```
CREATE USER user_name
IDENTIFIED {BY password | EXTERNALLY
                        | GLOBALLY AS 'CN=user' }
[ DEFAULT TABLESPACE tablespace]
[ TEMPORARY TABLESPACE tablespace]
[ QUOTA { number [K|M] | UNLIMITED } ON tablespace ]
  [, QUOTA { number [K|M] | UNLIMITED } ON tablespace ]
[ PROFILE profile_name ]
[ PASSWORD EXPIRE ]
[ { ACCOUNT LOCK | ACCOUNT UNLOCK} ]
```

```
CREATE TABLE [schema.] table_name
[
  ( { column_name datatype [DEFAULT expr]
      [{ [column_constraint] }[...]
      |
      table_constraint
      }][...] )

column_constraint ::=
[CONSTRAINT constraint_name]
{
[NOT] NULL
|
{UNIQUE | PRIMARY KEY}
|
REFERENCES [schema.] table_name [ (column_name1
                                   [,column_name2, ...]  ) ]
 [ON DELETE CASCADE ]
|
CHECK (condition)
}

table_constraint ::=
[CONSTRAINT constraint_name]
{
{ UNIQUE | PRIMARY KEY} ( { column_name1} [,column_name2,
                              ...] )
|
FOREIGN KEY (column_name1 [,column_name2, ...]  )
       REFERENCES [schema.] table_name
 [ column_name1 [,column_name2, ...]] [ON DELETE CASCADE ]
|
CHECK (condition)
}
```

```
DROP TABLE table_name [ CASCADE CONSTRAINTS | PURGE ];
```

```
PURGE TABLE table_name;
PURGE INDEX index_name;
PURGE RECYCLEBIN;
PURGE TABLESPACE tablespace_name;
PURGE TABLESPACE tablespace_name USER user_name;
```

```
FLASHBACK TABLE table_name TO BEFORE DROP;
FLASHBACK TABLE table_name TO BEFORE DROP
                RENAME TO new_table_name;
```

```
CREATE [UNIQUE] INDEX index_name
 ON table_name (column_name1 [ASC | DESC],...);
```

```
DROP INDEX index_name;
```

```
CREATE SEQUENCE sequence_name
  [ INCREMENT BY integer_value ]
  [ START WITH integer_value]
  [ {MAXVALUE integer_value | NOMAXVALUE }]
  [ {MINVALUE integer_value | NOMINVALUE }]
  [ {CYCLE | NOCYCLE}]
  [ {CACHE positive_integer_value | NOCACHE }]
  [ {ORDER | NOORDER }];
```

```
ALTER SEQUENCE sequence_name INCREMENT BY integer_value;
ALTER SEQUENCE sequence_name MAXVALUE integer_value;
ALTER SEQUENCE sequence_name {CYCLE | NOCYCLE };
ALTER SEQUENCE sequence_name
                {CACHE positive_integer_value | NOCACHE};
ALTER SEQUENCE sequence_name {ORDER | NOORDER};
```

```
DROP SEQUENCE sequence_name;
```

```
schema_name.object_name@dblink_name
schema_name.object_name
object_name
```

```
INSERT INTO table_name [(column_list)]
{
   VALUES (list_of_values)
|
   SELECT-statement
}
```

```
DELETE FROM table_name
 [WHERE conditions];
```

```
UPDATE table_name SET
{
   column_name1 = expression1[,...]
   |
   {    (column_list)
        |
        *
   } = (expression_list)
}
[WHERE conditions]
```

```
SELECT [ALL | DISTINCT]
   { *| column_name1| function_name1[(parameters1)]} [,...]
   FROM table_reference1 [table_alias1] [,...]
   [WHERE conditions]
   [GROUP BY column_list]
   [HAVING conditions]
   [ORDER BY column_list [ASC | DESC],...]
```

```
   FROM table_name1 [table_alias1]
  {  [{LEFT|RIGHT|FULL}[OUTER]] JOIN table_name2
                              [table_alias2]
        { ON (join_conditions1)| USING(column_list_join1)}
    |
     [INNER] JOIN table_name3 [table_alias3]
        { ON (join_conditions3)| USING(column_list_join3)}

    | {CROSS | NATURAL [INNER]} JOIN table_name4
                                        [table_alias4] }
```

```
vyraz relacny-operator vyraz
vyraz [NOT] BETWEEN vyraz AND vyraz
vyraz [NOT] IN (polozky)
meno_stlpca [NOT] LIKE 'string' [ESCAPE escape-znak]
vyraz relacny-operator {ALL | [ANY | SOME]} (SELECT-prikaz)
vyraz [NOT] IN (SELECT-prikaz)
vyraz [NOT] EXISTS (SELECT-prikaz)
meno_stlpca IS [NOT] NULL
```

```
  SELECT-statement1
  {UNION [ALL] | INTERSECT | MINUS }
  SELECT-statement2
     [ {UNION [ALL] | INTERSECT | MINUS }
         SELECT-statement3
     ] ...
```

```
SUBSTR(string, m [,n])
LENGTH(string)
UPPER(string)
LOWER(string)
INITCAP(string)
Operátor ||
CONCAT(string1, string2)
INSTR(string,substring,[m[,n]])
```

```
LIKE '%\_%' ESCAPE '\';
%  ľubovoľný počet znakov
_  jeden znak
```

```
ABS(expression)
ROUND(n [,m])
TRUNC(n [,m])
```

```
ALTER SESSION
   SET nls_date_format='DD.MM.YYYY HH24:MI:SS';
ALTER SESSION
   SET nls_timestamp_format='DD.MM.YYYY HH24:MI:SS:FF';
ALTER SESSION
   SET nls_date_language='English';
ALTER SESSION
   SET nls_territory='Slovakia';
        -- 1 (číslo dňa) - pondelok
ALTER SESSION
   SET nls_territory= 'America';
        -- 1 (číslo dňa) - nedeľa
```

```
TO_CHAR(date_value, [format [, nls_param] ])
TO_DATE(string_value, [format [, nls_param]])
```

```
SYSDATE
SYSTIMESTAMP
```

```
ADD_MONTHS(d,n)
NEXT_DAY(d, day_value)
LAST_DAY(d)
TRUNC(d [, format])
ROUND(d [, format])
EXTRACT(format FROM d)
MONTHS_BETWEEN(d1, d2)
```

```
COALESCE(expr1, expr2, ..., exprn)
DECODE(expression, if1, then1 [, ifn, thenn] [,else])
NVL(expression1, expression2)
NVL2(expression1, expression2, expression3)
```

```
    case expression
         when value1 then result1
         [when valuen then valuen] [...]
         [else result]
    end
```

```
    case
         when condition1 then result1
         [when conditionn then resultn] [...]
         [else result]
    end
```

```
ROWID
```

```
USER
```

```
row_number() over ( [ partition by vyraz ]
                          ORDER BY  zoznam_stlpcov )
rank() over ( [ partition by vyraz ]
                          ORDER BY  zoznam_stlpcov )
dense_rank() over ( [ partition by vyraz ]
                          ORDER BY  zoznam_stlpcov )
```

```
GRANT database_privilege_list
 TO {PUBLIC | list_of_users}
  [WITH ADMIN OPTION]
```

```
GRANT object_privilege_list ON object_name
  TO {PUBLIC | list_of_users }
   [WITH GRANT OPTION]
```

```
REVOKE { privilege_name ON object_name
             |
         database_privilege_name
               |
         role_name}
 FROM {PUBLIC | list_of_users }
```

```
CREATE ROLE role_name;
```

```
BEGIN WORK
COMMIT [WORK]
ROLLBACK [WORK]
SAVEPOINT savepoint_name
ROLLBACK TO SAVEPOINT savepoint_name
```

```
IF condition THEN
  statements;
END IF;
```

```
IF condition1 THEN
  statements;
ELSIF condition2 THEN
  statements;
[ELSE
  statements;]
END IF;
```

```
IF condition THEN
  statements;
[ELSE
  statements;]
END IF;
```

```
LOOP
  ...
  IF condition THEN
    EXIT;
  END IF;
  ...
END LOOP;
```

```
LOOP
  ...
  EXIT WHEN condition;
END LOOP;
```

```
WHILE condition LOOP
  statements;
END LOOP;
```

```
FOR variable_name IN min..max LOOP
  statements;
END LOOP;
```

```
FOR variable_name IN REVERSE min..max LOOP
  statements;
END LOOP;
```

```
[DECLARE              -- cast deklaracii
   variable_name  data_type[:= init_value];
]
BEGIN
  statements;     -- cast prikazov
[EXCEPTION     -- cast odchytenia a spracovania vynimiek
  WHEN  exception_type1 THEN
    statements;
  WHEN exception_type2 THEN
    statements;
  ...
]
END;
/
```

```
CREATE [OR REPLACE] PROCEDURE procedure_name
 [( parameter1 [ mode1] data_type1,
    parameter2 [ mode2] data_type2, . . .)]
IS|AS
  [ variable_name  data_type[:= init_value]; ]
BEGIN
  statements;
  [ EXCEPTION
      WHEN exception_type1 THEN
         statements;
      [WHEN ...]
  ]
END  [procedure_name];
/
```

```
CREATE [OR REPLACE] FUNCTION function_name
 [( parameter1 [ mode1] datatype1,
    parameter2 [ mode2] datatype2, . . .)]
RETURN datatype
IS|AS
  [ variable_name  data_type[:= init_value]; ]
BEGIN
  statements;
  RETURN expression;
[ EXCEPTION
      WHEN exception_type1 THEN
         statements;
      [WHEN ...]
  ]
END  [function_name];
/
```

```
DROP PROCEDURE procedure_name;
```

```
DROP FUNCTION function_name;
```

```
RAISE_APPLICATION_ERROR
            (error_code, error_text[, {TRUE | FALSE} ]);
```

```
CREATE  [OR REPLACE]  TRIGGER [schema.] trigger
{ {BEFORE | AFTER }
  {DELETE | INSERT | UPDATE [ OF column1 [, column2 [,…]]]}
  [ OR {DELETE | INSERT | UPDATE [ OF column1 [, column2
                                          [,…] ] ] }] […]
|
 INSTEAD OF {DELETE | INSERT | UPDATE }}
ON [schema.] [table_name | view_name]
[ REFERENCING { OLD [AS] stary | NEW [AS] novy}]
[ FOR EACH ROW ]
[ WHEN (condition)]
   Ttrigger_body
```

```
ALTER TRIGGER [schema.] trigger_name   {ENABLE | DISABLE};
```

```
ALTER TABLE [schema.] table_name   {ENABLE | DISABLE}
   ALL TRIGGERS;
```

```
DROP TRIGGER  [schema.] trigger_name ;
```

```
CREATE  [OR REPLACE]  [ FORCE  |  NOFORCE ]
 VIEW  [schema.] view_name  [(column_alias1 [,…])]
  AS Select_statements
  [WITH  [  READ ONLY  |
            CHECK OPTION  [CONSTRAINT constraint_def] ] ]
```

```
SELECT zoznam_stlpcov
   INTO zoznam_premennych
 FROM table_list
```

```
  SELECT expr1,    expr2 ...,    exprn
     BULK COLLECT INTO var1,    var2 ..,    varn
  FROM table_list
```

```
OPEN cursor_name;
FETCH cursor_name INTO list_of_variables;
CLOSE cursor_name;

cursor_name%ISOPEN
cursor_name%FOUND
cursor_name%NOTFOUND
cursor_name%ROWCOUNT
```