Bazy danych

Opracowano na podstawie:
Thomas Connolly, Carolyn Begg, Systemy baz danych, Wydawnictwo RM, 2004

1 Fizyczne projektowanie bazy danych dla modelu relacyjnego

"Fizyczne projektowanie bazy danych [3] – Proces tworzenia implementacji bazy danych w pamięci zewnętrznej; opis ten zawiera relacje bazowe, organizację plików, indeksy ułatwiające efektywny dostęp do danych, a także wszystkie więzy integralności i mechanizmy ochrony bezpieczeństwa."

Krok 4

Wyrażenie globalnego logicznego modelu danych w docelowym SZBD

Krok 4.1

Zaprojektuj relacje bazowe

Krok 4.2

Zaprojektuj reprezentację danych pochodnych

Krok 4.3

Zaprojektuj więzy ogólne

Krok 5

Projektowanie reprezentacji fizycznej

Krok 5.1

Przeanalizuj transakcje

Krok 5.2

Ustal organizację plików

Krok 5.3

Wybierz indeksy

Krok 5.4

Oszacuj wymaganą pamięć dyskową

Krok 6

Projektowanie perspektyw użytkownika

Krok 7

Projektowanie mechanizmów bezpieczeństwa

Krok 8

Rozważenie wprowadzenia kontrolowanej redundancji

Krok 8.1

Łączenie relacji występujących w związkach wzajemnie jednoznacznych (1:1)

Krok 8.2

Redukowanie liczby złączeń poprzez powielanie atrybutów, które nie należą do kluczy w związkach typu „jeden do wielu" (1:*)

Krok 8.3

Redukowanie liczby złączeń poprzez powielanie atrybutów kluczy obcych w związkach typu „jeden do wielu" (1:*)

Krok 8.4

Redukowanie liczby złączeń poprzez powielanie atrybutów w związkach „wiele do wielu" (*:*)

Krok 8.5

Wprowadzenie powtarzających się grup

Krok 8.6

Scalanie tabel podglądu z relacjami bazowymi

Krok 8.7

Tworzenie tabel skrótowych

Krok 9

Monitorowanie i dostrajanie systemu operacyjnego

2 Model relacyjny

Model relacyjny [3] dostarcza tylko jednego sposobu reprezentowania danych: jest nim dwuwymiarowa tabela, nazywana relacją. W nagłówku relacji są podawane atrybuty.

Wiersze relacji, poza wierszem nagłówka, zawierającym atrybuty relacji, są nazywane krotkami. W krotce każdy atrybut ma swój odpowiednik w postaci składowej krotki.

NrIndeksu	Nazwisko	Imię	NrGrupy
99001	AAAA	ZZZZ	211A
99002	BBBB	YYYY	211A
99003	CCCC	XXXX	211B
99004	DDDD	WWWW	211B

Nazwa relacji oraz jej zbiór atrybutów nazywają się schematem relacji. Schemat relacji zapisujemy za pomocą jej nazwy, po której występuje lista atrybutów ujęta w nawiasy okrągłe, np.

```
Students(id, lastname, firstname, group_id)
```

Zbiór schematów relacji projektu [3] jest określany schematem relacyjnym bazy danych lub po prostu schematem bazy danych.

Zakłada się, że z każdym atrybutem relacji jest powiązana dziedzina, czyli pewien określony typ elementarny. Każda składowa każdej krotki w relacji ma wartość, która należy do dziedziny odpowiedniej kolumny.

W modelu relacyjnym każda składowa każdej relacji musi mieć określony typ atomowy, tzn. jej typ musi należeć do typów elementarnych, np. musi być całkowity lub znakowy. Wartość atrybutu nie może być ani rekordem, ani listą, ani tablicą, ani zbiorem, ani żadną inną strukturą, którą można podzielić na mniejsze części.

Odpowiedniki terminologiczne dla pojęć modelu relacyjnego [2]:

Nazwa formalna	Pierwszy odpowiednik	Drugi odpowiednik
Relacja	Tablica	Plik
Krotka	Wiersz	Rekord
Atrybut	Kolumna	Pole

3 SQL (Structured Query Language)

SQL (ang. Structured Query Language) jest strukturalnym językiem zapytań umożliwiającym wykonywanie operacji na relacyjnych bazach danych. SQL zawiera [3]:

- język definicji danych DDL (ang. Data Definition Language), służący do definiowania struktur danych i do kontroli dostępu do danych,
- język manipulowania danymi DML (ang. Data Manipulation Language) służący do wyszukiwania i modyfikowania dancyh.

4 Typy danych

Źródło:

`https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/Data-Types.html`

Wybrane typy danych Oracle	Opis
VARCHAR2(size [BYTE | CHAR])	Zmiennej długości ciąg znaków o długości od 1 do size bajtów lub znaków. Maksymalny rozmiar jest równy: 32767 bytów lub znaków jeżeli MAX_STRING_SIZE = EXTENDED, 4000 bytów lub znaków MAX_STRING_SIZE = STANDARD
NVARCHAR2(size)	Zmiennej długości ciąg znaków Unicode o maksymalnej długości size znaków. Liczba bajtów może być do dwóch razy większa dla kodowania AL16UTF16 oraz trzy razy większa dla kodowania UTF8. Maksymalny rozmiar jest równy: 32767 bytów jeżeli MAX_STRING_SIZE = EXTENDED, 4000 bytów MAX_STRING_SIZE = STANDARD
NUMBER [(p[,s])]	Liczba o precyzji p z zakresu od 1 do 38 i skali s z zakresu od -84 do 127.
FLOAT [(p)]	Podtyp NUMBER o precyzji *precision* z zakresu od 1 do 126 binarnych cyfr.

LONG	Dane znakowe zmiennej długości maksymalnie do 2GB lub $2^{31} - 1$ bajtów.
DATE	Data z zakresu od 1 stycznia 4712 r.p.n.e. do 31 grudnia 9999 r.n.e.
BINARY_FLOAT	32-bitowa liczba zmiennoprzecinkowa.
BINARY_DOUBLE	64-bitowa liczba zmiennoprzecinkowa
TIMESTAMP [(fractional_seconds_precision)]	Znacznik czasu
TIMESTAMP [(fractional_seconds_precision)] WITH TIME ZONE	Znacznik czasu ze strefą czasową
TIMESTAMP [(fractional_seconds_precision)] WITH LOCAL TIME ZONE	Znacznik czasu z lokalną strefą czasową
INTERVAL YEAR [(year_precision)] TO MONTH	Przechowuje okres czasu wyrażony w latach oraz miesiącach.
INTERVAL DAY [(day_precision)] TO SECOND [(fractional_seconds_precision)]	Przechowuje okres czasu wyrażony w dniach, godzinach, minutach i sekundach.
RAW(size)	Dane binarne. Maksymalna długość to 32767 bytów jeżeli MAX_STRING_SIZE = EXTENDED, 2000 bytów jeżeli MAX_STRING_SIZE = STANDARD.
LONG RAW	Dane binarne o długości do 2GB.
ROWID	Tekst reprezentujący unikalny adres wiersza w tabeli.
UROWID [(size)]	Tekst reprezentujący logiczny adres wiersza w tabeli zorganizowanej przy pomocy indeksu.
CHAR [(size [BYTE | CHAR])]	Stałej długości ciąg znaków o maksymalnej długości od 1 do 2000 bajtów lub znaków.
NCHAR[(size)]	Stałej długości ciąg znaków o maksymalnej długości wyrażonej w znakach. Liczba bajtów może być do dwóch razy większa dla kodowania AL16UTF16 oraz trzy razy większa dla kodowania UTF8. Maksymalny rozmiar zależy od użytego kodowania z limitem 2000 bajtów.
CLOB	Duży obiekt znakowy zawierający jednobajtowe lub wielobajtowe znaki. Maksymalny rozmiar to (4GB - 1) * (rozmiar bloku bazy danych).
NCLOB	Duży obiekt znakowy zawierający znaki zapisane w formacie Unicode. Maksymalny rozmiar to (4GB - 1) * (rozmiar bloku bazy danych).
BLOB	Duży obiekt binarny. Maksymalny rozmiar to (4GB - 1) * (rozmiar bloku bazy danych).
BFILE	Zawiera locator do dużego pliku binarnego przechowywanego poza bazą danych. Makysmalny rozmar to 4GB.

NUMBER[(precision [, scale])]

Liczba o precyzji precision z zakresu od 1 do 38 i skali scale z zakresu od -84 do 127.

```
DROP TABLE POSITIVE_SCALE;
CREATE TABLE POSITIVE_SCALE(
    id NUMBER (2),
    f NUMBER(4,1)
);

INSERT INTO POSITIVE_SCALE VALUES (1, 0.009876);
INSERT INTO POSITIVE_SCALE VALUES (2, 0.09876);
INSERT INTO POSITIVE_SCALE VALUES (3, 0.9876);
INSERT INTO POSITIVE_SCALE VALUES (4, 9.876);
INSERT INTO POSITIVE_SCALE VALUES (5, 98.76);
INSERT INTO POSITIVE_SCALE VALUES (6, 987.6);
INSERT INTO POSITIVE_SCALE VALUES (7, 9876);
```

```
ORA-01438: value larger than specified precision allowed for this column
```

```
SELECT * FROM POSITIVE_SCALE;
```

```
        ID          F
---------- ----------
         1          0
         2         ,1
         3          1
         4        9,9
         5       98,8
         6      987,6
```

```
DROP TABLE NEGATIVE_SCALE;
CREATE TABLE NEGATIVE_SCALE(
    id NUMBER (2),
    f NUMBER(4,-1)
);

INSERT INTO NEGATIVE_SCALE VALUES (1, 0.009876);
INSERT INTO NEGATIVE_SCALE VALUES (2, 0.09876);
INSERT INTO NEGATIVE_SCALE VALUES (3, 0.9876);
INSERT INTO NEGATIVE_SCALE VALUES (4, 9.876);
INSERT INTO NEGATIVE_SCALE VALUES (5, 98.76);
INSERT INTO NEGATIVE_SCALE VALUES (6, 987.6);
INSERT INTO NEGATIVE_SCALE VALUES (7, 9876);
```

```
SELECT * FROM NEGATIVE_SCALE;
```

```
        ID          F
---------- ----------
         1          0
         2          0
         3          0
         4         10
         5        100
         6        990
         7       9880
```

5 Tworzenie tabel (CREATE TABLE)

Instrukcja CREATE TABLE służy do tworzenia tabel.

Źródło: SQL Language Reference
https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/CREATE-TABLE.html

create_table

```
CREATE [ { GLOBAL | PRIVATE } TEMPORARY | SHARDED | DUPLICATED ] TABLE
  [ schema. ] table
  [ SHARING = { METADATA | DATA | EXTENDED DATA | NONE } ]
  { relational_table | object_table | XMLType_table }
  [ MEMOPTIMIZE FOR READ ]
  [ PARENT [ schema. ] table ] ;
```

relational_table

```
[ (relational_properties) ]
[ DEFAULT COLLATION collation_name ]
[ ON COMMIT { DROP | PRESERVE } DEFINITION ]
[ ON COMMIT { DELETE | PRESERVE } ROWS ]
[ physical_properties ]
[ table_properties ]
```

relational_properties

```
{ column_definition
| virtual_column_definition
| period_definition
| { out_of_line_constraint | out_of_line_ref_constraint }
| supplemental_logging_props
}
  [, { column_definition
     | virtual_column_definition
     | period_definition
     | { out_of_line_constraint | out_of_line_ref_constraint }
     | supplemental_logging_props
```

```
    }
  ]...
```

column_definition

```
column [ datatype [ COLLATE column_collation_name ] ]
  [ SORT ] [ VISIBLE | INVISIBLE ]
  [ DEFAULT [ ON NULL ] expr | identity_clause ]
  [ ENCRYPT encryption_spec ]
  [ { inline_constraint }...
  | inline_ref_constraint
  ]
```

virtual_column_definition

```
column [ datatype [ COLLATE column_collation_name ] ]
  [ VISIBLE | INVISIBLE ]
  [ GENERATED ALWAYS ] AS (column_expression) [ VIRTUAL ]
  [ evaluation_edition_clause ] [ unusable_editions_clause ]
  [ inline_constraint [ inline_constraint ]... ]
```

inline_constraint

```
[ CONSTRAINT constraint_name ]
{ [ NOT ] NULL
| UNIQUE
| PRIMARY KEY
| references_clause
| CHECK (condition)
}
[ constraint_state ]
```

out_of_line_constraint

```
[ CONSTRAINT constraint_name ]
{ UNIQUE (column [, column ]...)
| PRIMARY KEY (column [, column ]...)
| FOREIGN KEY (column [, column ]...) references_clause
| CHECK (condition)
} [ constraint_state ]
```

inline_ref_constraint

```
{ SCOPE IS [ schema. ] scope_table
| WITH ROWID
| [ CONSTRAINT constraint_name ]
  references_clause
  [ constraint_state ]
}
```

out_of_line_ref_constraint

```
{ SCOPE FOR ({ ref_col | ref_attr })
    IS [ schema. ] scope_table
| REF ({ ref_col | ref_attr }) WITH ROWID
| [ CONSTRAINT constraint_name ] FOREIGN KEY
    ( { ref_col [, ref_col ] | ref_attr [, ref_attr ] } ) references_clause
    [ constraint_state ]
}
```

references_clause

```
REFERENCES [ schema. ] object [ (column [, column ]...) ]
  [ON DELETE { CASCADE | SET NULL } ]
```

Uproszczona składnia CREATE TABLE:

```
CREATE TABLE tabele_name (
    column_name data_type [column_constriants]
    {, column_name data_type [column_constriants]}
    [table_constriants]
);
```

Przykłady:

```
CREATE TABLE grupy(
    id NUMBER(6) CONSTRAINT grupa_pk PRIMARY KEY,
    nazwa VARCHAR2(20));
```

```
CREATE TABLE studenci(
    id NUMBER(6) CONSTRAINT student_pk PRIMARY KEY,
    nazwisko VARCHAR2(20),
    imie VARCHAR2(20),
    srednia NUMBER (3,2),
    id_grupa NUMBER(6) CONSTRAINT student_fk REFERENCES grupy(id));
```

Listing 1: {sql/test_create1.sql}

```
CREATE TABLE grupy(
    id NUMBER(6),
    nazwa VARCHAR2(20),
    CONSTRAINT grupa_pk PRIMARY KEY(id));

CREATE TABLE studenci(
    id NUMBER(6) CONSTRAINT student_pk PRIMARY KEY,
    nazwisko VARCHAR2(20),
    imie VARCHAR2(20),
    srednia NUMBER (3,2),
    id_grupa NUMBER(6),
    CONSTRAINT student_fk FOREIGN KEY (id_grupa) REFERENCES grupy(id));
```

Listing 2: {sql/test_create2.sql}

Constraints

Constraints [6]:

1. NOT NULL – uniemożliwia wprowadzenie do kolumny wartości pustej,

```
DROP TABLE test_null;
CREATE TABLE test_null(
    pole_not_null VARCHAR2 (4) NOT NULL,
    pole_null VARCHAR2 (4)
);
```

```
INSERT INTO test_null (pole_not_null, pole_null)
VALUES (null, null);
```

```
1  SQL Error: ORA-01400: cannot insert NULL into
```

2. UNIQUE KEY – wartość w kolumnie (lub grupie kolumn) będzie unikatowa w obrębie tabeli,

```
DROP TABLE test_unique;
```

```
CREATE TABLE test_unique(
    lastname VARCHAR2 (10),
    firstname VARCHAR2 (10),
    UNIQUE (lastname, firstname)
);
```

```
INSERT INTO test_unique(lastname, firstname)
VALUES ('Kowalski', 'Jan');

INSERT INTO test_unique(lastname, firstname)
VALUES ('Kowalski', 'Janusz');
```

```
DROP TABLE test_unique2;
```

```
CREATE TABLE test_unique2(
    lastname VARCHAR2 (10),
    firstname VARCHAR2 (10),
    UNIQUE (lastname),
    UNIQUE (firstname)
);
```

```
INSERT INTO test_unique2(lastname, firstname)
VALUES ('Kowalski', 'Jan');

INSERT INTO test_unique2(lastname, firstname)
VALUES ('Kowalski', 'Janusz');
```

```
1  SQL Error: ORA-00001: unique constraint violated
```

3. PRIMARY KEY – klucz główny:
   - może być zdefiniowany tylko jeden dla danej tabeli,

- może składać się z jednej lub większej ilości kolumn,
- nie może zawierać wartości null,
- nie może się powtarzać.

Która z poniższych instrukcji jest poprawna?

```
CREATE TABLE test_primary_key(
    nip PRIMARY KEY,
    pesel PRIMARY KEY,
    numer_indeksu PRIMARY KEY
);
```

Listing 3: {sql/test_pk1.sql}

```
SQL Error: ORA-02260: table can have only one primary key
```

```
CREATE TABLE test_primary_key2(
    nazwisko VARCHAR2 (10) ,
    imie VARCHAR2 (10) ,
    PRIMARY KEY (nazwisko, imie)
);
```

Listing 4: {sql/test_pk2.sql}

Która z poniższych instrukcji jest poprawna?

```
CREATE TABLE test_primary_key2(
    nazwisko VARCHAR2 (10) ,
    imie VARCHAR2 (10) ,
    PRIMARY KEY (nazwisko, imie)
);
```

Listing 5: {sql/test_pk2.sql}

```
INSERT INTO test_primary_key2(lastname, firstname)
VALUES('Kowalski', NULL);
```

```
SQL Error: ORA-01400: cannot insert NULL into ...
```

```
INSERT INTO test_primary_key2(lastname, firstname)
VALUES('Kowalski', 'Jan');
```

```
1 row inserted.
```

```
INSERT INTO test_primary_key2(lastname, firstname)
VALUES('Kowalski', 'Jan');
```

```
SQL Error: ORA-00001: unique constraint violated
```

4. FOREIGN KEY – klucz obcy,

   Klucz obcy [10] – Jest to kolumna lub grupa kolumn występujących w definicji tabeli i związanych ograniczeniem integralnościowym z kolumnami klucza innej tabeli.

   Klucz wskazywany (nadrzędny) [10] – Klucz unikatowy lub główny, na podstawie którego tworzony jest klucz obcy w innej tabeli.

   Tabela zależna (tabel-potomek) [10] – Tabela zawierająca klucz obcy oparty na kluczu innej tabeli.

   Tabela wskazywana (tabela-rodzic) [10] – Tabela, na której oparty jest klucz obcy w innej tabeli (klucz wskazywany tej tabeli wyznacza zbiór dopuszczalnych wartości da klucza obcego).

```
DROP TABLE students;
DROP TABLE groups;

CREATE TABLE groups(
    id NUMBER(6) CONSTRAINT group_pk PRIMARY KEY,
    name VARCHAR2(10));

INSERT INTO groups (id, name) VALUES(54, '101a');

CREATE TABLE students(
    id_student NUMBER(6) CONSTRAINT student_pk PRIMARY KEY,
    lastname VARCHAR2(10),
    id_group NUMBER(6) CONSTRAINT students_fk REFERENCES groups(id));
```

```
INSERT INTO students (id_student, lastname, id_group)
    VALUES(38, 'Kowalski', 54);
```

```
INSERT INTO student (id_student, nazwisko, id_grupa)
    VALUES(39, 'Malinowski', 45);
```

```
SQL Error: ORA-02291: integrity constraint (SYS.STUDENTS_FK) violated - parent key not found
```

```
DROP TABLE students;
DROP TABLE groups;

CREATE TABLE groups(
    id NUMBER(6) CONSTRAINT group_pk PRIMARY KEY,
    name VARCHAR2(10));

INSERT INTO groups (id, name) VALUES(54, '101a');

CREATE TABLE students(
    id_student NUMBER(6) CONSTRAINT student_pk PRIMARY KEY,
    lastname VARCHAR2(10),
    id_group NUMBER(6) CONSTRAINT students_fk REFERENCES groups(id));

INSERT INTO students (id_student, lastname, id_group)
    VALUES(38, 'Kowalski', 54);
```

```
DELETE FROM groups
WHERE id = 54;
```

```
SQL Error: ORA-02292: integrity constraint violated - child record found
```

```
DROP TABLE students;
DROP TABLE groups;

CREATE TABLE groups(
    id NUMBER(6) CONSTRAINT group_pk PRIMARY KEY,
    name VARCHAR2(10));

INSERT INTO groups (id, name) VALUES(54, '101a');

CREATE TABLE students(
    id_student NUMBER(6) CONSTRAINT student_pk PRIMARY KEY,
    lastname VARCHAR2(10),
    id_group NUMBER(6) CONSTRAINT students_fk REFERENCES groups(id)
    ON DELETE CASCADE);

INSERT INTO students (id_student, lastname, id_group)
    VALUES(38, 'Kowalski', 54);
```

```
DELETE FROM groups
WHERE id = 54;
```

```
1 row deleted
```

```
SELECT * FROM STUDENTS;
```

```
no rows selected
```

```
DROP TABLE students;
DROP TABLE groups;

CREATE TABLE groups(
    id NUMBER(6) CONSTRAINT group_pk PRIMARY KEY,
    name VARCHAR2(10));

INSERT INTO groups (id, name) VALUES(54, '101a');

CREATE TABLE students(
    id_student NUMBER(6) CONSTRAINT student_pk PRIMARY KEY,
    lastname VARCHAR2(10),
    id_group NUMBER(6) CONSTRAINT students_fk REFERENCES groups(id)
    ON DELETE SET NULL);

INSERT INTO students (id_student, lastname, id_group)
    VALUES(38, 'Kowalski', 54);
```

```
DELETE FROM groups
WHERE id = 54;
```

```
1 row deleted
```

```
SELECT * FROM STUDENTS;
```

```
ID_STUDENT LASTNAME   ID_GROUP
---------- ---------- ----------
```

```
3          38 Kowalski
```

```
DROP TABLE employees;
CREATE TABLE employees(
    lastname VARCHAR2 (10) PRIMARY KEY,
    boss VARCHAR2 (10) REFERENCES employees(lastname)
);

INSERT INTO employees(lastname, boss)
VALUES('director', null);

INSERT INTO employees(lastname, boss)
VALUES('manager', 'director');

INSERT INTO employees(lastname, boss)
VALUES('worker', 'manager');

SELECT * FROM employees;
```

```
1 LASTNAME   BOSS
2 ---------- ----------
3 director
4 manager    director
5 worker     manager
```

5. Złożone testy integralności (CHECK),

```
CREATE TABLE czlowiek (
    oczy VARCHAR2(10) DEFAULT 'NIEBIESKIE' CONSTRAINT oczy_check
        CHECK (oczy IN ('NIEBIESKIE', 'PIWNE', 'ZIELONE')),
    wzrost NUMBER(3,2) CONSTRAINT wzrost_check
        CHECK (wzrost > 0 AND wzrost < 3)
);
```

Listing 6: {sql/test_check.sql}

```
INSERT INTO people(eyes, height)
VALUES ('GREEN', 2);
```

6. wyzwalacze [6] – jeden z nazwanych bloków, który jest uruchamiany w momencie wystąpienia zdarzenia wyzwalającego.

6 Tworzenie tabeli na podstawie innej tabeli (CREATE TABLE AS)

```
CREATE TABLE table_name AS
    SELECT_STATEMENT;
```

```
CREATE TABLE prymus AS
    SELECT * FROM student WHERE srednia > 4.5;
```

Listing 7: {sql/test_create3.sql}

7 Usuwanie tabel (DROP TABLE)

Instrukcja DROP TABLE służy do usuwania tabel.

Źródło: SQL Language Reference
`https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/DROP-TABLE.html`

drop_table

```
DROP TABLE [ schema. ] table
  [ CASCADE CONSTRAINTS ] [ PURGE ] ;
```

Uproszczona składnia DROP TABLE:

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

Czym różnią się wyniki wykonania poniższych instrukcji?

```
CREATE TABLE groups(
    id NUMBER(6) CONSTRAINT group_pk PRIMARY KEY,
    name VARCHAR2(10));

INSERT INTO groups (id, name) VALUES(54, '101a');

CREATE TABLE students(
    id_student NUMBER(6) CONSTRAINT student_pk PRIMARY KEY,
    lastname VARCHAR2(10),
    id_group NUMBER(6) CONSTRAINT students_fk REFERENCES groups(id));

INSERT INTO students (id_student, lastname, id_group)
    VALUES(38, 'Kowalski', 54);
```

```
DROP TABLE students;
DROP TABLE groups;
```

```
DROP TABLE groups;
DROP TABLE students;
```

```
SQL Error: ORA-02449: unique/primary keys in table referenced by foreign keys
```

```
SELECT * FROM students;
```

```
ID_STUDENT LASTNAME   ID_GROUP
---------- ---------- ----------
        38 Kowalski           54
```

```
SELECT constraint_name FROM user_constraints WHERE table_name = 'STUDENTS';
```

```
CONSTRAINT_NAME
------------------------------

STUDENT_PK
STUDENT_FK
```

```
DROP TABLE groups cascade constraints;
```

```
Table GROUPS dropped.
```

```
SELECT * FROM students;
```

```
ID_STUDENT LASTNAME   ID_GROUP
---------- ---------- ----------
        38 Kowalski           54
```

```
SELECT constraint_name FROM user_constraints WHERE table_name = 'STUDENTS';
```

```
CONSTRAINT_NAME
------------------------------

STUDENT_PK
```

8 Modyfikowanie struktury tabel (ALTER TABLE)

Instrukcja ALTER TABLE służy modyfikowania struktury tabeli.

Źródło: SQL Language Reference
`https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/ALTER-TABLE.html`

alter_table

```
ALTER TABLE [ schema. ] table
  [ memoptimize_read_clause ]
  [ alter_table_properties
  | column_clauses
  | constraint_clauses
  | alter_table_partitioning
  | alter_external_table
  | move_table_clause
  | modify_to_partitioned
  | modify_opaque_type
  ]
  [ enable_disable_clause
  | { ENABLE | DISABLE }
    { TABLE LOCK | ALL TRIGGERS | CONTAINER_MAP | CONTAINERS_DEFAULT }
  ] ...
  ;
```

column_clauses

```
{ { add_column_clause
  | modify_column_clauses
  | drop_column_clause
  | add_period_clause
  | drop_period_clause
  }...
| rename_column_clause
| { modify_collection_retrieval }...
| { modify_LOB_storage_clause }...
| { alter_varray_col_properties }...
}
```

add_column_clause

```
ADD
   ( {column_definition | virtual_column_definition
      [, column_definition | virtual_column_definition] ...
     } )
   [ column_properties ]
   [ ( out_of_line_part_storage [, out_of_line_part_storage]... ) ]
```

modify_column_clauses

```
MODIFY
{ ( modify_col_properties | modify_virtcol_properties
    [, modify_col_properties | modify_virtcol_properties ]... )
| ( modify_col_visibility [, modify_col_visibility ]... )
| modify_col_substitutable
}
```

rename_column_clause

```
RENAME COLUMN old_name TO new_name
```

drop_column_clause

```
{ SET UNUSED { COLUMN column
             | (column [, column ]...)
             }
  [ { CASCADE CONSTRAINTS | INVALIDATE }... ]
  [ ONLINE ]
| DROP { COLUMN column
       | (column [, column ]...)
       }
  [ { CASCADE CONSTRAINTS | INVALIDATE }... ]
  [ CHECKPOINT integer ]
| DROP { UNUSED COLUMNS
       | COLUMNS CONTINUE
       }
  [ CHECKPOINT integer ]
}
```

constraint_clauses

```
{ ADD { { out_of_line_constraint }...
      | out_of_line_REF_constraint
      }
| MODIFY { CONSTRAINT constraint_name
         | PRIMARY KEY
         | UNIQUE (column [, column ]...)
         } constraint_state [ CASCADE ]
| RENAME CONSTRAINT old_name TO new_name
| { drop_constraint_clause }...
}
```

enable_disable_clause

```
{ ENABLE | DISABLE }
[ VALIDATE | NOVALIDATE ]
{ UNIQUE (column [, column ]...)
| PRIMARY KEY
| CONSTRAINT constraint_name
}
[ using_index_clause ]
[ exceptions_clause ]
[ CASCADE ]
[ { KEEP | DROP } INDEX ]
```

Wybrane warianty instrukcji ALTER TABLE:

alter

```
ALTER TABLE table_name ADD (column_name datatype [constraint]);
ALTER TABLE table_name DROP (column_name);
ALTER TABLE table_name MODIFY (column_name datatype);
ALTER TABLE table_name ADD constraint (column_name);
```

Uwaga! Nie zawsze wszystkie modyfikacje są możliwe. Niektóre mogą być wykonywane tylko na pustych tabelach.

```
ALTER TABLE student
    ADD (drugie_imie VARCHAR2(30)
    CONSTRAINT student_drugie_imie_nn NOT NULL );

ALTER TABLE student DROP (drugie_imie);
ALTER TABLE student
    MODIFY (nazwisko VARCHAR2(30)
    CONSTRAINT student_nazwisko_nn NOT NULL );
ALTER TABLE student
    ADD UNIQUE(nazwisko);
ALTER TABLE grupa ENABLE CONSTRAINT grupa_pk;
ALTER TABLE student DISABLE PRIMARY KEY;
```

Listing 8: {sql/test_alter.sql}

Literatura

[1] Jeffrey D. Ullman, Jennifer Widom, "Podstawowy wykład z systemów baz danych", WNT, 2001

[2] Jeffrey D. Ullman, Jennifer Widom, "Systemy baz danych, Pełny wykład", WNT, 2006

[3] Thomas Connolly, Carolyn Begg, "Systemy baz danych", Wydawnictwo RM, 2004

[4] Thomas Connolly, Carolyn Begg, "Database systems", Peason, 2015

[5] Kevin Loney, Bob Bryla, Oracle Database 10g, Podręcznik administratora baz danych. Helion 2008

[6] Scott Urman, Ron Hardman, Michael McLaughlin, "Oracle Database 10g, Programowanie w języku PL/SQL", Helion, 2008

[7] Price J, "Oracle Database 11g i SQL Programowanie", Helion, 2009

[8] Ramez Elmasri, Shamkant B. Navathe, "Wprowadzeni do systemów baz danych", Helion, 2005

[9] Sharon Allen, "Modelowanie danych", Helion, 2006

[10] Jose A. Ramalho, Oracle 8i, Mikom, 2001

[11] Dokumentacja do bazy danych Oracle dostępna na stronie www.oracle.com

[12] Sadalage P. J., Fowler M., "NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence", Addison-Wesley Professional, 2012

[13] Michael Stonebraker,"New Sql: An Alternative to Nosql and Old Sql For New Oltp Apps", June 16, 2011

[14] Oracle Database PL/SQL Language Reference 11g Release 1 (11.1) B28370-02 (B28370.pdf)

Materiały do przedmiotu dostępne są na stronie: http://weaii-moodle.tu.kielce.pl/