SQL

- Acronim pentru Structured Query Language
- Dezvoltat pentru sistemul de gestiune a bazelor de date System R, creat de IBM Research Laboratory, San Jose, California, la sfârşitul anilor ’70.
- SQL a fost standardizat şi a devenit limbajul de referinţă pentru bazele de date relaţionale.
- SQL are proprietăţile:
  - unui *limbaj de definire a datelor*, LDD (comenzi pentru definirea unei scheme a unei baze de date relaţionale);
  - unui *limbaj de manipulare a datelor*, LMD (comenzi pentru modificarea şi interogarea unei instanţe a unei baze de date relaţionale).

5.1 Definirea datelor în SQL

În această secţiune vom ilustra utilizarea SQL pentru definirea schemei unei baze de date.

Notaţii folosite în sintaxa limbajului:

- cuvintele cheie - caractere normale
- variabilele - caractere italice.
- parantezele unghiulare <***> marchează termenii;
- parantezele pătrate [***] - termenii delimitaţi sunt opţionali (pot să nu apară sau să apară doar o singură dată);
- acoladele {***} - termenul din interior poate să nu apară sau poate fi repetat de un număr arbitrar de ori;
- barele verticale - unul dintre termenii delimitaţi de acestea trebuie să apară.
- parantezele () rotunde - cuvinte cheie ale SQL.

5.1.1 Domenii elementare

SQL pune la dispoziţie şase familii de domenii elementare, care pot fi utilizate pentru definirea domeniilor asociate atributelor schemei.

1) *Caracter* - permite reprezentarea caracterelor sau a şirurilor de caractere.

Lungimea şirurilor poate fi fixă sau variabilă; în cazul şirurilor de lungime variabilă trebuie specificată lungimea maximă.

Pentru fiecare schemă este specificat un set de caractere implicit (latin, chirilic, grecesc etc.).

În cazul în care este necesară folosirea a mai mult de un set de caractere se specifică acest lucru pentru fiecare domeniu.

Sintaxa:

character [varying] [(*Lungime*)] [character set *NumeSetCaracter*]

Dacă lungimea nu este specificată, domeniul reprezintă un singur caracter.

Exemplu - şir de caractere de lungime variabilă, cu lungimea maximă de 1000 caractere, setul de caractere grecesc

character varying (1000) character set Greek

---

2) *Bit -* este utilizat pentru atribute ce pot avea doar două valori: 0 sau 1.

Se foloseşte pentru reprezentarea atributelor de tip *flag* (specifică dacă un obiect are sau nu o anumită proprietate).

SQL pune la dispoziţie de asemenea domeniul „şir de biţi”, lungimea şirului fiind specificată ca parametru.

Şirurile de biţi sunt utile pentru reprezentarea unui grup de proprietăţi.

Sintaxa:

bit [varying] [(*Lungime*)]

Exemplu - şir de biţi de lungime variabilă, cu lungimea maximă de 100 caractere

bit varying (100)

---

3) *Domenii numerice exacte* - permit reprezentarea valorilor exacte, de tip întreg sau cu parte fracţionară.

SQL pune la dispoziţie patru domenii numerice diferite:

1. numeric [(*Precizie* [,*Scală*])]
2. decimal [(*Precizie* [,*Scală*])]
3. integer
4. smallint

Domeniile *numeric* şi *decimal* reprezintă numere în baza 10. Parametrul *Precizie* specifică numărul total de digiţi, iar parametrul *Scală* indică numărul de digiţi folosiţi pentru partea fracţionară.

Exemplu

decimal (4) – valori între -9999 şi +9999

numeric (6,3) – valori între -999,999 şi +999,999

Domeniile *numeric* şi *decimal* sunt similare funcţional.

Diferenţe: numeric (precizie fixă), decimal (precizia – cerinţă minimă)

Dacă precizia nu este specificată, sistemul utilizează valoarea implicită. Dacă scala nu este specificată, se presupune a fi zero.

Domeniile *integer* şi *smallint* pot fi utilizate când nu este nevoie de parte fracţionară.

4) *Domenii numerice aproximative* - permit descrierea numerelor reale, prin intermediul reprezentării în virgulă mobilă, unde fiecare număr corespunde unei perechi *mantisă – exponent*.

Mantisa este o valoarea fracţionară iar exponentul este un întreg.

Valoarea aproximativă a unui număr real se obţine înmulţind mantisa cu puterea lui 10 specificată prin exponent.

Exemplu

$0.17E16 \rightarrow 0.17 \cdot 10^{16}$

0.17 – mantisă; 16 – exponent; valoare = mantisă $\cdot$ $10^{exponent}$

SQL pune la dispoziţie următoarele domenii numerice aproximative:

- float [(*Precizie*)]
- real
- double precision

Pentru domeniul *float* se poate furniza, ca parametru, numărul de digiţi dedicaţi pentru reprezentarea mantisei (parametrul *Precizie*).

Domeniul *double precision* reprezintă numere cu o precizie ridicată faţă de domeniul *real*.

5) *Dată calendaristică şi timp* - oferă suport pentru gestiunea informaţiilor temporale

- date
- time [(*Precizie*)] [with time zone]
- timestamp [(*Precizie*)] [with time zone]

Fiecare domeniu poate fi structurat pe câmpuri:

- domeniul *date* pune la dispoziţie câmpurile *year*, *month* şi *day*
- domeniul *time* câmpurile *hour*, *minute* şi *second*
- domeniul *timestamp* pune la dispoziţie toate câmpurile celor două domenii amintite anterior.

Pentru domeniile *time* şi *timestamp* se poate specifica numărul de poziţii zecimale utilizate în reprezentarea fracţiunilor de secundă (parametrul *Precizie*). Dacă parametrul *Precizie* este omis:

- domeniul *time* va folosi precizie 0 (rezoluţie la nivel de secundă)
- domeniul *timestamp* va folosi o precizie de 6 (rezoluţie la nivel de microsecundă).

Dacă este specificată opţiunea *with time zone*, va fi posibilă accesarea a două câmpuri suplimentare: *timezone_hour* şi *timezone_minute* (reprezintă diferenţa dintre timpul local şi timpul Greenwich).

6) *Intervale temporale* - oferă posibilitatea reprezentării intervalelor de timp (de exemplu durata unei acţiuni).

Sintaxa:

Interval *PrimaUnitateDeTimp* [to *UltimaUnitateDeTimp*]

PrimaUnitateDeTimp şi *UltimaUnitateDeTimp* definesc unităţile de măsură ce trebuie folosite.

Unităţile de măsură se împart în două grupuri distincte:

- *year* şi *month*;
- de la *day* la *second*.

Această separare are loc deoarece este imposibilă compararea exactă a zilelor şi a lunilor (deoarece o lună poate avea între 28 şi 31 zile).

```
INTERVAL YEAR [(year_precision)] TO MONTH

INTERVAL DAY [(day_precision)] TO SECOND [(fractional_seconds_precision)]
```

INTERVAL YEAR [(year_precision)] TO MONTH

INTERVAL DAY [(day_precision)] TO SECOND [(fractional_seconds_precision)]

- Prima unitate de timp poate fi însoţită de precizie - numărul de digiţi zecimali utilizaţi pentru reprezentare;
  - dacă cea mai mică unitate este *second*, putem specifica numărul de poziţii zecimale utilizate (*precizia*).

- Dacă a doua unitate de timp este şi prima (deci singura) atunci primul parametru reprezintă numărul de poziţii zecimale semnificative, iar cel de-al doilea poate reprezenta numărul de poziţii zecimale pentru partea fracţionară.

Dacă precizia nu este specificată, se foloseşte valoarea implicită 2.

Exemplu

interval year(5) to month – permite reprezentarea intervalelor până la 99999 ani şi 11 luni

interval day(4) to second(6) – permite reprezentarea intervalelor până la 9999zile, 23 ore, 59 minute şi 59,999999 secunde (precizie de 1 milionime de secundă)

5.1.2 Definirea schemei bazei de date

SQL permite definirea unei scheme de baze de date ca o colecţie de obiecte.

Fiecare schemă conţine o mulţime de *domenii*, *tabele*, *indici*, *aserţii*, *vederi* şi *privilegii* şi este definită cu ajutorul următoarei sintaxe:

create schema [*NumeSchemă*] [[authorization] *Autorizare*]
{*DefiniţieElementeDinSchemă*}

- *Autorizare* - numele utilizatorului proprietar al schemei - dacă este omis se consideră că utilizatorul care a executat comanda este proprietarul schemei.
- Dacă *NumeSchemă* este omis va fi adoptat ca nume al schemei numele utilizatorului ce a executat comanda.
- După comanda create schema se pot defini obiectele din schema respectivă.

5.1.3 Definirea tabelelor

Un tabel SQL este format dintr-o mulţime ordonată de atribute şi o mulţime, posibil vidă, de constrângeri.

create table *NumeTabel*
(*NumeAtribut Domeniu* [*ValoareImplicită*] [*Constrângeri*]
{, *NumeAtribut Domeniu* [*ValoareImplicită*] [*Constrângeri*]}
[,*AlteConstrângeri*])

După ce au fost definite toate atributele, se pot defini alte constrângeri ce implică mai multe atribute.

Iniţial tabelul nu conţine înregistrări, proprietarul deţinând toate privilegiile asupra tabelului, adică drepturi de a accesa şi de a modifica datele din tabel.

Exemplu

```
create table Departament
(NumeDept  char(20) primary key,
 Adresa       char(50),
 Oraş          char(20))
```

5.1.4 Domenii utilizator

La definirea tabelelor, pe lângă domeniile predefinite pot fi utilizate şi domenii definite explicit de utilizator.

Comanda SQL pentru definirea unui domeniu utilizator pe baza unui domeniu predefinit este:

create domain *NumeDomeniu* as *DomeniuElementar*
[*ValoareImplicită*]
[*Constrângeri*]

Un domeniu este caracterizat deci de un nume, de un domeniu elementar (predefinit sau un alt domeniu utilizator), de o posibilă valoare implicită şi de o mulţime, posibil vidă, de constrângeri (condiţii ce trebuie îndeplinite de valorile legale din domeniu).

SQL-2 nu dispune de constructori de domeniu de tip *structură* sau *vector*. Această limitare este dată de conceptul de model relaţional de date, model ce impune ca toate atributele să fie definite pe domenii elementare.

Declaraţia domeniilor asociază un nume de domeniu cu o mulţime de constrângeri. Acest lucru este util atunci când trebuie să repetăm aceeaşi definiţie de domeniu în mai multe tabele.

5.1.5 Valori implicite de domeniu

Termenul *ValoareImplicită* din definiţia domeniilor şi a tabelelor indică valoarea ce va fi considerată pentru atributul asociat în cazul inserării unei linii ce nu specifică o valoare pentru acel atribut.

Dacă este omisă specificarea unei valori implicite, atunci se va utiliza valoarea NULL ca valoare implicită.

Sintaxa: default <*ValoareGenerică* | user | NULL>

- *ValoareGenerică* este o valoare compatibilă cu domeniul asociat;
- opţiunea user setează ca valoare implicită numele de login al utilizatorului ce a executat comanda de actualizare a tabelului.

Exemplu

NumărCopii smallint default 0 – dacă se inserează o linie şi nu se specifică valoarea pentru acest atribut, atunci acestui atribut i se va atribui valoarea 0.

5.1.6 Constrângeri intra-relaţionale

În timpul definirii atât a domeniilor cât şi a tabelelor există posibilitatea de a defini constrângeri.

Constrângerile sunt proprietăţi ce trebuie verificate de fiecare instanţă a bazei de date şi se împart în:

- *constrângeri intra-relaţionale* (implică o singură relaţie);
- *constrângeri inter-relaţionale* (iau în considerare mai multe relaţii).

Cele mai simple constrângeri intra-relaţionale sunt:

not NULL

unique

primary key

Not Null

Această constrângere indică faptul că valoarea NULL nu este admisă ca valoare pentru atributul afectat de constrângere.

În acest caz valoarea atributului trebuie să fie specificată la inserare.

Este posibilă inserarea unei linii fără a specifica valoarea unui atribut cu constrângere *not NULL* doar dacă pentru atributul respectiv s-a definit o valoare implicită diferită de valoarea NULL.

Specificarea acestei constrângeri se face prin adăugarea cuvintelor cheie *not NULL* la definirea atributului.

Exemplu

Nume character(20) not NULL

Unique - impune ca un atribut sau o mulţime de atribute să formeze o (super) cheie.

- Se impune astfel ca linii diferite să nu conţină aceleaşi valori.
- Excepţie face valoarea NULL, care poate apărea în diverse linii fără a încălca constrângerea. Aceasta se datorează faptului că fiecare valoare NULL reprezintă o valoare necunoscută diferită de a altei valori NULL.

Există două moduri de definire a acestei constrângeri.

Prima variantă se utilizează doar în cazul în care constrângerea implică un singur atribut şi constă în adăugarea cuvântului cheie *unique* la definirea atributului.

Exemplu

NrInreg numeric(4) unique

A doua variantă se aplică în cazul în care constrângerea implică mai multe atribute şi constă în utilizarea clauzei unique (Atribut{, Atribut}) după definirea atributelor.

Exemplu

Nume character(20) not NULL,
Prenume character(20) not NULL,
Unique (Nume, Prenume)

Primary key

Această constrângere poate fi specificată o singură dată pentru un tabel şi poate fi definită pe un singur atribut sau pe o mulţime de atribute.

Definiţia unei astfel de constrângeri implică definirea implicită a unor constrângeri *not NULL* pentru atributul (atributele) cheii primare.

Exemplu

```
Nume character(20),
Prenume character(20),
Dept character(15),
Salariu numeric(9) default 0,
primary key (Nume, Prenume)
```

5.1.7 Constrângeri inter-relaţionale

Cele mai importante constrângeri inter-relaţionale sunt constrângerile de integritate referenţială.

Constrângerea de tip *foreign key* impune ca pentru fiecare linie dintr-un tabel (numit tabel intern), valoarea corespunzătoare unui atribut, diferită de NULL, să se regăsească printre valorile unui atribut din liniile unui alt tabel (numit tabel extern).

Singura cerinţă impusă de sintaxă este ca atributul referit din tabelul extern să fie subiectul unei constrângeri *unique*. Această cerinţă este satisfăcută dacă atributul în cauză formează o cheie primară pentru tabelul extern.

Constrângerile de referinţă pot fi definite în două moduri.

În cazul în care în constrângere este implicat un singur atribut se utilizează construcţia sintactică *references*, care indică tabelul extern şi atributul asociat.

Exemplu

```
Create table Angajati
(
NrInreg     numeric(6) primary key,
Nume        character(20) not NULL,
Prenume     character(20) not NULL,
Dept        character(15) references Departament (NumeDept),
Salariu     numeric(9) default 0,
Oras        character(15),
unique (Nume, Prenume)
)
```

Dacă legătura se face între o mulţime de atribute se va utiliza construcţia *foreign key,* plasată după definirea tuturor atributelor.

Această construcţie listează mai întâi atributele constrânse din tabelul intern, urmate de numele tabelului extern şi de numele atributelor referite.

Exemplu

```
Create table Angajati
(
NrInreg     numeric(6) primary key,
Nume        character(20) not NULL,
Prenume     character(20) not NULL,
Dept        character(15) references Departament (NumeDept),
Salariu     numeric(9) default 0,
Oras        character(15),
unique (Nume, Prenume),
foreign key (Nume, Prenume) references DatePersonale (Nume, Prenume)
)
```

În cazul constrângerilor discutate până acum sistemul va rejecta (generând un mesaj de eroare) orice operaţie de actualizare ce violează constrângerea.

Pentru constrângerea de referinţă SQL permite utilizatorului selecţia acţiunii ce va fi executată în cazul violării constrângerii.

Exemplu. Să considerăm constrângerea de tip cheie externă asupra atributului *Dept* în tabelul ANGAJATI.

Constrângerea poate fi violată operând atât asupra liniilor din tabelul intern ANGAJATI cât şi asupra liniilor din tabelul extern DEPARTAMENT.

Există doar două căi de a încălca constrângerea prin modificarea conţinutului tabelului intern:

- prin inserarea unei linii invalide;
- prin modificarea atributului *Dept*.

În aceste cazuri nu este oferit un suport special, operaţiile fiind pur şi simplu rejectate.

Se oferă opţiuni variate de reacţie la încălcarea constrângerii de referinţă determinată de modificarea tabelului extern.

Această diferenţă faţă de cazul anterior este dată de importanţa tabelului extern care, din punctul de vedere al aplicaţiei, reprezintă tabelul principal (*master*); tabelul intern (*slave*) trebuie să se adapteze la schimbările produse în tabelul extern.

Operaţiile asupra tabelului extern care pot produce violări ale constrângerii de referinţă sunt:

- modificarea valorilor atributelor referite;
- ştergerea de înregistrări (în exemplul anterior modificarea atributului *NumeDept* respectiv ştergerea de înregistrări din tabelul DEPARTAMENT).

Tipul reacţiei la astfel de încălcări ale constrângerii diferă în funcţie de comanda ce a generat violarea constrângerii.

În cazul operaţiilor de actualizare este posibilă reacţia în unul din modurile următoare:

- *cascade*: noua valoare pentru atributul din tabelul extern va fi atribuită tuturor liniilor corespunzătoare din tabelul intern;
- *set NULL*: valoarea NULL este asignată atributului din tabelul intern în locul valorii modificate din tabelul extern;
- *set default*: atributului din tabelul intern îi va fi asignată valoarea implicită în locul valorii modificate în tabelul extern;
- *no action*: operaţia de actualizare este rejectată.

Opţiunile disponibile în cazul încălcării constrângerii de referinţă prin ştergerea de înregistrări din tabelul extern sunt:

- *cascade*: vor fi şterse toate liniile din tabelul intern corespunzătoare liniei şterse din tabelul extern;
- *set NULL*: se asignează valoarea NULL atributului din tabelul intern în locul valorii şterse din tabelul extern;
- *set default*: atributul din tabelul intern va primi valoarea implicită în locul valorii şterse din tabelul extern;
- *no action*: operaţia de ştergere este rejectată.

Specificarea modului de reacţie în cazul violării unei constrângeri de referinţă se face imediat după definirea constrângerii prin sintaxa:

```
on <delete | update>
      <cascade | set NULL | set default | no action>
```

Exemplu

```
Create table Angajati
(
NrInreg     numeric(6),
Nume        character(20) not NULL,
Prenume  character(20) not NULL,
Dept         character(15)
Salariu      numeric(9) default 0,
Oras         character(15),
primary key (Nr Inreg),
foreign key (Dept) references Departament (NumeDept)
      on delete set NULL
      on update cascade,
unique (Nume, Prenume)
)
```

5.1.8 Actualizarea schemei unei relaţii

Comenzile definite în SQL pentru manipularea schemelor unei baze de date sunt *alter* şi *drop*.

Comanda alter - permite modificarea domeniilor şi schemelor de tabele şi poate avea o varietate de forme:

alter domain *NumeDomeniu* <set default *ValoareImplicită* |
drop default |
add constraint *DefiniţieConstrângere* |
drop constraint *NumeConstrângere*>

alter table *NumeTabel*
<alter column *NumeAtribut*
<set default *ValoareImplicită* | drop default> |
add constraint *DefiniţieConstrângere* |
drop constraint *NumeConstrângere* |
add column *DefiniţieAtribut* |
drop column *NumeAtribut*>

Prin utilizarea celor două comenzi se pot opera următoarele modificări asupra domeniilor şi tabelelor:

- adăugare / eliminare constrângeri;
- modificare valoare implicită;
- adăugare / eliminare atribute din schema unui tabel.

Notă

În momentul definirii unei noi constrângeri, datele din tabel trebuie să satisfacă acea constrângere. În caz contrar definiţia constrângerii nu va fi validată.

Comanda drop - permite eliminarea datelor de tip schemă, domeniu, tabel, vedere sau aserţie (constrângere ce nu este asociată unui anumit tabel).

Sintaxa este:

drop <schema|domain|table|view|assertion> *NumeComponentă* [<restrict | cascade>]

restrict – o componentă nu va fi validată în situaţia în care componenta nu este vidă sau este referită în definiţia altei componente.

- schemă nu va fi eliminată dacă ea conţine tabele sau alte componente;
- un domeniu nu va fi eliminat dacă apare în definiţia unui tabel ş.a.m.d.
- această opţiune este implicită.

cascade - o componentă este eliminată împreună cu toate componentele dependente.

- eliminarea unei scheme care nu este vidă va conduce la eliminarea tuturor obiectelor care intră în componenţa sa;
- ştergerea unui tabel folosind această opţiune implică ştergerea tuturor liniilor din tabel;
- opţiunea *cascade* trebuie utilizată cu mare atenţie deoarece în cazul în care există dependenţe care nu au fost luate în calcul, rezultatul poate fi diferit de cel scontat;
- multe din sistemele comerciale oferă posibilitatea testării rezultatului comenzii *drop* cu opţiunea *cascade* înainte de execuţia efectivă a comenzii.