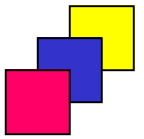
Język SQL. Rozdział 9.
Język definiowania danych DDL, część 2.

Ograniczenia integralnościowe, modyfikowanie struktury relacji, zarządzanie ograniczeniami.



Ograniczenia integralnościowe

- **Służą do weryfikacji poprawności danych relacji.**
- **Mogą być definiowane:**
  - **dla atrybutu,**
  - **dla relacji.**
- **Nazwa ograniczenia:**
  - **może zostać jawnie zdefiniowana przez użytkownika w klauzuli *CONSTRAINT*,**
  - **jest generowana automatycznie jeśli użytkownik nie zdefiniował jej jawnie.**
- **Moment weryfikacji danych przez ograniczenie:**
  - **domyślnie: w momencie wykonania operacji modyfikującej/ wstawiającej/usuwającej dane,**
  - **możliwość: w momencie zakończenia transakcji, w której miała miejsce operacja modyfikacji/wstawiania/usuwania danych.**

Wymagalność wartości atrybutu

- **Nie pozwala na umieszczenie w atrybucie wartości pustej.**
- **Miejsce definicji: dla atrybutu.**
- **Składnia:**

```
<nazwa_atrybutu> <typ_wartości>
        [CONSTRAINT <nazwa_ograniczenia>] NOT NULL
```

- **Dopuszczenie wartości pustej w atrybucie:**
  - **pominięcie definicji ograniczenia lub**
  - **jawne umieszczenie słowa NULL.**
- **Przykład:**

```
-- wartości puste niedozwolone:
nazwisko VARCHAR(15)  CONSTRAINT nn_nazwisko NOT NULL,
etat VARCHAR(10) NOT NULL,
-- wartości puste dozwolone
placa_pod NUMBER(6,2),
placa_dod NUMBER(6,2) NULL,
…
```

Klucz podstawowy (1)

- **Atrybut lub zbiór atrybutów, których wartości jednoznacznie określają rekord w relacji.**
- **Nie dopuszcza wartości pustych.**
- **Tylko jeden klucz podstawowy w relacji.**
- **Miejsce definicji:**
  - **dla atrybutu – dla klucza założonego na jednym atrybucie, składnia:**

```
<nazwa_atrybutu> <typ_wartości>
        [CONSTRAINT <nazwa_ograniczenia>] PRIMARY KEY
```

  - **dla relacji – zarówno dla klucza na jednym atrybucie jak i dla klucza na zbiorze atrybutów, składnia:**

```
[CONSTRAINT <nazwa_ograniczenia>]
        PRIMARY KEY (<lista_atrybutów>)
```

Klucz podstawowy (2)

- **Przykład 1. Definicja klucza podstawowego z jednym atrybutem:**
  - **dla atrybutu:**

```
…,
id_prac NUMBER(6)  CONSTRAINT pk_id_prac PRIMARY KEY,
…
```

  - **dla relacji:**

```
…
id_prac NUMBER(6),
…,
CONSTRAINT pk_id_prac PRIMARY KEY(id_prac),
…
```

Klucz podstawowy (3)

- **Przykład 2. Definicja klucza podstawowego z trzema atrybutami:**
  - **tylko dla relacji:**

```
…
imie VARCHAR(15),
nazwisko VARCHAR(15),
data_urodzenia DATE,
…,
CONSTRAINT pk_nazwisko_imie_data_ur
        PRIMARY KEY(nazwisko, imie, data_urodzenia),
…
```

Klucz unikalny (1)

- **Wymusza unikalność wartości atrybutu lub zbioru atrybutów w zbiorze rekordów relacji, pomija wartości puste.**
- **W relacji może być zdefiniowanych wiele kluczy unikalnych.**
- **Miejsce definicji:**
  - **dla atrybutu – dla klucza założonego na jednym atrybucie, składnia:**

```
<nazwa_atrybutu> <typ_wartości>
        [CONSTRAINT <nazwa_ograniczenia>] UNIQUE
```

  - **dla relacji – zarówno dla klucza na jednym atrybucie jak i dla klucza na zbiorze atrybutów, składnia:**

```
[CONSTRAINT <nazwa_ograniczenia>]
        UNIQUE(<lista_atrybutów>)
```

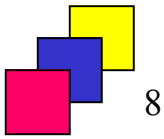
Klucz unikalny (2)

- **Przykład 1. Definicja klucza unikalnego z jednym atrybutem:**
  - **dla atrybutu:**

```
…,
nazwisko VARCHAR(15)  CONSTRAINT uk_nazwisko UNIQUE,
…
```

  - **dla relacji:**

```
…
nazwisko VARCHAR(15),
…,
CONSTRAINT uk_nazwisko UNIQUE(nazwisko),
…
```

Klucz unikalny (3)

- **Przykład 2. Definicja klucza unikalnego z dwoma atrybutami:**
  - **tylko dla relacji:**

```
…
imie VARCHAR(15),
nazwisko VARCHAR(15),
…,
CONSTRAINT uk_nazwisko_imie UNIQUE(nazwisko, imie),
…
```

Klucz obcy (1)

- **Definiuje zależność między relacjami: rekord w jednej relacji („relacja podrzędna”) jest „połączony” ze wskazanym rekordem innej relacji („relacja nadrzędna”),**
  - przykład: rekord opisujący pracownika (relacja *PRACOWNICY*) jest „połączony” z rekordem opisującym zespół, do którego pracownik należy (relacja *ZESPOLY*).
- **Może istnieć klucz obcy, w którym relacja nadrzędna i relacja podrzędna to ta sama relacja:**
  - przykład: rekord opisujący pracownika (relacja *PRACOWNICY*) jest „połączony” z rekordem opisującym jego przełożonego (relacja *PRACOWNICY*).
- **Alternatywna nazwa: ograniczenie referencyjne.**
- **Klucz obcy w relacji podrzędnej musi wskazywać na klucz podstawowy lub unikalny w relacji nadrzędnej.**
- **Klucz obcy dopuszcza wartości puste, chyba że zostaną wyeliminowane przez inne ograniczenie.**

Klucz obcy (2)

- **Miejsce definicji:**
  - **dla atrybutu – dla klucza założonego na jednym atrybucie, składnia:**

```
<nazwa_atrybutu> <typ_wartości>
        [CONSTRAINT <nazwa_ograniczenia>]
        REFERENCES <relacja_nadrzędna>
                (<nazwa_atrybutu_relacji_nadrzędnej>)
```

  - **dla relacji – zarówno dla klucza na jednym atrybucie jak i dla klucza na zbiorze atrybutów, składnia:**

```
[CONSTRAINT <nazwa_ograniczenia>]
        FOREIGN KEY(<lista_atrybutów_relacji_podrzędnej>)
        REFERENCES <relacja_nadrzędna>
                (<lista_atrybutów_relacji_nadrzędnej>)
```

Klucz obcy (3)

- **Przykład 1. Definicja klucza obcego z jednym atrybutem:**
  - **dla atrybutu:**

```
…,
id_zespolu NUMBER(4)  CONSTRAINT fk_zespoly
        REFERENCES zespoly(id_zesp),
…
```

  - **dla relacji:**

```
…,
id_zespolu NUMBER(4),
…,
CONSTRAINT fk_zespoly
        FOREIGN KEY(id_zespolu) -- atrybut relacji podrzędnej
        REFERENCES zespoly(id_zesp),
…
```

Klucz obcy (4)

- **Przykład 2. Definicja klucza obcego z dwoma atrybutami, założenie: klucz podstawowy w relacji nadrzędnej *POMIESZCZENIA* składa się z dwóch atrybutów o nazwach: *symbol_budynku* i *numer_pomieszczenia*:**
  - **tylko dla relacji:**

```
…,
budynek CHAR(10),
nr_pomieszczenia NUMBER(3),
…,
CONSTRAINT fk_pomieszczenia
  FOREIGN KEY(budynek, numer_pomieszczenia) -- atrybuty w relacji podrzędnej
  REFERENCES pomieszczenia(symbol_budynku, numer_pomieszczenia),
…
```

Klucz obcy (5)

- **Typ atrybutu w kluczu obcym musi być zgodny z typem atrybutu w kluczu podstawowym lub unikalnym, na który wskazuje klucz obcy; przykład:**

```
id_zespolu DATE CONSTRAINT fk_zespoly  REFERENCES zespoly(id_zesp),
-- Błąd: DATE nie jest zgodne z NUMBER(4)
```

- **Można pominąć definicję typu dla atrybutu w kluczu obcym – typ atrybutu zostanie ustawiony na identyczny z typem atrybutu w kluczu podstawowym lub unikalnym, z którym związany jest definiowany klucz obcy; przykład:**

```
id_zespolu CONSTRAINT fk_zespoly REFERENCES zespoly(id_zesp),
```

  - **atrybut *id_zespolu* uzyska typ *NUMBER(4)* – identyczny z typem atrybutu *id_zesp* relacji *ZESPOLY*.**

Usuwanie rekordu z relacji nadrzędnej (1)

- **Domyślne działanie: jeśli dla usuwanego rekordu relacji nadrzędnej istnieją w relacji podrzędnej rekordy powiązane, operacja usunięcia zostaje odrzucona.**

- **Alternatywa 1.: jeśli dla usuwanego rekordu relacji nadrzędnej istnieją w relacji podrzędnej rekordy powiązane, usunięty zostaje rekord relacji nadrzędnej oraz powiązane z nim rekordy relacji podrzędnej – tzw. usuwanie kaskadowe.**
  - implementacja: dodanie do ograniczenia klauzuli **ON DELETE CASCADE**.

- **Alternatywa 2.: jeśli dla usuwanego rekordu relacji nadrzędnej istnieją w relacji podrzędnej rekordy powiązane, usunięty zostaje rekord relacji nadrzędnej a w powiązanych rekordach relacji podrzędnej atrybut (-y) klucza obcego uzyskuje wartość NULL.**
  - implementacja: dodanie do ograniczenia klauzuli **ON DELETE SET NULL**.

Usuwanie rekordu z relacji nadrzędnej (2)

- **Przykład 1. Definicja klucza obcego z cechą usuwania kaskadowego:**

```
…,
id_zesp NUMBER(4)  CONSTRAINT fk_zespoly
        REFERENCES zespoly(id_zesp) ON DELETE CASCADE,
...
```

- **Przykład 2. Definicja klucza obcego z ustawianiem wartości pustych:**

```
…,
budynek CHAR(10),
nr_pomieszczenia NUMBER(3),
…,
CONSTRAINT fk_pomieszczenia FOREIGN KEY(budynek, nr_pomieszczenia)
        REFERENCES pomieszczenia(symbol_budynku, nr_pomieszczenia)
        ON DELETE SET NULL,
…
```

Ograniczenie domeny atrybutu (1)

- **Pozwala na definicję warunku logicznego, który musi być spełniony przez dane relacji.**

- **Warunek jest spełniony dla rekordu relacji, jeśli jego wartość jest równa TRUE (prawda) lub UNKNOWN (wartość nieznana, pusta).**

- **Warunek logiczny w ograniczeniu:**
  - **nie może odwoływać do atrybutów innej relacji,**
  - **nie może wykorzystywać zapytań,**
  - **nie może wołać funkcji, zdefiniowanych przez użytkownika, dopuszczalne jest wołanie funkcji predefiniowanych.**

Ograniczenie domeny atrybutu (2)

- **Miejsce definicji:**
  - **dla atrybutu – warunek odwołuje się tylko do jednego atrybutu relacji, składnia:**

```
<nazwa_atrybutu> <typ_wartości>
        [CONSTRAINT <nazwa_ograniczenia>]
        CHECK(<warunek_logiczny>)
```

  - **dla relacji – warunek odwołuje się do więcej niż jednego atrybutu relacji, składnia:**

```
[CONSTRAINT <nazwa_ograniczenia>]
        CHECK(<warunek_logiczny>)
```

Ograniczenie domeny atrybutu (3)

- **Przykład 1. Warunek odwołujący się do jednego atrybutu:**
  - **dla atrybutu:**

```
…,
placa_pod NUMBER(6,2)  CONSTRAINT chk_min_placa
          CHECK(placa_pod > 100),
…
```

  - **dla relacji:**

```
…,
placa_pod NUMBER(6,2),
…,
CONSTRAINT chk_min_placa CHECK(placa_pod > 100),
…
```

Ograniczenie domeny atrybutu (4)

- **Przykład 1. Warunek odwołujący się do dwóch atrybutów:**
  - **tylko dla relacji:**

```
…,
data_urodzenia DATE,
data_zatrudnienia DATE,
…,
CONSTRAINT chk_daty CHECK(data_urodzenia < data_zatrudnienia),
…
```

Łączenie ograniczeń integralnościowych

- **Atrybut może mieć zdefiniowanych wiele ograniczeń integralnościowych.**
- **Łączenie dotyczy tylko ograniczeń dla atrybutu.**
- **Kolejne ograniczenia wymienia się, oddzielając je spacją.**
- **Przykład:**

```
…,
placa_pod NUMBER(6,2)
        CONSTRAINT nn_placa_pod NOT NULL
        CONSTRAINT chk_min_placa CHECK(placa_pod > 100),
…,
id_zesp NUMBER(4)
        NOT NULL
        CONSTRAINT fk_zespoly REFERENCES zespoly(id_zesp),
…
```

Tworzenie relacji – przykład (1)

Tabele *DYDAKTYCY, PRZEDMIOTY i POMIESZCZENIA* przechowują, odpowiednio: dane wszystkich nauczycieli, dane o wykładanych przedmiotach oraz dane o pomieszczenia, w których mogą zostać przeprowadzone zajęcia.

```
CREATE TABLE dydaktycy (
  id_dydaktyka NUMBER(2) CONSTRAINT pk_dydaktycy PRIMARY KEY,
  nazwisko VARCHAR2(15) NOT NULL  CONSTRAINT uk_nazwisko UNIQUE,
  tytuł VARCHAR2(10) NOT NULL);

CREATE TABLE przedmioty (
  id_przedmiotu NUMBER(2) CONSTRAINT pk_przedmioty PRIMARY KEY,
  nazwa VARCHAR2(15) NOT NULL UNIQUE);

CREATE TABLE pomieszczenia (
  nr_pomieszczenia NUMBER(2),   nr_budynku NUMBER(2),
  pojemność NUMBER(4) NOT NULL CHECK(pojemność > 0),
  CONSTRAINT pk_pomieszczenia
          PRIMARY KEY(nr_pomieszczenia, nr_budynku));
```

Tworzenie relacji - przykład (2)

Tabela *ZAJECIA* łączy dane z tabel *DYDAKTYCY, PRZEDMIOTY* i *POMIESZCZENIA*, w tej tabeli przechowujemy dane o tym, kto wykłada jaki przedmiot i w jakiej formie.

```
CREATE TABLE zajecia (
 id_zajec NUMBER(2) CONSTRAINT pk_zajecia PRIMARY KEY,
 rodzaj_zaj VARCHAR2(15)
   CHECK (rodzaj_zaj IN ('wykład','ćwiczenia','laboratorium','projekt')),
 id_dydaktyka NUMBER(2) CONSTRAINT fk_zajecia_dydakt
        REFERENCES  dydaktycy(id_dydaktyka) ON DELETE SET NULL,
 id_przedmiotu NUMBER(2) NOT NULL CONSTRAINT fk_zajecia_przed
        REFERENCES przedmioty(id_przedmiotu) ON DELETE CASCADE,
 nr_pomieszczenia NUMBER(2) NOT NULL,
 nr_budynku NUMBER(2) NOT NULL,
 CONSTRAINT fk_zajecia_pom FOREIGN KEY (nr_pomieszczenia,
 nr_budynku) REFERENCES pomieszczenia(nr_pomieszczenia, nr_budynku));
```

Modyfikowanie struktury relacji (1)

- **Dodawanie nowych atrybutów:**

```
ALTER TABLE nazwa_relacji
ADD nazwa typ(rozmiar) [DEFAULT wartość]
      [definicja_ograniczeń_integralnościowych_atrybutu];
```

```
ALTER TABLE pracownicy
ADD data_ur DATE
   CONSTRAINT chk_data_ur CHECK(data_ur > DATE '1900-01-01');
```

- **Modyfikowanie istniejących atrybutów:**

```
ALTER TABLE nazwa_relacji
MODIFY nazwa typ(rozmiar) [DEFAULT wartość] [NOT NULL];
```

```
ALTER TABLE pracownicy
MODIFY nazwisko VARCHAR(50) NOT NULL;
```

- **Jeśli dodajesz lub modyfikujesz listę atrybutów, otocz je nawiasami.**

Modyfikowanie struktury relacji (2)

- **Usuwanie atrybutów z relacji:**

```
ALTER TABLE nazwa_relacji
DROP [COLUMN nazwa_atrybutu] | (lista_atrybutów);
```

```
ALTER TABLE pracownicy DROP COLUMN placa_pod;
```

```
ALTER TABLE pracownicy DROP (placa_pod, placa_dod);
```

- **Zmiana nazwy atrybutu relacji:**

```
ALTER TABLE nazwa_relacji
RENAME COLUMN stara_nazwa TO nowa_nazwa;
```

Zarządzanie ogr. integralnościowymi (1)

- **Dodawanie nowych ograniczeń integralnościowych:**

```
ALTER TABLE nazwa_relacji
ADD [CONSTRAINT nazwa_ograniczenia] definicja_ograniczenia;
```

  - **używamy składni ograniczenia dla relacji,**
  - **istniejące ograniczenie nie może być zmodyfikowane,**
  - **jeśli definiujesz kilka ograniczeń, otocz je nawiasami.**

```
ALTER TABLE pracownicy
ADD CONSTRAINT uk_nazwisko UNIQUE(nazwisko);
```

  - **ograniczenie NOT NULL dodajemy/usuwamy poleceniem modyfikacji struktury atrybutu:**

```
ALTER TABLE pracownicy MODIFY data_ur NOT NULL;
```

```
ALTER TABLE pracownicy MODIFY nazwisko NULL;
```

Zarządzanie ogr. integralnościowymi (2)

- **Wyłączenie ograniczenia integralnościowego:**

```
ALTER TABLE relacja
  DISABLE [CONSTRAINT nazwa] | [PRIMARY KEY] |
            [UNIQUE(lista_atrybutów_w_kluczu_unikalnym)];
```

```
ALTER TABLE pracownicy DISABLE CONSTRAINT prac_pk;
```

- **Włączenie ograniczenia integralnościowego:**

```
ALTER TABLE relacja
  ENABLE [CONSTRAINT nazwa] | [PRIMARY KEY] |
            [UNIQUE(lista_atrybutów_w_kluczu_unikalnym)];
```

```
ALTER TABLE pracownicy ENABLE PRIMARY KEY;
```

Zarządzanie ogr. integralnościowymi (3)

- **Usunięcie ograniczenia integralnościowego:**

```
ALTER TABLE relacja
  DROP [CONSTRAINT nazwa] |
          [[PRIMARY KEY] | [UNIQUE(lista_atrybutów_w_kluczu)]
          [CASCADE]];
```

```
ALTER TABLE pracownicy DROP  UNIQUE(nazwisko);
```

```
ALTER TABLE zespoly DROP PRIMARY KEY CASCADE;
```

Zmiana nazwy relacji, usuwanie relacji

- **Zmiana nazwy istniejącej relacji:**

```
ALTER TABLE stara_nazwa RENAME TO nowa nazwa;
```

- **Usunięcie relacji:**

```
DROP TABLE nazwa_relacji [CASCADE CONSTRAINTS];
```

  - **usuwane są dane z relacji i indeksy założone dla relacji,**
  - **jeżeli nie podano CASCADE CONSTRAINTS to polecenie może zakończyć się błędem (jeśli istnieją relacje zależne).**

Słownik bazy danych (dot. Oracle)

- **Perspektywy słownikowe opisujące ograniczenia integralnościowe**

Perspektywa	Opis
USER_CONSTRAINTS	Ograniczenia integralnościowe
USER_CONS_COLUMNS	Atrybuty ograniczeń integralnościowych

- **Przykład:**

```
SELECT constraint_name, constraint_type
FROM user_constraints
WHERE table_name = 'PRACOWNICY';
```

```
SELECT constraint_name, column_name
FROM user_cons_columns
WHERE table_name = 'PRACOWNICY'
ORDER BY constraint_name, position;
```