima, dete tabela FK ograničenje uključuje ON DELETE CASCADE dozvolu koja omoguzćava njenim roditeljima da obrišu redove ne koj se referišu

DEPARTMENTS - Parent

DEPARTMENT_ID	DEPT_NAME	MANAGER_ID	LOCATION_ID
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting	-	1700

EMPLOYEE - Child

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID
100	Steven	King	90
101	Neena	Kochhar	90
102	Lex	De Haan	90
205	Shelley	Higgins	110
206	William	Gietz	110

ON DELETE CASCADE

- Ako department_id kolona u employees je kreirana sa ON DELETE CASCADE opcijom specificiranom, DELETE iskaz izveden na tabeli departments će biti izveden
- Ako ON DELETE CASCADE opcija nije bila specificirana kada FK je kreiran, pokušaj brisanja sektora iz tabele departments koji ima unos u employees tabeli će pasti
- Tabela kreirana bez ON DELETE CASCADE:

```
CREATE TABLE copy_employees
(employee_id NUMBER(6,0) CONSTRAINT copy_emp_pk PRIMARY KEY,
 first_name VARCHAR2(20),
 last_name VARCHAR2(25),
 department_id NUMBER(4,0),
 email VARCHAR2(25),
 CONSTRAINT cdept_dept_id_fk FOREIGN KEY (department_id)
        REFERENCES copy_departments(department_id));
```

- Pokušaj brisanja department_id 110 iz tabele departments neće uspeti pošto postoje zavisni redovi u tabeli employees

```
ORA-02292: integrity constraint (US_A009EMEA815_PLSQL_T01.CDEPT_DEPT_ID_FK)
violated - child record found
```

- Tabela kreirana sa ON DELETE CASCADE:

```
CREATE TABLE copy_employees
(employee_id NUMBER(6,0) CONSTRAINT copy_emp_pk PRIMARY KEY,
 first_name VARCHAR2(20),
 last_name VARCHAR2(25),
 department_id NUMBER(4,0),
 email VARCHAR2(25),
 CONSTRAINT cdept_dept_id_fk FOREIGN KEY (department_id)
    REFERENCES copy_departments(department_id) ON DELETE CASCADE);
```

- Pokušaj brisanja department:id 110 iz tabele departments je uspeo i zavisni redovi u tabeli empoloyees su takođe obrisani (1 row(s) deleted)
- Kada FK je definisan sa ON CASCADE DELETE, nikakvo upozorenje se ne izdaje koje briše redove iz roditelj tabele što znači da zavisni redovi u dete tabeli su takođe obrisani

ON DELETE SET NULL

- Pre nego što će se obrisati redovi u dete tabeli pri korišćenju ON DELETE CASCADE opcije, dete redovi mogu biti napunjeni sa null vrednostima korišćenjem ON DELETE SET NULL opcijom

```
CREATE TABLE copy_employees
(employee_id NUMBER(6,0) CONSTRAINT copy_emp_pk PRIMARY KEY,
 first_name VARCHAR2(20),
 last_name VARCHAR2(25),
 department_id NUMBER(4,0),
 email VARCHAR2(25),
 CONSTRAINT cdept_dept_id_fk FOREIGN KEY (department_id)
    REFERENCES copy_departments(department_id) ON DELETE SET NULL);
```

- Ovo može biti korisno pri promeni vrednosti u roditelj tabeli na novi boj poput promene inventarskog broja u bar kod brojeve
- Ne želite obrisati redove u dete tabeli
- Kada novi bar-kod brojevi se unesu u roditelj tabelu, oni će onda moći da se unesu u dete tabelu bez obaveze da se iz korena ponovo prave redovi dete tabele

CHECK Constraints

- CHECK ograničenja eksplicitno definišu uslove koji se moraju ispuniti
- Da bi se zadovoljila ograničenja, svaki red u tabeli mora napraviti uslov ili True ili unknown (zbog null)
- Uslov od CHECK ograničenja se može referisati na bilo koju kolonu u specificiranoj tabeli, ali ne na kolone drugih tabela

CHECK Constraints Example

- CHECK ograničenje osigurava sa uneta vrednost za end_date je kasnija od start_date

```
CREATE TABLE copy_job_history
(employee_id NUMBER(6,0),
 start_date DATE,
 end_date DATE,
 job_id VARCHAR2(10),
 department_id NUMBER(4,0),
 CONSTRAINT cjhist_emp_id_st_date_pk
           PRIMARY KEY(employee_id, start_date),
 CONSTRAINT cjhist_end_ck CHECK (end_date > start_date));
```

- Pošto CHECK CONSTRAINT referencira dve kolone u tabeli, mora biti definisano na nivou tabele

CHECK Constraint Conditions

- CHECK ograničenje mora biti samo na redu gde je ograničenjew definisano
- CHECK ograničenje se ne može koristiti u upitima koji referišu na vrednsoti u drugim kolonama
- CHECK ograničenja ne mogu sadržati pozive na funkcije SYSDATE, UID, USER ili USERENV

- Iskaz CHECK (SYSDATE > '05-May-1999') nije dozvoljen
- CHECK ograničenje ne može koristiti pseudokolone CURRVAL, NEXTVAL, LEVEL ili ROWNUM
- Iskaz CHECK (NEXTVAL > 0) nije dozvoljeno
- Jedna kolona može imati više CHECK ograničenja koja referenciraju kolonu u njenu definiciju
- Ne postoji limit broja CHECK ograničenja koja se mogu definisati na kolonu

CHECK Constraint Syntax

- CHECK ograničenja se mogu definisati na nivou kolone ili nivou tabele
- Sintaksa za definisanje CHECK ograničenja:

– Column-level syntax:

```
salary NUMBER(8,2) CONSTRAINT employees_min_sal_ck CHECK (salary > 0)
```

– Table-level syntax:

```
CONSTRAINT employees_min_sal_ck CHECK (salary > 0)
```

14-3 Managing Constraints

- Db sistem mora biti sposoban da enfosira biznis pravila i istovremeno, spreči dodavanje, modifikovanje ili brisanje podataka koji mogu rezultovati u prekršaju referntial integrity db
- Treba znati praviti promene u ograničenjima na tabeli tako da se održavaju referntial integrity i dobri podaci u db čak i kada se podaci treba da menjaju

Managing Constraints

- Iskaz ALTER TABLE se koristi za pravljenje promena na ograničenjima u postojećim tabelama
- Ove promene mogu uključiti dodavanje ili izbacivanje ograničenja, omogućavajući ili sprečavajući ograničenja i dodavajući NOT NULL ograničenja u kolonu
- Uputstva za pravljenje promena na ograničenjima su:
    - Može se dodati, ispustiti, dozvoliti ili zabraniti ograničenje, ali se ne može modifikovati njena struktura
    - Može se dodati NOT NULL ograničenje na postojeću kolonu korišćenjem MODIFY iskaza od ALTER TABLE iskaza
    - MODIFY se koristi pošto NOT NULL je promena na nivou kolone
    - Može se definisati NOT NULL ograničenje samo ako tabela je prazna ili ako kolona sadrži vrednost za svaki red

ALTER Statement

- ALTER iskaz zahteva ime tabele, ime ograničenja, tip ograničenja, ime kolone na koju ograničenje ima efeket
- U sledećm primeru, korišćenje employees tabele, PK ograničenje je moglo da s edoda posle prvog kreiranja tabele:

```
ALTER TABLE employees
ADD CONSTRAINT  emp_id_pk PRIMARY KEY (employee_id);
```

Adding Constraints

- Za dodavanje ograničenja na postojeću tabelu, koristiti sledeću SQL sintaksu:

```
ALTER TABLE table_name
ADD [CONSTRAINT constraint_name] type of constraint (column_name);
```

- Ako je ograničenje FK ograničenje, REFERENCES služben areč mora biti uključena u iskazu
- Sintaksa:

```
ALTER TABLE tablename
ADD CONSTRAINT constraint_name FOREIGN KEY(column_name) REFERENCES
tablename(column_name);
```

Adding Constraints Example

- employees db; PK iz DEPARTMENTS tabele je unet u EMPLOYEES tabelu kao FK

DEPARTMENTS - Parent

DEPARTMENT_ID	DEPT_NAME	MANAGER_ID	LOCATION_ID
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting	-	1700

EMPLOYEE - Child

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID
100	Steven	King	90
101	Neena	Kochhar	90
102	Lex	De Haan	90
205	Shelley	Higgins	110
206	William	Gietz	110

- Sledeći primer demonstrira sintaksu kojom se dodaje ovaj FK u EMPLOYEES tabelu:

```
ALTER TABLE employees
ADD CONSTRAINT emp_dept_fk FOREIGN KEY (department_id)
   REFERENCES departments (department_id)
ON DELETE CASCADE;
```

DEPARTMENTS - Parent

DEPARTMENT_ID	DEPT_NAME	MANAGER_ID	LOCATION_ID
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting	-	1700

EMPLOYEE - Child

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID
100	Steven	King	90
101	Neena	Kochhar	90
102	Lex	De Haan	90
205	Shelley	Higgins	110
206	William	Gietz	110

Adding Constraints Conditions

- Ako ograničenje je NOT NULL ograničenje, ALTER TABLE iskaz koristi MODIFY umesto ADD
- NOT NULL ograničenje se može dodati samo ako tabela je prazna ili ako kolona sadrži vrednost za svaki red:

```
ALTER TABLE table_name
MODIFY (column_name CONSTRAINT constraint_name NOT NULL);
```

```
ALTER TABLE employees
MODIFY (email CONSTRAINT emp_email_nn NOT NULL);
```

Why Enable and Disable Constraints ?

- Da bi enforisao pravilo definisano sa integrity ograničenjem, ograničenje mora uvek da bude dozvoljeno
- U određenim slučajevima, ipak, poželjno je privremeno onemogućiti integrity ograničenja tabele zbog razloga performansi:
    - kada se učitava velika količina podataka u tabelu
    - kada se izvodi batch operacija koja čini velike promene u tabeli (kao što je promena broja svakog zaposlenog dodavanjem 1000 na postojeći broj)
- Provera opgraničenja, iako izvedena automatski, traži vreme, posebno proverom FK ograničenja. DBA ponekad ubrzava high-volume batch operacije ukidajući ograničenja. Rizik je da nepravilne promene mogu se napraviti na tabeli i da se ne mogu identifikovati kasnije kada se ograničenej ponovo dozvoli

Dropping Constraints

- Za odbacivanje ograničenja, treba znati ime ograničenja
- Ako se nezna ime ograničenja, možete pronaći ime ograničenja iz USER_CONSTRAINTS i USER_CONS_COLUMNS u DD
- CASCADE opcija od DROP izraza izaziva bilo koje zavisno ograničenje da takođe bude odbačeno
- Primetiti da kada se odbaci integrity ograničenje, to ograničenje nije više enforsirano od strane Oracle Servera i više nije dostupno u DD
- Odbacivanje PK ograničenje iz DEPARTMENTS tabele, reč CASCADE se koristi za odbacivanje FK ograničenja u dete tabeli. Tako da ako PK ograničenje u tabeli departments je odbačeno sa CASCADE opcijom, FK ograničenje u tabeli employees će takođe biti odbačeno
- Nijedan red ili podatak nije obrisan u bilo kojoj od tabela u pitanju kad ase odbaci ograničenje

```
ALTER TABLE table_name
DROP CONSTRAINT  name [CASCADE}
```

```
ALTER TABLE copy_departments
DROP CONSTRAINT  c_dept_dept_id_pk CASCADE;
```

Disabling Constraints

- Po difoltu, kada god integrity ograničenje je definisano u CREATE ili ALTER TABLE iskazu, ograničenje je automatski dopušteno (enforced) od Oracle osim ako nije specifično kreirano u disable stanju korišćenjem DISABLE izraza
- Ako onemogućite (disable) ograničenje bez njenog odbacivanja (dropping) ili ponovnog kreiranja njenog korišćenjem ALTER TABLE opcije DISABLE
- DISABLE omogućava unos podataka, bez obzira da li odgovara ograničenju
- Ova funkcija omogućava podacima da se dodaju u dete tabelu bez postojanja odgovarajućih vrednosti u roditelj tabeli
- DISABLE jednostavno gasi ograničenja

Using the DISABLE Clause

- Može se koristiti DISABLE izraz u ALTER tABLE iskazu i u CREATE TABLE iskazu

```
CREATE TABLE copy_employees
( employee_id NUMBER(6,0) PRIMARY KEY DISABLE,
   ...
   ...);

ALTER TABLE copy_employees
DISABLE CONSTRAINT c_emp_dept_id_fk;
```

- Onemogućavanje jedinstvenog ili PK ograničenja odstranjuje jedinstveni indeks

USING the CASCADE Clause

- CASCADE izraz onemogućava zavisni integritet ograničenja. Ako ograničenje je kasnije dopušteno, zavisno ograničenje nisu automatski dopuštena
- Sintaksa i primer:

```
ALTER TABLE table_name
DISABLE CONSTRAINT constraint_name [CASCADE];
```

```
ALTER TABLE copy_departments
DISABLE CONSTRAINT c_dept_dept_id_pk CASCADE;;
```

Enabling Constraints

- Za aktiviranje integritet ograničenja trenutno onemogućenog (disabled), koristiti ENABLE izraz u ALTER TABLE iskazu
- ENABLE osigurava da svi ulazni podaci su dopušteni od ograničenja
- Sintaks i primer:

```
ALTER TABLE table_name
ENABLE CONSTRAINT constraint_name;
```

```
ALTER TABLE copy_departments
ENABLE CONSTRAINT c_dept_dept_id_pk;
```

- Može se koristiti ENABLE izraz u CREATE TABLE iskazu i ALTER TABLE iskazu

Anabling Constraint Considerations

- Ako onemogućiš ograničenje, ovo ograničenje se primenjuje na sve podatke u tabeli
- Svi podaci u tabeli moraju odgovarati ograničenju
- Ako omogućiš UNIQUE KEY ili PK ograničenje, UNIQUE ili PK indeks se kreira automatski
- Dopuštanje PK ograničenja koje je onemogućeno sa CASCADE opcijom ne omogućava bilo koji FK koji je zavistan od PK
- ENABLE uključuje ograničenje ponovo

Cascading Constraints

- Kaskadiranje referential integrity ograničenja omogućava definisanje akcija koje db server preuzima kada korisnik pokušava obrisati ili updejtovati ključ na koji postojeći FK upućuje
- CASCADE CONSTRAINTS izraz se koristi zajedno sa DROP COLUMN izrazom

- Odbacuje sva referential integrity ograničenja koja referišu na PK i jedinstveni ključ definisane na odbačenim kolonama
- Takođe odbacuje sva višekolonska ograničenja definisana na odbačenim kolonama
- Ako ALTER TABLE iskaz ne uključuje CASCADE CONSTRAINTS opciju, bilo koji pokušaj odbacivanja PK ili višekolonskog ograničenja će pasti
- Zapamtiti, ne može se obrisati roditeljska vrednost ako dete vrednsot postoji u drugoj tabeli

```
ALTER TABLE table_name
DROP(column name(s)) CASCADE CONSTRAINTS;
```

When CASCADE is Not Required

- Ako sve kolone koje se referenciraju sa ograničenjem definisanim na odbačenim kolonama su takođe odbačene, onda CASCADE CONSTRAINTS nije obavezno
- Npr, pretpostavljajući da nijedan referential ograničenje od drugih tabela ne referiše na kolonu PK, validno je pretpostaviti sledeće iskaz bez CASCADE CONSTRAINTS izaraza:

```
ALTER TABLE tablename DROP
(pk_column_name(s));
```

- Ipak, ako bilo koje ograničenje je referencirano sa kolonama iz drugih tabela ili preostalim kolonama u ciljnim tabelama, moraš specificirati CASCADE CONSTRAINTS za izbegavanje bilo koje greške

Viewing Constraints

- Posle kreiranja tabele, možeš potvrditi njeno postojanje izdavanjem DESCRIBE komande
- Jedino ograničenje koje može verifikovati korišćenje DESCRIBE je NOT NU/LL ograničenje
- NOT NULL ograničenje će takođe se pojaviti u DD kao CHECK ograničenje
- Za pregled svih ograničenja u tabeli, upit USER_CONSTRAINTS daje tabelu:

```
SELECT constraint_name, table_name, constraint_type, status
FROM USER_CONSTRAINTS
WHERE table_name ='COPY_EMPLOYEES';
```

CONSTRAINT_NAME	TABLE_NAME	CONSTRAINT_TYPE	STATUS
COPY_EMP_PK	COPY_EMPLOYEES	P	ENABLED
CDEPT_DEPT_ID_FK	COPY_EMPLOYEES	R	ENABLED

Query USER_CONSTRAINTS

- Izlistani tipovi ograničenja u DD su: P (PK), R (REFERENCES, FK), C (CHECK uključujući NOT NULL), U (UNIQUE)