8. SQL sakiniai taikomosiose programose

SQL gali būti vartojama dviem režimais:

- **interaktyviai**
- **taikomosiose programose**

Dvilypumo **privalumai**:

- interaktyvaus režimo prieinamos ir programose;
- SQL derinamas interaktyviai ir keliamas į programas.

SQL **nėra pilnavertė programavimo kalba**.

SQL tik **papildo bazinę programavimo kalbą**:

- SQL sakiniai programose – sąsaja su DB
- Programavimo kalba – patogus veiksmų valdymas.

SQL vartojimo programose būdai:

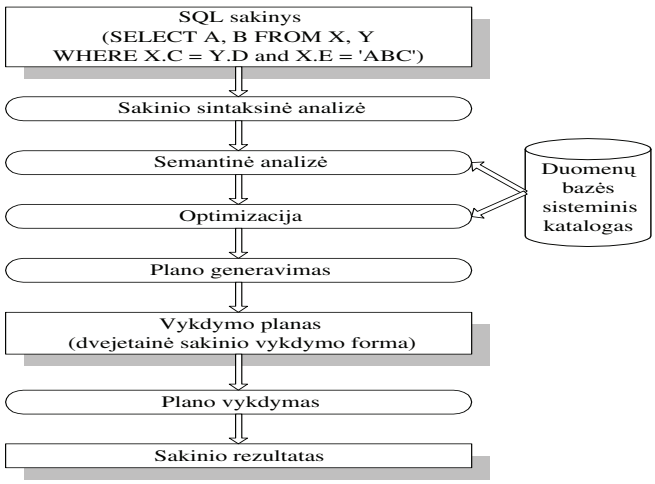
- **Programų SQL** (angl. ***embedded SQL***). Prieš kompiliuojant programą tekstas apdorojamas specialiu SQL preprocesoriumi.
  SQL2 apibrėžia SQL programavimo kalboms:
  Ada, C, COBOL, FORTRAN, Pascal ir PL/1.
- **Taikomųjų programų sąsaja** (angl. *application program interface* – API) – DBVS funkcijų rinkinys.

  TP sąsajos realizacijos:
  - ODBC (**Open DataBase Connectivity**).
  - JDBC (**Java DataBase Connectivity**).

8.1. SQL sakinio vykdymo etapai

SQL sakinių vykdyme skiriami 5 pagrindiniai etapai:

1. **Sintaksinė analizė**.
2. **Semantinė analizė.**
3. **Optimizacija.**
4. **Plano generavimas.**
5. **Plano vykdymas.**



Interaktyviame režime:
– visi 5 etapai vykdomi nuosekliai;
– DBVS SQL sakinius interpretuoja.

Programose:
dalis etapų gali būti atlikti kompiliuojant.

8.2. Programų SQL ypatybės

SQL ir bazinės programavimo kalbos sakinių bendro naudojimo principai:

- SQL sakiniai rašomi tarp programavimo kalbos sakinių. Programa yra apdorojama SQL preprocesoriumi.
- SQL sakiniuose galima naudoti programos kintamuosius.
- SQL sakinio rezultatas perduodamas programai per kintamuosius, apibrėžtus bazine programavimo kalba.
- `NULL` reikšmei naudojami programos kintamieji.
- Nuosekliam užklausos rezultato perrinkimui yra naudojami papildomi SQL sakiniai, kurių nėra interaktyviame SQL.

SQL sakinių užrašymo programose taisyklės priklauso nuo bazinės programavimo kalbos.

C programose:

```
EXEC SQL <SQL sakinys> ;
```

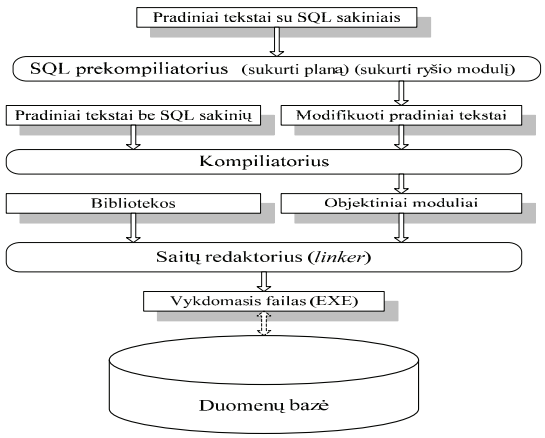
Programų SQL sakiniai yra skirstomi į

– **statinius**

– **dinaminius**

SQL sakinys yra statinis ar dinaminis priklauso nuo galimybės sugeneruoti vykdymo planą preprocesavimo (kompiliavimo) metu.

8.3. Programos parengimo schema



8.4. Paprastų statinių SQL sakinių vartojimas

```
EXEC SQL INSERT INTO Vykdytojai
     VALUES (6, 'Baltakis', 'Informatikas', 2, NULL) ;

EXEC SQL DELETE FROM Vykdytojai
         WHERE Pavardė = 'Baltakis';

EXEC SQL UPDATE Projektai
         SET Terminas = Terminas * 1.1
WHERE Projektai.Nr IN
  (SELECT Projektas FROM Vykdymas,Vykdytojai
   WHERE Vykdytojas = Vykdytojai.Nr AND
         Pavardė = 'Baltakis') ;
```


Klaidų apdorojimas

Klaidų tipai:

- **Kompiliacijos klaidos**.
- **Vykdymo klaidos**.

DBVS klaidos kodą perduoda specialiu kintamuoju `SQLCODE`.

SQL ryšio sritis (SQL Communication Area):

```
EXEC SQL INCLUDE SQLCA;
```

SQL preprocesorius įtraukia eilutes:

```
#include <sqlca.h>
struct sqlca sqlca ;
```


sqlca.h

```
struct sqlca
{
   ...
   long sqlcode;    /* SQL return code */
   ...
};

#define SQLCODE sqlca.sqlcode
```

`SQLCODE` reikšmės:

- 0 – SQL sakinys įvykdytas sėkmingai.
- < 0 – rimta klaida, dėl kurios sakinys neįvykdytas;
- > 0 – ypatinga situacija, pvz. 100 – „nėra duomenų“.


```
EXEC SQL DELETE FROM Vykdytojai
         WHERE Pavardė = 'Baltakis';
if (0 > SQLCODE)
  printf("Įvyko klaida. Kodas: %ld\n", SQLCODE) ;
else if (0 == SQLCODE)
  printf("Duomenys pašalinti sėkmingai\n") ;
else if (100 == SQLCODE)
  printf("Šalintinų duomenų nerasta\n") ;
```


Ypatingų situacijų tvarkymo SQL sakinys `WHENEVER`

```
WHENEVER NOT FOUND | SQLERROR | SQLWARNING
         CONTINUE | GO TO <programos žymė>

EXEC SQL WHENEVER SQLERROR GOTO error;
 – atitinka:
if (0 > SQLCODE) GOTO error;

EXEC SQL WHENEVER NOT FOUND GOTO end;
 – atitinka:
if (100 == SQLCODE) GOTO error;

EXEC SQL WHENEVER SQLERROR CONTINUE;
                          – atšaukia „goto“
```


Bazinis kintamasis tai – programos kintamasis, apibrėžiamas bazine programavimo kalba ir naudojamas SQL sakinyje.

Bazinius kintamuosius **galima naudoti ten**, kur SQL konstantas.

```
EXEC SQL BEGIN DECLARE SECTION;
  long nr ;
EXEC SQL END DECLARE SECTION;
  nr = 1;
EXEC SQL DELETE FROM Vykdytojai WHERE Nr = :nr ;
```


SQL ir C duomenų tipų suderinamumas:

```
INTEGER  – long
SMALLINT – short
CHAR(n)  – char[n+1]
DATE     – char[11]
TIME     – char[9]
```

`NULL` realizuojama **kintamuoju–indikatoriumi**.

- neigiama kintamojo–indikatoriaus reikšmė – `NULL`.
- neneigiama kintamojo–indikatoriaus reikšmė – baziniame kintamajame yra tikroji reikšmė.

```
EXEC SQL UPDATE Vykdytojai
     SET Kategorija = :value:ind WHERE Nr = 1;
```


Kintamojo–indikatoriaus **negalima** vartoti paieškos sąlygoje:

```
EXEC SQL UPDATE Vykdytojai
  SET Kategorija = COALESCE(Kategorija, 0) + 1
  WHERE Išsilavinimas = :value :ind ;
⇒
if(ind < 0)
   EXEC SQL UPDATE Vykdytojai
      SET Kategorija = COALESCE(Kategorija, 0) + 1
      WHERE Išsilavinimas IS NULL ;
else
   EXEC SQL UPDATE Vykdytojai
      SET Kategorija = COALESCE(Kategorija, 0) + 1
      WHERE Išsilavinimas = :value ;
```

8.5. Statinių užklausų apdorojimas

Užklausos rezultato perrinkimui eilutė po eilutės programų SQL yra naudojamas **eilučių žymeklis** (angl. *cursor*):

DECLARE CURSOR – apibrėžiama užklausa ir jos žymeklis.

OPEN pradedamas užklausos vykdymas (žymeklis atveriamas).

FETCH – žymeklis perkeliams prie artimiausios užklausos rezultato eilutės ir tos eilutės duomenys priskiriami baziniams kintamiesiems.

CLOSE – nutraukiamas rezultato perrinkimas (žymeklis uždaromas).


```
void Vykdytojai_Kategorijos ()
{
 EXEC SQL INCLUDE SQLCA ;
 EXEC SQL BEGIN DECLARE SECTION;
   char   name [31];     /* pavardė    */
   short  category;      /* kategorija */
   short  ind;           /* indikatorius  */
 EXEC SQL END DECLARE SECTION;
 EXEC SQL WHENEVER SQLERROR  GOTO error;
 EXEC SQL WHENEVER NOT FOUND GOTO end;
 EXEC SQL DECLARE curs CURSOR FOR
        SELECT Pavardė, Kategorija
        FROM Vykdytojai ORDER BY Pavardė;
```



```
EXEC SQL CONNECT TO Darbai ;
EXEC SQL OPEN curs ;
while ( 1 ) {
  EXEC SQL FETCH curs INTO :name, :category:ind;
  printf("Pavardė: %s ", name ) ;
  if ( ind >= 0 )
     printf( "Kategorija: %d\n", category) ;
  else
     printf("Kategorijos nėra\n") ;
}
```



```
error:
  printf( "SQL klaida: %ld\n", SQLCODE) ;
end:
  EXEC SQL WHENEVER SQLERROR  CONTINUE;
  EXEC SQL WHENEVER NOT FOUND CONTINUE;
  EXEC SQL CLOSE curs ;
  EXEC SQL CONNECT RESET ;
} /* funkcijos Vykdytojai_Kategorijos pabaiga */
```


- Žymens susiejimas su užklausa leidžia: vienu metu apdoroti keletą užklausų.
- Apibrėžiant užklausą (**DECLARE CURSOR**) galima vartoti bazinius kintamuosius.
- **SELECT** frazėje baziniai kintamieji neleidžiami.
- Atidarytas žymuo uždaromas transakcijai pasibaigus.
- SQL1 numato užklausos rezultato perrinkimą tik viena kryptimi.
- SQL2 **FETCH** papildytas: **FIRST**, **LAST**, **PRIOR**, **ABSOLUTE** <eilutės numeris> **RELATIVE** <+|–> <eilučių kiekis>

Kai iš anksto žinome, kad užklausos **rezultatas -1 eilutė**:

```
SELECT <stulpeliai>
INTO <programos kintamieji> FROM  <lentelės>
[WHERE      <paieškos sąlyga>]
[GROUP BY <grupavimo stulpeliai>]
[ORDER BY <sutvarkymo stulpeliai>]
[FETCH FIRST ROW ONLY]
```


Vykdytojo pavardė ir kategorija, kurio Nr yra kintamajame *nr*

```
EXEC SQL SELECT Pavardė, Kategorija
            INTO :name, :category :ind
            FROM Vykdytojai WHERE Nr = :nr;
if (0 > SQLCODE)
  printf("Įvyko klaida, kodas: %ld\n", SQLCODE);
else if  (100  == SQLCODE)
  printf("Vykdytojas Nr.: %d yra nežinomas\n", nr) ;
else if (0  == SQLCODE)
  printf("Nr.: %d, pavardė: %s, kategorija: %d\n",
         nr, name, (ind < 0 ? -1 : category)) ;
```


Jei **SELECT INTO** rezultatą sudaro **> 1 eilutė** – **klaida**.

Jei reikalavimus gali tenkinti kelios eilutės, bet mus domina tik 1-oji, tai fraze

FETCH FIRST ROW ONLY

galima nurodyti sistemai pateikti tik pirmąją eilutę.

8.6. Pozicinis duomenų šalinimas ir atnaujinimas

Užklausos rezultato apdorojimas – tai ne tik duomenų peržiūrėjimas, bet ir duomenų šalinimas bei atnaujinimas.

Programų SQL yra sakinių **DELETE** ir **UPDATE** pozicinės formos:

```
DELETE FROM <lentelės vardas>
WHERE CURRENT OF <žymeklio vardas>

UPDATE <lentelės vardas> ...
WHERE CURRENT OF <žymeklio vardas>
```



```
DELETE FROM <lentelės vardas>
WHERE CURRENT OF <žymeklio vardas>
```

- Šalinama einamoji eilutė.
- Po šalinimo žymeklis perkeliamas prie kitos eilutės, žymeklio pozicija yra prieš kitą eilutę.
- Kita eilutė tampa einamaja po **FETCH**.


```
UPDATE <lentelės vardas> ...
WHERE CURRENT OF <žymeklio vardas>
```

Tam, kad einamąją eilutę galima būtų keisti, užklausa turi tenkinti:

- užklausos **FROM** frazėje yra tik viena lentelė;
- užklausoje nėra **ORDER BY**;
- užklausoje nėra **DISTINCT**;
- užklausoje nėra **GROUP BY**;
- užklausa yra apibrėžta nurodant, kad jos rezultatas gali būti keičiamas – nurodyta frazė **FOR UPDATE**.

Kai kurios DBVS reikalauja šalia frazės **FOR UPDATE** išvardinti stulpelius, kurių reikšmės gali būti keičiamos perrenkant rezultatą.

Jei yra žinoma, kad atidarius žymeklį, eilutės nebus nei keičiamos ir nei šalinamo, tai galima nurodyti

FOR READ ONLY

```
EXEC SQL DECLARE curs CURSOR FOR
      SELECT Pavardė, Kategorija
      FROM Vykdytojai FOR READ ONLY;
```



```
short Kategorija_Atnaujinimas()
{
  EXEC SQL INCLUDE SQLCA ;
  EXEC SQL BEGIN DECLARE SECTION;
    char   name [31];
    short  category, ind;
  EXEC SQL END DECLARE SECTION;
  char   inbuffer[40] ;
  short  ok = 1;
  EXEC SQL WHENEVER SQLERROR GOTO error;
  EXEC SQL WHENEVER NOT FOUND GOTO end;
```



```
EXEC SQL DECLARE curs CURSOR
    FOR SELECT Pavardė, Kategorija
    FROM Vykdytojai FOR UPDATE OF Kategorija;
EXEC SQL CONNECT TO Darbai ;
EXEC SQL OPEN curs ;
while(1) {
  EXEC SQL FETCH curs INTO :name, :category:ind;
  printf("Pavardė: %s ", name ) ;
  if ( ind >= 0 )
     printf("Kategorija: %d\n", category) ;
  else
     printf("Kategorija: – \n") ;
```



```
action:
  printf("Pasirinkite: N, D, U, F, C\n") ;
  scanf("%s", inbuffer) ;
  switch(inbuffer[0]) {
    case 'N':
       break ;
    case 'D':
       EXEC SQL DELETE FROM Vykdytojai
                WHERE CURRENT OF curs ;
       break ;
```

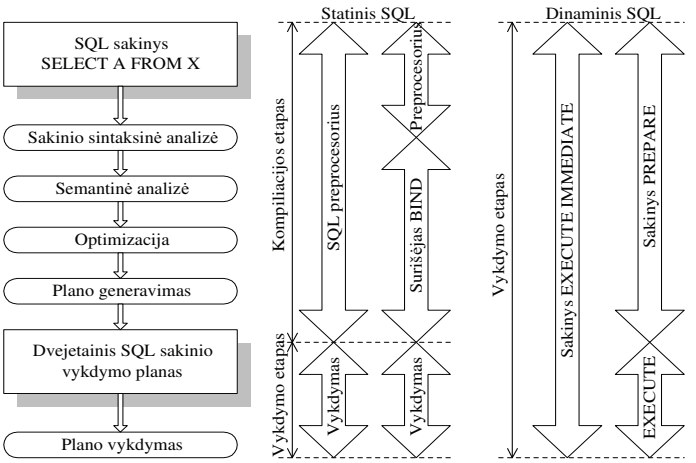


```
    case 'U': printf("Įveskite kategoriją: ") ;
              scanf("%d", &category);
              EXEC SQL UPDATE Vykdytojai
                       SET Kategorija = :category
                       WHERE CURRENT OF curs ;
              break ;
    case 'F': goto end;
    case 'C': ok = 0 ;
              goto end;
    default: goto action;
  } /* switch pabaiga*/
} /* ciklo pabaiga*/
```

```
error:
   printf("SQL Klaida: %ld\n", SQLCODE) ; ok = 0;
end:
   EXEC SQL WHENEVER SQLERROR CONTINUE;
   EXEC SQL WHENEVER NOT FOUND CONTINUE;
   EXEC SQL CLOSE curs ;
   if (ok)
      EXEC SQL COMMIT ;
   else
      EXEC SQL ROLLBACK ;
   EXEC SQL CONNECT RESET ;
 return ok ;
} /* Funkcijos Kategorija_Atnaujinimas pabaiga */
```


8.7. Dinaminių SQL sakinių vykdymas

- Statiniuose SQL sakiniuose lankstumo siekiama tik baziniais kintamaisiais.
- Statiniuose SQL sakiniuose galima parametrizuoti tik konstantas, DB objektų – NE.
- Statinis SQL puikiai tinka, kai programos sudarymo metu yra žinoma visi SQL sakiniai ir juose dalyvaujantys objektai.
- Jei kreipimosi į DB sakiniai paaiškėja tik programos vykdymo metu, yra naudojami dinaminiai SQL sakiniai.




```
EXEC SQL BEGIN DECLARE SECTION;
  char sqlStmt[256] ;
EXEC SQL END DECLARE SECTION;
long nr ;
printf("Įveskite šalinamo vykdytojo Nr: ");
scanf("%ld", &nr);
sprintf( sqlStmt,
     "delete from Vykdytojai where Nr=%ld", nr ) ;
EXEC SQL EXECUTE IMMEDIATE FROM :sqlStmt ;
```


Sakiniu **EXECUTE IMMEDIATE** galima įvykdyti daugelį SQL DML sakinių: **INSERT**, **UPDATE**, **DELETE**, **COMMIT**, **ROLLBACK** ir pan.

Daugumą DDL (**CREATE**, **DROP**) - taip pat.

Programos fragmentas, bet kuriam iš šių SQL sakinių įvykdyti:

```
printf("Įveskite DML sakinį: ");
scanf("%s", sqlStmt);
EXEC SQL EXECUTE IMMEDIATE FROM :sqlStmt ;
if(SQLCODE < 0)
  printf("SQL klaida: %ld\n", SQLCODE);
else
  printf("Sakinys įvykdytas sėkmingai\n");
```


8.8. Dvietapis dinaminis SQL sakinių vykdymas

Dvietapis dinaminių SQL sakinių vykdymas:

- Suformuojama SQL sakinio simbolių eilutė, kaip ir **EXECUTE IMMEDIATE** atveju. Konstantos sakinyje gali būti pakeistos **parametro markeriu** – klaustuku.
- Sakiniu **PREPARE** yra analizuojama simbolių eilutės sintaksė ir semantika, parenkamas optimalus sakinio vykdymo kelias ir generuojamas planas. Paruoštam sakiniui suteikiamas vardas.
- Sakiniu **EXECUTE** paruoštas sakinys yra vykdomas reikiamą kiekį kartų, kiekvieną kartą nurodant parametrų reikšmės.


```
EXEC SQL BEGIN DECLARE SECTION;
  char  sqlStmt[256] ;
  short newValue, searchValue ;
EXEC SQL END DECLARE SECTION;
char yes_no[2] ;
strcpy( sqlStmt,
"UPDATE Projektai SET Trukmė = ? WHERE Nr = ?") ;
EXEC SQL PREPARE stmt FROM :sqlStmt ;
```

```
while(1) {
  printf("Įveskite projekto Nr.: ");
  scanf("%d", &searchValue) ;
  printf("Įveskite projekto Nr. %d naują trukmę: ",
          searchValue) ;
  scanf("%d", &newValue) ;
  EXEC SQL EXECUTE stmt
           USING :newValue, :searchValue ;
  if(SQLCODE < 0) {
    printf("SQL klaida: %ld\n", SQLCODE);
    break ;
  }
```



```
  printf("Ar tęsti (Y/N)? ");
  scanf("%s", yes_no) ;
  if('N'== yes_no[0])
     break ;
}
```


Parametrizuotos užklausos pavyzdys:

```
EXEC SQL BEGIN DECLARE SECTION;
  char sqlStmt[256] ;
  char buffer[80] ;
  char name[31] ;
EXEC SQL END DECLARE SECTION;

strcpy( sqlStmt,"SELECT Pavardė FROM Vykdytojai ");
strcat( sqlStmt, "WHERE Kvalifikacija = ?") ;

EXEC SQL PREPARE s1 FROM :sqlStmt ;
EXEC SQL DECLARE c1 CURSOR FOR s1 ;
```



```
do {
  printf("Įveskite kvalifikaciją: " );
  scanf("%s", buffer) ;
  if( 0 == buffer[0]) break ;
  EXEC SQL OPEN c1 USING :buffer ;
  do {
      EXEC SQL FETCH c1 INTO :name ;
      if(SQLCODE != 0) break;
      printf("Pavardė: %s\n", name );
  } while(1);
  EXEC SQL CLOSE c1;
} while(1);
```