

Based on:

Thomas Connolly, Carolyn Begg, Systemy baz danych, Wydawnictwo RM, 2004

1 Physical Database Design for Relational Databases

Physical Database Design [3] – "The process of producing a description of the implementation of the database on secondary storage; it describes the base relations, file organizations, and indexes used to achieve efficient access to the data, and any associated integrity constraints and security measures."

Step 4

Translate logical data model for target DBMS

Step 4.1

Design database relations

Step 4.2

Design representation of derived data

Step 4.3

Design general constraints

Step 5

Design file organizations and indexes

Step 5.1

Analyze transactions

Step 5.2

Choose file organizations

Step 5.3

Choose indexes

Step 5.4

Estimate disk space requirements

Step 6

Design users views

Step 7

Design security mechanisms

Step 8

Consider the introduction of controlled redundancy

Step 8.1

Combining one-to-one (1:1:) relationships

Step 8.2

Duplicating non-key attributes in one-to-many (1:*) relationships to reduce joins

Step 8.3

Duplicating foreign key attributes in one-to-many (1:*) relationships to reduce joins

Step 8.4

Duplicating attributes in many-to-many (*:*) relationships to reduce joins

Step 8.5

Introducing repeating groups

Step 8.6

Creating extract tables

Step 8.7

Partitioning relations

Step 9

Monitor and tune the operational system

2 Definitions

2.1 Relational model

The relational model [3] provides only one way to represent data: it is a two-dimensional table called a relation. Attributes are provided in the relation header.

Relation lines, except for the header line, which contains relation attributes, are called tuples.

id	lastname	firstname	group_id
99001	AAAA	ZZZZ	211A
99002	BBBB	YYYY	211A
99003	CCCC	XXXX	211B
99004	DDDD	WWWW	211B

The relation name with set of attributes are called the relationship schema. We write the relationship schema by its name followed by a list of attributes enclosed in parentheses, e.g.

<code>Students(id, lastname, firstname, group_id)</code>
--

The set of relation schemas in the project is called a relational database schema or simply a database schema.

It is assumed that each attribute of a relation has an associated domain, some specific elementary type. Each member of each tuple in the relationship has a value that belongs to the domain of the corresponding column.

In the relational model, each element of each relation must have a specific atomic type, i.e. its type must belong to elementary types, e.g. it must be integral or character. The value of an attribute cannot be a record, a list, an array, a collection, or any other structure that can be broken down into smaller parts.

Equivalents for the concepts of relational model [2]:

Formal name	First equivalent	Second equivalent
Relation	Table	File
Tuple	Row	Record
Attribute	Column	Field

2.2 SQL (Structured Query Language)

Structured Query Language (SQL) is used to execute operations on relational databases. There are two basic parts of SQL:

- Data Definition Language (DDL) used to define data structures and data access control,
- Data Manipulation Language (DML) used to search and modify data.

3 Oracle data types

Source:

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

Selected Oracle data types	Description
VARCHAR2 (size [BYTE CHAR])	Variable-length character string with length from 1 to maximal size of bytes or characters. Maximal size is: 32767 bytes or characters if MAX_STRING_SIZE = EXTENDED, 4000 bytes or characters if MAX_STRING_SIZE = STANDARD
NVARCHAR2(size)	Variable-length Unicode character string with length from 1 to maximal size of characters. The number of bytes can be up to two times size for AL16UTF16 encoding and three times size for UTF8 encoding. Maximal size is: 32767 bytes if MAX_STRING_SIZE = EXTENDED, 4000 bytes if MAX_STRING_SIZE = STANDARD
NUMBER [(p[,s])]	Number having precision p from 1 to 38 and scale s from -84 to 127.
FLOAT [(p)]	A subtype of the NUMBER data type having precision p from 1 to 126 binary digits.
LONG	Character data of variable length up to 2 gigabytes, or $2^{31} - 1$ bytes.
DATE	Valid date range from January 1, 4712 BC, to December 31, 9999 AD.

BINARY_FLOAT	32-bit floating point number.
BINARY_DOUBLE	64-bit floating point number.
TIMESTAMP [(fractional_seconds_precision)]	Timestamp
TIMESTAMP [(fractional_seconds_precision)] WITH TIME ZONE	Timestamp with time zone
TIMESTAMP [(fractional_seconds_precision)] WITH LOCAL TIME ZONE	Timestamp with local time zone
INTERVAL YEAR [(year_precision)] TO MONTH	Stores a period of time in years and months.
INTERVAL DAY [(day_precision)] TO SECOND [(fractional_seconds_precision)]	Stores a period of time in days, hours, minutes, and seconds.
RAW(size)	Raw binary data. Maximum size is: 32767 bytes if MAX_STRING_SIZE = EXTENDED, 2000 bytes if MAX_STRING_SIZE = STANDARD.
LONG RAW	Raw binary data of variable length up to 2 gigabytes.
ROWID	String representing the unique address of a row in its table.
UROWID [(size)]	String representing the logical address of a row of an index-organized table.
CHAR [(size [BYTE CHAR])]	Fixed-length character data of length from 1 to 2000 bytes or characters.
NCHAR[(size)]	Fixed-length character data of length size characters. The number of bytes can be up to two times size for AL16UTF16 encoding and three times size for UTF8 encoding. Maximum size is determined by the national character set definition, with an upper limit of 2000 bytes.
CLOB	A character large object containing single-byte or multibyte characters. Maximum size is (4 gigabytes - 1) * (database block size).
NCLOB	A character large object containing Unicode characters. Maximum size is (4 gigabytes - 1) * (database block size).
BLOB	A binary large object. Maximum size is (4 gigabytes - 1) * (database block size).
BFILE	Contains a locator to a large binary file stored outside the database. Maximum size is 4 gigabytes.

4 SQL statements

Below the basic syntax for selected SQL statements is presented. Full specifications are available in the Oracle documentation.

4.1 CREATE TABLE

The CREATE TABLE statement is used to create tables.

Source: 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 } ]
```

Simplified syntax for the CREATE TABLE statement:

```
CREATE TABLE table_name (  
    column_name data_type [DEFAULT expression] [column_constraints]  
    {, column_name data_type [DEFAULT expression] [column_constraints]}  
    [table_constraints]  
);
```

Examples:

```
CREATE TABLE groups(  
    id_group NUMBER(6) CONSTRAINT group_pk PRIMARY KEY,  
    name VARCHAR2(10));  
  
CREATE TABLE students(  
    id_student NUMBER(6) CONSTRAINT student_pk PRIMARY KEY,  
    lastname VARCHAR2(10),  
    firstname VARCHAR2(10),  
    mark_avg NUMBER (3,2),  
    id_group NUMBER(6) CONSTRAINT students_fk REFERENCES groups(id_group));
```

Listing 1: {sql/test_create1.sql}

```
CREATE TABLE groups(  
    id_group NUMBER(6),  
    name VARCHAR2(10),  
    CONSTRAINT group_pk PRIMARY KEY (id_group));  
  
CREATE TABLE students(  
    id_student NUMBER(6) CONSTRAINT student_pk PRIMARY KEY,  
    lastname VARCHAR2(10),  
    firstname VARCHAR2(10),  
    mark_avg NUMBER (3,2),  
    id_group NUMBER(6),  
    CONSTRAINT students_fk FOREIGN KEY (id_group) REFERENCES groups(id_group));
```

Listing 2: {sql/test_create2.sql}

Constraints

Constraints [5]:

1. NOT NULL – prevents entering of an empty value into the column,
2. UNIQUE KEY – value in a column (or group of columns) must be unique within the table,
3. PRIMARY KEY :
 - can be only one defined for a table,
 - may consist of one or more columns,
 - can not contain null values,
 - cannot be repeated.

Which of the following instructions is correct?

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

Listing 3: {sql/test_pk1.sql}

```
CREATE TABLE test_primary_key2(  
    lastname VARCHAR2 (10) ,  
    firstname VARCHAR2 (10) ,  
    PRIMARY KEY (lastname, firstname)  
);
```

Listing 4: {sql/test_pk2.sql}

4. FOREIGN KEY

Foreign Key [9] - a column or group of columns that occur in a table definition and are associated with an integrity constraint with a key columns of another table.

5. complex integrity tests (CHECK),

```
CREATE TABLE people (  
  eyes VARCHAR2(10) DEFAULT 'BLUE' CONSTRAINT eyes_check  
  CHECK (eyes IN ('BLUE', 'BROWN', 'GREEN')),  
  height NUMBER(3,2) CONSTRAINT height_check  
  CHECK (height > 0 AND height < 3)  
);
```

Listing 5: {sql/test_check.sql}

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

6. triggers [5] - one of the named blocks that are executed when a trigger event occurs.

4.2 CREATE TABLE AS

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

```
CREATE TABLE the_best_students AS  
SELECT * FROM students WHERE mark_avg > 4.5;
```

Listing 6: {sql/test_create3.sql}

4.3 DROP TABLE

The DROP TABLE statement is used to remove tables.

Source: 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 ] ;
```

Simplified syntax for the DROP TABLE statement:

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

4.4 ALTER TABLE

The ALTER TABLE statement is used to modify the table structure.

Source: 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 ]
```

Selected variants of the ALTER TABLE statement:

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);
```

Warning! Not all modifications are always possible. Some can only be performed on empty tables.

```
ALTER TABLE students  
  ADD (second_name VARCHAR2(30)  
  CONSTRAINT students_second_name_nn NOT NULL );  
ALTER TABLE students DROP (second_name);  
ALTER TABLE students  
  MODIFY (lastname VARCHAR2(30)  
  CONSTRAINT student_lastname_nn NOT NULL );  
ALTER TABLE students  
  ADD UNIQUE(lastname);  
ALTER TABLE groups ENABLE CONSTRAINT group_pk;  
ALTER TABLE students DISABLE PRIMARY KEY;
```

Listing 7: {sql/test_alter.sql}

References

- [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] Kevin Loney, Bob Bryla, Oracle Database 10g, Podręcznik administratora baz danych. Helion 2008
- [5] Scott Urman, Ron Hardman, Michael McLaughlin, "Oracle Database 10g, Programowanie w języku PL/SQL", Helion, 2008
- [6] Price J, "Oracle Database 11g i SQL Programowanie", Helion, 2009
- [7] Ramez Elmasri, Shamkant B. Navathe, "Wprowadzeni do systemów baz danych", Helion, 2005
- [8] Sharon Allen, "Modelowanie danych", Helion, 2006
- [9] Jose A. Ramalho, Oracle 8i, Mikom, 2001
- [10] Dokumentacja do bazy danych Oracle dostępna na stronie www.oracle.com
- [11] Sadalage P. J., Fowler M., "NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence", Addison-Wesley Professional, 2012
- [12] Michael Stonebraker, "New Sql: An Alternative to Nosql and Old Sql For New Oltp Apps", June 16, 2011

Materials for the subject are available at: <http://weaii-moodle.tu.kielce.pl/>