Projektowanie i implementacja Baz Danych DDL i tworzenie tabel

Antoni Ligęza

ligeza@agh.edu.pl
http://galaxy.uci.agh.edu.pl/~ligeza

DDL – Data Definition Language

Instrukcje DDL – CREATE

- `CREATE AGGREGATE` – tworzenie funkcji agregacji,
- `CREATE CONSTRAINT TRIGGER` – tworzenie procedury wyzwalanej jako ograniczenia,
- `CREATE DATABASE` – tworzenie bazy danych,
- `CREATE FUNCTION` – tworzenie nowej funkcji,
- `CREATE GROUP` – tworzenie grupy,
- `CREATE INDEX` – tworzenie indeksu,
- `CREATE LANGUAGE` – definiowanie nowego języka dla funkcji,
- `CREATE OPERATOR` – definiowanie nowego operatora użytkownika,
- `CREATE RULE` – definiowanie nowej reguły,
- `CREATE SEQUENCE` – definiowanie nowej sekwencji,
- `CREATE TABLE` – tworzenie tabel,
- `CREATE TABLE AS` – tworzenie tabel jako wyniku instrukcji SELECT,
- `CREATE TRIGGER` – tworzenie nowego wyzwalacza,
- `CREATE TYPE` – definiowanie nowego typu danych,
- `CREATE USER` – tworzenie nowego uzytkownika,
- `CREATE VIEW` – tworzenie nowej perspektywy.

DDL – Data Definition Language

Instrukcje DDL – ALTER i DROP

- `ALTER GROUP` – modyfikacja grupy (dodanie lub usunięcie użytkownika),
- `ALTER TABLE` – modyfikacja struktury tabeli,
- `ALTER USER` – modyfikacja konta uzytkownika
- `DROP AGGREGATE` – usuwanie funkcji agregacji,
- `DROP DATABASE` – usuwanie bazy danych,
- `DROP FUNCTION` – usuwanie funkcji,
- `DROP GROUP` – usuwanie grupy,
- `DROP INDEX` – usuwanie indeksu,
- `DROP LANGUAGE` – usuwanie zdefiniowanego języka dla funkcji,
- `DROP OPERATOR` – usuwanie zdefiniowanego operatora użytkownika,
- `DROP RULE` – usuwanie zdefiniowanej reguły,
- `DROP SEQUENCE` – usuwanie zdefiniowanej sekwencji,
- `DROP TABLE` – usuwanie tabeli,
- `DROP TRIGGER` – usuwanie wyzwalacza,
- `DROP TYPE` – usuwanie zdefiniowanego typu danych,
- `DROP USER` – usuwanie użytkownika,
- `DROP VIEW` – usuwanie perspektywy.

DDL – Data Definition Language

DDL – inne instrukcje

- `COMMENT ON` – dodaje komentarz do obiektu bazy danych,
- `COPY` – kopiuje dane do lub z tabeli,
- `DELETE FROM` – usuwanie danych z tabeli,
- `GRANT` – definiowanie uprawnień,
- `INSERT INTO` – wpisywanie danych do tablicy,
- `RESET` – przywraca domyślną wartość parametru,
- `REVOKE` – odbiera uprawnienia uzytkownikom,
- `SET` – ustawia parametry startowe bazy danych,

Definiowanie tabel

Definicja tabeli – zasada tworzenia

```
CREATE TABLE <nazwa_tablicy> (
          <A_1> <typ> <ograniczenia>,
          <A_2> <typ> <ograniczenia>,
           ...
          <A_n> <typ> <ograniczenia>,
          CONSTRAINT <nazwa_1> <typ> (<definicja>),
           ...
          CONSTRAINT <nazwa_k> <typ> (<definicja>)
                              );
```

Definicja tabeli – Przykład

```
booktown=# CREATE TABLE books (
booktown(#              id integer UNIQUE,
booktown(#              title text NOT NULL,
booktown(#              author_id integer,
booktown(#              subject_id integer,
booktown(#              CONSTRAINT books_id_pkey PRIMARY KEY (id));
```

Definicja tabeli – DDL

```
CREATE [ TEMPORARY | TEMP ] TABLE table_name (
            { column_name type [ column_constraint [...] ] |
              table_constraint }
            [, ...]
       )
       [ INHERITS ( inherited_table [,...] ) ]
```

Definiowanie tabel

Definicja ograniczeń na poziomie kolumny

```
column_constraint ::=
  [ CONSTRAINT column_constraint_name ]
  { NOT NULL | NULL | UNIQUE | PRIMARY KEY |
    DEFAULT default_value |
    CHECK (condition |
    REFERENCES foreign_table [ ( foreign_column ) ]
      [ MATCH FULL | MATCH PARTIAL ]
      [ ON DELETE action ]
      [ ON UPDATE action ]
      [ DEFERRABLE | NOT DEFERRABLE ]
      [ INITIALLY DEFERRED | INITIALLY IMMEDIATE ]}
```

Ograniczenia na poziomie tablic

```
table constraint ::=
  [ CONSTRAINT table_constraint_name ]
  { UNIQUE ( column_name [, ... ] ) |
    PRIMARY KEY ( column_name [, ... ] ) |
    CHECK ( condition ) |
    FOREIGN KEY ( column_name [, ... ] )
      REFERENCES foreign_table
      [ ( foreign_column [, ... ] ) ]
      [ MATCH FULL | MATCH PARTIAL ]
      [ ON DELETE action ]
      [ ON UPDATE action ]
      [ DEFERRABLE | NOT DEFERRABLE ]
      [ INITIALLY DEFERRED | INITIALLY IMMEDIATE ]}
action ::= { NO ACTION | RESTRICT | CASCADE | SET NULL | SET DEFAULT }
```

Przykłady definicji tablic

```
booktown=# CREATE TABLE shipments (
booktown(#   id integer NOT NULL DEFAULT nextval('shipments_ship_id_seq')
booktown(#   customer_id integer,
booktown(#   isbn text,
booktown(#   ship_date timestamp);
CREATE

booktown=# CREATE TABLE employees
booktown-#        (id integer PRIMARY KEY CHECK (id > 100),
booktown(#         last_name text NOT NULL,
booktown(#         first_name text);
NOTICE:  CREATE TABLE/PRIMARY KEY will create implicit index
'employees_pkey' for table 'employees'
CREATE

booktown=# CREATE TABLE distinguished_authors (award text)
booktown-#              INHERITS (authors);
CREATE
booktown=# \d distinguished_authors
  Table "distinguished_authors"
 Attribute  |  Type   | Modifier
------------+---------+----------
 id         | integer | not null
 last_name  | text    |
 first_name | text    |
 award      | text    |
```

Przykłady definicji tablic

Tablica z ograniczeniami i kluczem obcym

```
booktown=# CREATE TABLE editions
booktown-#    (isbn text,
booktown(#     book_id integer,
booktown(#     edition integer,
booktown(#     publisher_id integer,
booktown(#     publication date,
booktown(#     type char,
booktown(#     CONSTRAINT pkey PRIMARY KEY (isbn),
booktown(#     CONSTRAINT integrity CHECK (book_id IS NOT NULL
booktown(#                                 AND edition IS NOT NULL),
booktown(#     CONSTRAINT book_exists FOREIGN KEY (book_id)
booktown(#                REFERENCES books (id)
booktown(#                ON DELETE CASCADE
booktown(#                ON UPDATE CASCADE);
NOTICE:  CREATE TABLE/PRIMARY KEY will create
implicit index 'pkey' for table 'editions'
NOTICE:  CREATE TABLE will create implicit trigger(s) for
FOREIGN KEY check(s)
CREATE
```

Modyfikacja Tablic

Schemat modyfikacji – ALTER TABLE

```
ALTER TABLE table
        ADD [ CONSTRAINT name ]
        { CHECK ( condition ) |
          FOREIGN KEY ( column [, ... ] )
                      REFERENCES table [ ( column [, ... ] ) ]
                      [ MATCH FULL | MATCH PARTIAL ]
                      [ ON DELETE action ]
                      [ ON UPDATE action ]
                      [ DEFERRABLE | NOT DEFERRABLE ]
                      [ INITIALLY DEFERRED | INITIALLY IMMEDIATE ]
        }
```

```
booktown=# ALTER TABLE books ADD CONSTRAINT legal_subjects
booktown-#                   FOREIGN KEY (subject_id)
booktown-#                   REFERENCES subjects (id);
NOTICE:  ALTER TABLE ... ADD CONSTRAINT will create
implicit trigger(s) for FOREIGN KEY check(s)
CREATE
```

Przykład

CREATE TABLE z INSERT INTO i zmiana nazw tabel

```
booktown=# DROP INDEX books_id_pkey;
DROP
booktown=# CREATE TABLE new_books
booktown-#              (id integer CONSTRAINT books_id_pkey PRIMARY KEY,
booktown(#              title text NOT NULL,
booktown(#              author_id integer,
booktown(#              subject_id integer);
NOTICE:  CREATE TABLE/PRIMARY KEY will create implicit index
'books_id_pkey' for table 'new_books'
CREATE
booktown=# INSERT INTO new_books SELECT * FROM books;
INSERT 0 15
booktown=# ALTER TABLE books RENAME TO old_books;
ALTER
booktown=# ALTER TABLE new_books RENAME TO books;
ALTER
```

ALTER TABLE – Przykłady

Dodawanie kolumny

```
booktown=# ALTER TABLE books
booktown-#       ADD publication date;
ALTER
booktown=# \d books
          Table "books"
  Attribute  |  Type   | Modifier
-------------+---------+----------
 id          | integer | not null
 title       | text    | not null
 author_id   | integer |
 subject_id  | integer |
 publication | date    |
Index: books_id_pkey
```

Zmiana nazw kolumn

```
booktown=# \d daily_inventory
    Table "daily_inventory"
 Attribute |  Type   | Modifier
-----------+---------+----------
 isbn      | text    |
 in_stock  | boolean |
booktown=# ALTER TABLE daily_inventory
booktown-#       RENAME COLUMN in_stock TO is_in_stock;
ALTER
booktown=# ALTER TABLE daily_inventory
booktown-#       RENAME is_in_stock TO is_stocked;
ALTER
```

ALTER TABLE – Przykłady

Zmiana własności

```
booktown=# ALTER TABLE books
booktown-#       ALTER COLUMN id
booktown-#       SET DEFAULT nextval('book_ids');
ALTER
booktown=# \d books
                          Table "books"
 Attribute  |  Type   |                  Modifier
------------+---------+-------------------------------------------
 id         | integer | not null default nextval('book_ids'::text)
 title      | text    | not null
 author_id  | integer |
 subject_id | integer |
Index: books_id_pkey

booktown=# ALTER TABLE books
booktown-#       ALTER id
booktown-#       DROP DEFAULT;
ALTER
booktown=# \d books
          Table "books"
 Attribute  |  Type   | Modifier
------------+---------+----------
 id         | integer | not null
 title      | text    | not null
 author_id  | integer |
 subject_id | integer |
Index: books_id_pkey
```

ALTER TABLE – Przykłady

Zmiana nazwy tabeli

```
booktown=# ALTER TABLE books RENAME TO literature;
ALTER
booktown=# ALTER TABLE literature RENAME TO books;
ALTER
```

Dodawanie ograniczeń

```
booktown=# ALTER TABLE editions
booktown-#       ADD CONSTRAINT foreign_book
booktown-#       FOREIGN KEY (book_id) REFERENCES books (id);
NOTICE: ALTER TABLE ... ADD CONSTRAINT will create
implicit trigger(s) for FOREIGN KEY check(s)
CREATE
booktown=# ALTER TABLE editions
booktown-#       ADD CONSTRAINT hard_or_paper_back
booktown-#       CHECK (type = 'p' OR type = 'h');
ALTER
```

ALTER TABLE – Restrukturyzacja tablicy z CREATE TABLE AS

```
          Table "books"
  Attribute  |  Type   | Modifier
-------------+---------+----------
 id          | integer | not null
 title       | text    | not null
 author_id   | integer |
 subject_id  | integer |
 publication | date    |
Index: books_id_pkey
booktown=# CREATE TABLE new_books
booktown-#        (id, title, author_id, subject_id)
booktown-#        AS SELECT id, title, author_id, subject_id
booktown-#                  FROM books;
SELECT
booktown=# ALTER TABLE books RENAME TO old_books;
ALTER
booktown=# ALTER TABLE new_books RENAME TO books;
ALTER
booktown=# \d books
        Table "books"
 Attribute  |  Type   | Modifier
------------+---------+----------
 id         | integer |
 title      | text    |
 author_id  | integer |
 subject_id | integer |
booktown=# DROP TABLE books;
DROP
```

ALTER TABLE – Restrukturyzacja tablicy z INSERT INTO

```
booktown=# CREATE TABLE new_books (
booktown(#   id integer UNIQUE,
booktown(#   title text NOT NULL,
booktown(#   author_id integer,
booktown(#   subject_id integer,
booktown(#   CONSTRAINT books_pkey PRIMARY KEY (id)
booktown(# );
NOTICE:  CREATE TABLE/PRIMARY KEY will create implicit index 'books_id_pk
for table 'new_books'
CREATE
booktown=# INSERT INTO new_books
booktown-#             SELECT id, title, author_id, subject_id
booktown-#                    FROM books;
INSERT 0 12
booktown=# ALTER TABLE books RENAME TO old_books;
ALTER
booktown=# ALTER TABLE new_books RENAME TO books;
ALTER
booktown=# \d books
          Table "books"
 Attribute  |  Type   | Modifier
------------+---------+----------
 id         | integer | not null
 title      | text    | not null
 author_id  | integer |
 subject_id | integer |
Index: books_id_pkey
```

Definiowanie i użycie sekwencji

Definiowanie sekwencji

```
 CREATE SEQUENCE seqname [ INCREMENT increment ]
         [ MINVALUE minvalue ] [ MAXVALUE maxvalue ]
         [ START start ] [ CACHE cache ] [ CYCLE ]

booktown=# CREATE SEQUENCE shipments_ship_id_seq
booktown-#                 START 200 INCREMENT 1;
CREATE

booktown=# SELECT nextval ('shipments_ship_id_seq');
 nextval
---------
     200
(1 row)

booktown=# INSERT INTO shipments VALUES
booktown-#        (nextval('shipments_ship_id_seq'), 107,
   '0394800753', 'now');

booktown=# CREATE TABLE shipments (
booktown(#   id integer NOT NULL DEFAULT
   nextval('shipments_ship_id_seq'),
booktown(#   customer_id integer,
booktown(#   isbn text,
booktown(#   ship_date timestamp);
```

Usuwanie tablic – DROP TABLE

Schemat usuwania

```
DROP TABLE tablename

DROP TABLE name [, ...]
```

Przykład usuwania tablic

```
booktown=# DROP TABLE employees;
DROP
```

Uwaga: Usuwanie tablicy nie usuwa automatycznie związanych z nią obiektów, np. sekwencji. Dlatego dla "wyczyszczenia" bazy przed ponownym zakładaniem tablicy trzeba usunąć "ręcznie" takie obiekty. Przykład: usuwanie sekwencji związanych z tabelą.

```
booktown=# DROP SEQUENCE shipments_ship_id_seq;
DROP
```

Wprowadzanie danych do tabel – INSERT INTO

Schemat instrukcji

```
INSERT INTO table_name
        [ ( column_name [, ...] ) ]
        VALUES ( value [, ...] )

 INSERT INTO table_name
        [ ( column_name [, ...] ) ]
        query
```

Przykłady

```
booktown=# INSERT INTO books (id, title, author_id, subject_id)
booktown-#        VALUES (41472, 'Practical PostgreSQL', 1212, 4);
INSERT 3574037 1

booktown=# INSERT INTO books (subject_id, author_id, id, title)
booktown-#        VALUES (4, 7805, 41473, 'Programming Python');
INSERT 3574041 1

booktown=# INSERT INTO books (id, title, author_id, subject_id)
booktown-#        SELECT nextval('book_ids'), title, author_id, subject_i
booktown-#               FROM book_queue WHERE approved;
INSERT 0 2
```

Ładowanie tablic – COPY

Schemat instrukcji

```
COPY [ BINARY ] table_name [ WITH OIDS ]
     FROM { 'filename' | stdin }
     [ [USING] DELIMITERS 'delimiter' ]
     [ WITH NULL AS 'null_string' ]
```

Przykłady

```
booktown=# COPY subjects FROM '/tmp/subjects.sql'
booktown-#               USING DELIMITERS ',' WITH NULL AS '\null';
COPY

booktown=# COPY books TO 'filename';
COPY

1,Business,Productivity Ave
2,Children's Books,Kids Ct
3,Classics,Academic Rd
9,Horror,Black Raven Dr
10,Mystery,Black Raven Dr
11,Poetry,Sunset Dr
13,Romance,Main St
14,Science,Productivity Ave
15,Science Fiction,Main St
0,Arts,Creativity St
6,Drama,Main St
7,Entertainment,Main St
```

Modyfikacja zawartości tabel – UPDATE

Schemat instrukcji

```
UPDATE [ ONLY ] table SET
       column = expression [, ...]
       [ FROM source ]
       [ WHERE condition ]
```

Przykład postepowania

```
booktown=# SELECT retail FROM stock
booktown-#        WHERE isbn = '0590445065';
 retail
--------
  23.95
(1 row)

booktown=# UPDATE stock
booktown-#        SET retail = 25.95
booktown-#        WHERE isbn = '0590445065';
UPDATE 1
booktown=# SELECT retail FROM stock
booktown-#        WHERE isbn = '0590445065';
 retail
--------
  25.95
(1 row)
```

Modyfikacja zawartości tabel – UPDATE

Schemat postepowania

```
booktown=# SELECT isbn, retail, cost
booktown-#        FROM stock
booktown-#        ORDER BY isbn ASC
booktown-#        LIMIT 3;
    isbn    | retail | cost
------------+--------+-------
 0385121679 |  36.95 | 29.00
 039480001X |  32.95 | 30.00
 0394800753 |  16.95 | 16.00
(3 rows)

booktown=# UPDATE stock
booktown-#        SET retail =
booktown-#            (cost * ((retail / cost) + 0.1::numeric));
UPDATE 16

booktown=# SELECT isbn, retail, cost
booktown-#        FROM stock
booktown-#        ORDER BY isbn ASC
booktown-#        LIMIT 3;
    isbn    | retail | cost
------------+--------+-------
 0385121679 |  39.85 | 29.00
 039480001X |  35.95 | 30.00
 0394800753 |  18.55 | 16.00
(3 rows)
```

Modyfikacja zawartości tabel – UPDATE

Jednoczesna modyfikacja wielu kolumn

```
booktown=# UPDATE publishers
booktown-#        SET name = 'O\'Reilly & Associates',
booktown-#            address = 'O\'Reilly & Associates, Inc. '
booktown-#                   || '101 Morris St, Sebastopol, CA 95472'
booktown-#        WHERE id = 113;
UPDATE 1
booktown=# SELECT name, substr(address, 1, 40) || '...' AS short_address
booktown-#        FROM publishers
booktown-#        WHERE id = 113;
         name          |                short_address
-----------------------+-----------------------------------------
 O'Reilly & Associates | O'Reilly & Associates, Inc. 101 Morris S...
(1 row)
```

Kasowanie rekordów – DELETE FROM

Schemat instrukcji

```
DELETE FROM [ ONLY ] table
        [ WHERE condition ]
```

Przykłady

```
booktown=# SELECT * FROM stock
booktown-#          WHERE stock = 0;
    isbn    | cost  | retail | stock
------------+-------+--------+-------
 0394800753 | 16.00 |  16.95 |     0
 0394900014 | 23.00 |  23.95 |     0
 0451198492 | 36.00 |  46.95 |     0
 0451457994 | 17.00 |  22.95 |     0
(4 rows)

booktown=# DELETE FROM stock
booktown-#        WHERE stock = 0;
DELETE 4
```

Uwaga:

```
booktown=# DELETE FROM stock_backup;
DELETE 16
```

Tworzenie baz danych – CREATE DATABASE

Schemat instrukcji

```
CREATE DATABASE name
  [ WITH [ LOCATION = { 'dbpath' | DEFAULT } ]
         [ TEMPLATE = template | DEFAULT ]
         [ ENCODING = encoding_name | encoding_number | DEFAULT ] ]
```

Przykłady

```
template1=# CREATE DATABASE booktown;
CREATE DATABASE

template1=# CREATE DATABASE booktown WITH LOCATION = '/usr/local/pgsql/bo
CREATE DATABASE


-- polish

DROP DATABASE nf;
CREATE DATABASE nf WITH ENCODING = 'LATIN2';
```