SQL - 7

ALTE OBIECTE ALE BAZEI DE DATE

STUD

MATR	NUME	AN	GRUPA	DATAN	LOC	TUTOR	PUNCTAJ	CODS
1456	GEORGE	4	1141A	12-MAR-82	BUCURESTI		2890	11
1325	VASILE	2	1122A	05-OCT-84	PITESTI	1456	390	11
1645	MARIA	3	1131B	17-JUN-83	PLOIESTI		1400	11
3145	ION	1	2112B	24-JAN-85	PLOIESTI	3251	1670	21
2146	STANCA	4	2141A	15-MAY-82	BUCURESTI		620	21
3251	ALEX	5	2153B	07-NOV-81	BRASOV		1570	21
2215	ELENA	2	2122A	29-AUG-84	BUCURESTI	2146	890	21
4311	ADRIAN	3	2431A	31-JUL-83	BUCURESTI		450	24
3514	FLOREA	5	2452B	03-FEB-81	BRASOV		3230	24
1925	OANA	2	2421A	20-DEC-84	BUCURESTI	4311	760	24
2101	MARIUS	1	2412B	02-SEP-85	PITESTI	3514	310	24
4705	VOICU	2	2421B	19-APR-84	BRASOV	4311	1290	24

SPEC si BURSA

CODS	NUME	DOMENIU
11	MATEMATICA	STIINTE EXACTE
21	GEOGRAFIE	UMANIST
24	ISTORIE	UMANIST

TIP	PMIN	PMAX	SUMA
FARA BURSA	0	399	
BURSA SOCIALA	400	899	100
BURSA DE STUDIU	900	1799	150
BURSA DE MERIT	1800	2499	200
BURSA DE EXCEPTIE	2500	9999	300

OBIECTIV

- Acest capitol prezintă modul de lucru cu alte obiecte care pot exista într-o bază de date Oracle:
  - vederi,
  - secvenţe,
  - indecşi
  - sinonime.

VEDERI

- În capitolele anterioare a fost prezentată posibilitatea folosirii unei subcereri în clauza FROM a unei cereri SELECT precum şi ca bază de pornire pentru cereri de tip INSERT, DELETE şi UPDATE.
- Inconvenientul principal în această abordare este inexistenţa unui nume asociat cererii respective.
- În unele cazuri se pot folosi aliasuri de tabelă dar rescrierea subcererii este necesară în orice cerere care pleacă de la aceasta.

VEDERI (2)

- Ca şi toate celelalte sisteme relaţionale, Oracle permite asocierea unui nume pentru o cerere SQL de tip SELECT şi stocarea acesteia ca obiect al bazei de date.
- Termenul care desemnează o astfel de cerere este ***vedere*** (în limba engleză: ***view***).

VEDERI – UTILIZARE (1)

O vedere se comportă ca o tabelă în cazul execuţiei de cereri SELECT şi poate fi folosită uneori pentru efectuarea de actualizări ale bazei de date.

Există însă şi diferenţe între o vedere şi o tabelă, mai ales din punct de vedere al implementării:

- În cazul unei vederi, baza de date stochează definiţia acesteia (cererea SQL ca şir de caractere) şi nu datele regăsite prin aceasta.

VEDERI – UTILIZARE (2)

- Ori de câte ori se execută o cerere având la bază o vedere, ea este recalculată. Astfel, orice modificare efectuată în tabelele pe baza cărora e definită vederea se reflectă automat în aceasta.
- Structura unei vederi nu se poate modifica prin cereri de tip ALTER ci prin recrearea vederii cu specificarea unei alte cereri SQL.
- Constrângerile de integritate nu se pot defini la nivel de vedere. O vedere moşteneşte implicit toate constrângerile definite în tabelele pe baza cărora este definită pentru coloanele care sunt regăsite prin cererea asociată ei.

VEDERI – UTILIZARE (3)

Vederile permit implementarea de scheme externe prin care diverse categorii de utilizatori accesează doar acele porţiuni din baza de date pe care le folosesc în mod obişnuit, asigurându-se astfel:

- Confidenţialitatea datelor: utilizatorii care accesează baza de date doar prin intermediul unor vederi pot fi restricţionaţi în ceea ce priveşte datele pe care le pot utiliza prin însăşi definiţia vederilor respective (ele nu conţin coloanele/liniile/datele la care nu se doreşte accesul acelei categorii de utilizatori).
- Asigurarea corectitudinii datelor: prin faptul că un utilizator nu are acces la date pe care uzual nu le foloseşte este împiedicată coruperea accidentală a celorlalte date din cauza necunoaşterii semnificaţiei lor.

VEDERI-SINTAXA (1)

Sintaxa cererii de creare a unei vederi este următoarea:

```
CREATE [OR REPLACE] [FORCE | NOFORCE]
  VIEW nume_vedere
[(lista_de_coloane)]
AS subcerere
[WITH CHECK OPTION [CONSTRAINT
  nume_constrangere]]
[WITH READ ONLY [CONSTRAINT
  nume_constrangere]];
```

Semnificaţia elementelor prezente într-o astfel de cerere este următoarea:

- nume_vedere - specifică numele asociat vederii respective. Respectă aceleaşi condiţii ca în cazul numelor de tabele.
- subcerere - cererea SQL de tip SELECT asociată vederii.

VEDERI-SINTAXA (2)

Elementele opţionale au următoarea semnificaţie:

- lista_de_coloane - specifică numele coloanelor din vedere. Acestea trebuie să respecte restricţiile uzuale pentru nume de coloane descrise în capitolele precedente (30 de caractere, etc.). În lipsa acestei opţiuni coloanele vederii au aceleaşi nume cu ale rezultatului cererii SQL asociate.
- În cazul în care rezultatul cererii asociate vederii are nume de coloane care nu respectă restricţiile privind astfel de nume este obligatorie fie folosirea listei_de_coloane fie modificarea cererii prin adăugarea unor aliasuri de coloană corespunzătoare.

VEDERI-SINTAXA (3)

- OR REPLACE - în cazul în care există deja o vedere cu acel nume, ea este înlocuită. Este modul uzual prin care se modifică structura şi conţinutul unei vederi. În lipsa acestei opţiuni sistemul semnalează în astfel de situaţii o eroare.
- FORCE | NOFORCE (valoare implicită NOFORCE) - la crearea unei vederi sistemul verifică existenţa tabelelor folosite în definiţia acesteia şi semnalează erori în cazul detectării unor absenţe.
- În cazul folosirii opţiunii FORCE această verificare nu mai duce la eroare şi vederea este stocată în baza de date. Vederea nu poate însă fi folosită până când discrepanţa dintre definiţia sa şi structura bazei de date nu este înlaturată.

VEDERI-SINTAXA (4)

- WITH CHECK OPTION - ca şi în capitolul precedent, această opţiune împiedică efectuarea de modificări ale conţinutului bazei de date dacă liniile inserate/actualizate nu sunt regăsite prin cererea asociată vederii. Această opţiune este din categoria constrângerilor de integritate şi se poate asocia un nume cu CONSTRAINT nume.
- WITH READ ONLY - nu sunt permise modificări ale datelor prin intermediul vederii. De asemenea se poate asocia un nume pentru aceasta constrângere.
- Asocierea de nume pentru ultimele două opţiuni permite activarea şi dezactivarea lor.

EXEMPLE

```
CREATE OR REPLACE VIEW STUD_AN1 AS
SELECT * FROM STUD WHERE AN = 1
WITH CHECK OPTION;

CREATE OR REPLACE VIEW LISTA_AN1
(MATRICOLA, NUMELE, ANUL, NRPUNCTE) AS
SELECT MATR, NUME, AN, PUNCTAJ
FROM STUD_AN1
WHERE CODS = 21
WITH READ ONLY;
```

OBSERVATII

- O vedere se poate folosi pentru definirea altor vederi (LISTA_AN1 e definită pe baza STUD_AN1).
- Prin vederea STUD_AN1 nu se pot insera decât studenţi de anul 1 şi nu se poate modifica valoarea anului de studiu din 1 în altă valoare, nulă sau nenulă (constrângerea WITH CHECK OPTION).
- Prin vederea LISTA_AN1 se pot efectua doar regăsiri de date (constrângerea WITH READ ONLY).
- Numele coloanelor din vederea LISTA_AN1 este altul decât cel din tabela STUD şi vederea STUD_AN1 ca urmare a folosirii unei liste de nume în cererea de creare.

ALT EXEMPLU

Se pot defini vederi având asociate cereri SQL oricât de complicate. Un exemplu de vedere având la baza joinul celor trei tabele STUD, SPEC şi BURSA este următorul:

```
CREATE OR REPLACE VIEW LISTA_BURSA AS
SELECT MATR, ST.NUME NUMESTUD, AN,
  PUNCTAJ AS PCT,
        SP.NUME NUMESPEC, SUMA
FROM STUD ST, SPEC SP, BURSA
WHERE ST.CODS = SP.CODS AND
PUNCTAJ BETWEEN PMIN AND PMAX
ORDER BY SP.NUME, PUNCTAJ DESC;
```

STERGERE VEDERI

- Ştergerea unei vederi se face cu cererea DROP VIEW. Sintaxa acesteia este:

```
DROP VIEW nume_vedere
```

- Exemplu: ştergerea vederilor STUD_AN1 şi LISTA_A1 se face cu cererile:

```
DROP VIEW STUD_AN1;
DROP VIEW LISTA_AN1;
```

- Fiind o comandă DDL, ştergerea unei vederi nu poate fi revocată cu ROLLBACK.

MODIFICARE DATE PRIN VEDERI

Ştergere linii prin vederi

Cererea DELETE poate avea la bază o vedere dacă aceasta verifică următoarele restricţii:

- Are la bază o singură tabelă
- Nu conţine funcţii de grup
- Nu conţine clauza GROUP BY
- Nu conţine clauza DISTINCT
- Nu conţine funcţia ROWNUM

EXEMPLE (1)

- Următoarele vederi nu pot fi folosite pentru ştergere deoarece încalcă una sau mai multe din restricţiile de mai sus:
- Exemplul 1: Vedere care conţine DISTINCT, grupare şi funcţii de grup:

```
CREATE OR REPLACE VIEW NU_STERGE AS
SELECT DISTINCT MAX(PUNCTAJ) MAXIM FROM
  STUD
GROUP BY CODS;
```

- Exemplul 2: Vedere care conţine ROWNUM şi join:

```
CREATE OR REPLACE VIEW NU_STERGE AS
SELECT ROWNUM NRCRT, ST.NUME NUMESTUD,
  SP.NUME NUMESPEC
FROM STUD ST, SPEC SP
WHERE ST.CODS = SP.CODS;
```

EXEMPLE (2)

În ambele cazuri a fost nevoie de utilizarea unor aliasuri de coloană deoarece:

- MAX(PUNCTAJ) nu respectă convenţia pentru nume de coloane (conţine paranteze).
- ST.NUME şi SP.NUME duc la crearea unei vederi având două coloane cu acelaşi nume (NUME).
- ROWNUM este un nume ilegal de coloană (cuvânt rezervat SQL)

MODIFICARE DATE PRIN VEDERI

Actualizare

Pentru a efectua actualizări (UPDATE) prin intermediul unei vederi aceasta trebuie să verifice următoarele restricţii:

- Vederea verifică toate restricţiile enumerate anterior pentru ştergere,
- Comanda UPDATE nu actualizează coloanele definite prin expresii.

EXEMPLU (1)

Prin intermediul vederii TBURSA

- se pot actualiza toate coloanele vederii care provin din coloane ale tabelei BURSA dar
- nu se poate specifica modificarea coloanei suplimentare SUMAMARITA definită de expresia aritmetica SUMA*1.5:

```
CREATE OR REPLACE VIEW TBURSA AS
SELECT PMIN MINIM, PMAX MAXIM,
       TIP TIPBURSA, SUMA, SUMA*1.5
  SUMAMARITA
FROM BURSA;
```

EXEMPLU (2)

- Următoarea actualizare va mări valoarea sumei în cazul bursei având punctajul minim 2500:

```
UPDATE TBURSA
SET SUMA = 400
WHERE MINIM = 2500;
```

- În schimb cererea de actualizare:

```
UPDATE TBURSA
SET SUMAMARITA = 600
WHERE MINIM = 2500;
```

- va semnala eroarea *ORA-01733: virtual column not allowed here* deoarece se încearcă actualizarea unei coloane definite printr-o expresie.

MODIFICARE DATE PRIN VEDERI

Inserare

Pentru a se putea insera noi linii prin intermediul unei vederi, aceasta trebuie să satisfacă următoarele restricţii:

- Vederea verifică toate restricţiile enumerate anterior pentru ştergere,
- Vederea conţine toate coloanele tabelei de bază pentru care există constrângeri de tip NOT NULL
- Comanda de inserare nu încearcă să insereze date în coloane care provin din expresii.

EXEMPLU (1)

- cazul vederii TBURSA descrisă mai sus, aceasta conţine toate coloanele tabelei BURSA (deci inclusiv pe cele pentru care am putea avea constrângeri de tip NOT NULL) şi respectă şi restricţiile enumerate la ştergere.
- Rezultă că vom putea executa cu succes o cerere de inserare de tipul:

```
INSERT INTO TBURSA
(MINIM, MAXIM, TIPBURSA, SUMA)
VALUES (5000, 9999, 'SUPLIMENTARA',
  500);
```

EXEMPLU (2)

- în schimb cererea

```
INSERT INTO TBURSA
VALUES (5000, 9999,
  'SUPLIMENTARA', 500, 750);
```

va returna eroarea *ORA-01733: virtual column not allowed here*.

OBSERVATII

- Operaţiile de modificare a conţinutului tabelelor prin intermediul vederilor definite pe baza lor trebuie să respecte toate constrângerile de integritate ale acestora.
- De exemplu:
  - nu se poate şterge o linie printr-o vedere dacă ea conţine o cheie referită prin FOREIGN KEY,
  - nu se pot insera linii care duplică o valoare de cheie primară sau
  - nu se poate actualiza o linie astfel încât o constrângere de tip CHECK nu mai este verificată.

OBIECTE

- vederi,
- secvenţe,
- indecşi
- sinonime.

SECVENTE

- În multe aplicaţii este nevoie de generarea unor secvenţe de numere care să fie folosite în comenzi de tip INSERT sau UPDATE pentru valorile cheilor primare ale înregistrărilor inserate sau modificate.
- Soluţia calculării unei valori maxime şi a incrementării ei nu este corectă în contextul unei utilizări concurente a datelor.
- O cerere de tipul următor:

```
SELECT MAX(MATR) + 1
FROM STUD;
```

nu ne garantează obţinerea de valori unice pentru numerele matricole din înregistrările inserate în tabela STUD deoarece ea poate fi executată simultan, returnând aceeaşi valoare.

SECVENTE – cont.

- Soluţia care se foloseşte în Oracle este aceea a definirii unor obiecte numite ***secvenţe*** care pot fi apelate în cereri SQL şi care nu returnează niciodată o aceeaşi valoare.
- O secvenţă nu este asociată unei tabele ci poate fi folosită pentru a genera valori care se utilizează oriunde este necesar.

SECVENTE – SINTAXA (1)

- Sintaxa definirii unei secvenţe este următoarea:

```
CREATE SEQUENCE nume_secventa
[INCREMENT BY pas]
[START WITH valoare_initiala]
[MAXVALUE valoare_maxima | NOMAXVALUE]
[MINVALUE valoare_minima | NOMINVALUE]
[CYCLE | NOCYCLE]
[CACHE numar_valori | NOCACHE];
```

SECVENTE – SINTAXA (1)

- nume_secventa: numele obiectului de tip secvenţă. Respectă convenţiile Oracle privitoare la nume.
- pas: fiecare valoare generată se obţine din precedenta prin adăugarea acestui pas. Implicit pasul are valoarea 1. În cazul în care valoarea sa este negativă secvenţa este descrescătoare
- valoare_initiala: prima valoare generată de secvenţă.
- valoare_maxima: în cazul specificării unei astfel de valori, secvenţa nu poate genera valori mai mari decât aceasta. Implicit nu există o limitare superioară a valorilor generate (NOMAXVALUE).

SECVENTE – SINTAXA (2)

- valoare_minima: în cazul secvenţelor descrescătoare secvenţa nu poate genera valori mai mici decât aceasta. Implicit nu există o limitare inferioară a valorilor generate (NOMINVALUE).
- CYCLE: în cazul atingerii valorii maxime sau minime, secvenţa reporneşte de la valoarea iniţială. Implicit secvenţele se crează cu NOCYCLE (fără ciclare).
- CACHE numar_valori: implicit sistemul generează şi ţine în memorie următoarele 20 de valori din secvenţă. În cazul în care se doreşte ca numărul de valori din memorie să fie altul foloseşte această opţiune.

EXEMPLU

Pentru a genera noi numere matricole care să pornească de la valoarea 5000, pasul de incrementare să fie 2 iar valoarea maximă să fie 9998 se crează secvenţa MATRICOLA cu cererea:

```
CREATE SEQUENCE MATRICOLA
INCREMENT BY 2
START WITH 5000
MAXVALUE 9998;
```

SECVENTE - FOLOSIRE

Fiecare secvenţă se foloseşte prin intermediul celor două pseudocoloane pe care le are:

- nume_secventa.NEXTVAL returnează următoarea valoare din secvenţă.
- nume_secventa.CURRVAL returnează ultima valoare furnizată deja de secvenţă.

Exemplu: inserarea unui nou student folosind un număr matricol generat de secvenţa:

```
INSERT INTO STUD VALUES(MATRICOLA.NEXTVAL,
  'LUCA', 4, '1141B', '14-MAY-82', 'IASI',
  NULL, 1500, 24);
```

OBSERVATIE

- Inserarea unui nou student pentru care LUCA este tutor nu se poate face cu cererea:

```
INSERT INTO STUD (TUTOR, MATR, NUME,
 AN, GRUPA, DATAN, LOC, PUNCTAJ, CODS)
VALUES(MATRICOLA.CURRVAL,
  MATRICOLA.NEXTVAL, 'GEO', 1, '1111B',
  '14-MAY-85', 'IASI', 1500, 11);
```

- În acest caz, deşi CURRVAL este folosit în lista de valori înaintea lui NEXTVAL, atât în coloana MATR cât şi în TUTOR va fi inserată aceeaşi valoare (cea dată de NEXTVAL).

NEXTVAL: DA!

Pe lângă lista de valori pentru inserarea unei noi linii, NEXTVAL şi CURRVAL mai pot fi folosite:

- în clauza SET a unui UPDATE,
- în clauza SELECT a unei subcereri doar dacă aceasta face parte dintr-o comandă de inserare a rezultatelor sale,
- într-o clauză SELECT a unei cereri aflată pe primul nivel.

NEXTVAL: NU!

NEXTVAL şi CURRVAL nu se pot folosi:

- în clauza SELECT a cererii care defineşte o vedere,
- într-o cerere SELECT DISTINCT
- într-o cerere SELECT conţinând GROUP BY, HAVING sau ORDER BY,
- în subcereri ale unor cereri SELECT, DELETE sau UPDATE,
- ca valoare implicită de coloană în cereri CREATE TABLE sau ALTER TABLE.

SECVENTE - MODIFICARE

- Cererea prin care se modifică parametrii unei secvenţe are următoarea sintaxă:

```
ALTER SEQUENCE nume_secventa
[INCREMENT BY pas]
[MAXVALUE valoare_maxima | NOMAXVALUE]
[MINVALUE valoare_minima | NOMINVALUE]
[CYCLE | NOCYCLE]
[CACHE numar_valori | NOCACHE];
```

- Semnificaţia clauzelor este cea deja menţionată la cererea de creare. Singurul element care nu se poate modifica este valoarea de start.
- Noile valori ale parametrilor se aplică pentru valorile generate după modificarea secvenţei. În cazul clauzelor MAXVALUE şi MINVALUE, valorile respective nu pot fi mai mici, respectiv mai mari decat valoarea curentă a secvenţei.

SECVENTE - STERGERE

- Ştergerea unei secvenţe se face prin comanda

```
DROP SEQUENCE nume_secventa;
```

- Fiind o comandă DDL, ştergerea unei secvenţe nu poate fi revocată cu ROLLBACK.

OBIECTE

- vederi,
- secvenţe,
- indecşi
- sinonime.

INDECSI

- Un index este o structură de căutare rapidă care poate fi folosită de sistem pentru creşterea vitezei de evaluare a cererilor prin faptul că parcurgerea tabelelor nu se mai face secvenţial, înregistrare cu înregistrare, ci sunt accesate direct liniile necesare cererii respective.
- Sistemul Oracle crează automat indecşi de tip unic pentru cheile specificate la crearea tabelei prin constrângerile PRIMARY KEY şi UNIQUE.
- În plus faţă de aceştia se pot crea şi alţii, unici sau neunici (în acest ultim caz valorile indexate se pot repeta) prin cererea CREATE INDEX

INDECSI – SINTAXA

```
CREATE INDEX nume index
ON nume_tabela (expresie1 [,
  expresie2, ...]);
```

- Exemplul 1: în cazul în care se execută frecvent cereri de căutare după locul naşterii studenţilor, se poate crea un index mono-coloană după acesta:

```
CREATE INDEX INDEX_LOC
ON STUD (LOC);
```

EXEMPLE – cont.

- Exemplul 2: în cazul în care se execută frecvent cereri de căutare după locul naşterii şi codul specializării studenţilor, se poate crea un index multi-coloană:

```
CREATE INDEX INDEX_LOC_CODS
ON STUD (LOC, CODS);
```

EXEMPLE – cont.

- Exemplul 3: în cazul în care se execută frecvent cereri de căutare după anul şi locul naşterii studenţilor, se poate crea indexul următor.
- Acesta foloseşte o expresie pentru extragerea anului naşterii:

```
CREATE INDEX INDEX_AN_LOC ON STUD
(TO_NUMBER(TO_CHAR(DATAN, 'YYYY')),
 LOC);
```

OBSERVATII

- Indecşii sunt menţinuţi automat de sistem.
- Orice modificare a tabelei este însoţită de reactualizarea indecşilor dacă valorile pe baza cărora aceştia au fost creaţi se modifică.
- Din această cauză crearea unui număr mare de indecşi pentru tabele care se actualizează frecvent poate duce nu la creşterea vitezei de execuţie ci la scăderea sa.
- În documentaţia sistemului sunt enumerate cazurile în care este indicată crearea de indecşi şi cazurile în care aceasta este contraproductivă.

INDECSI: DA!

- Coloanele conţin un procent însemnat de valori nule (creşte viteza de regăsire a înregistrărilor care conţin valori nenule).
- Coloana/coloanele/expresiile respective sunt folosite în clauza WHERE a unor cereri executate frecvent.
- Coloanele conţin o mare varietate de valori,
- Tabela are un număr mare de linii şi majoritatea execuţiilor de cereri afectează un procent de maxim 4% din acestea

INDECSI: NU!

- Tabela are un număr mic de linii.
- Majoritatea execuţiilor de cereri afectează un procent mai mare de 4% din liniile tabelei.
- Coloana/coloanele/expresiile respective nu sunt folosite în clauza WHERE a unor cereri executate frecvent.
- Tabela este frecvent modificată (prin INSERT, UPDATE sau DELETE).
- Condiţiile din WHERE conţin expresii calculate pe baza coloanelor respective. În acest caz indexul nu poate fi utilizat pentru a creşte viteza de execuţie.

INDEX - STERGERE

- Ştergerea unui index se face prin cererea

```
DROP INDEX nume_index;
```

- Fiind o comandă DDL, ştergerea unui index nu poate fi revocată cu ROLLBACK.

OBIECTE

- vederi,
- secvenţe,
- indecşi
- sinonime.

SINONIME

Obiectelor din baza de date li se pot asocia nume alternative - sinonime.

Crearea sinonimelor are ca obiect simplificarea operării cu obiectele respective, putând asigura:

- Nume alternative mai sugestive pentru obiecte care la creare au primit o denumire criptică.
- Nume alternative mai scurte pentru obiecte care la creare au primit o denumire lungă.
- Nume alternative mai scurte pentru obiectele create de alţi utilizatori (pentru a fi accesate acestea trebuie prefixate cu numele utilizatorului care le deţine).

SINTAXA

- Cererea de creare a unui sinonim are următoarea sintaxă:

```
CREATE [PUBLIC] SYNONYM sinonim
FOR obiect;
```

- Opţiunea PUBLIC specifică faptul că sinonimul este accesibil şi celorlalţi utilizatori ai sistemului.

EXEMPLU

- Exemplu:

```
CREATE PUBLIC SYNONYM ST
FOR STUD;
```

- După crearea acestui sinonim, ST poate apare în cereri în locul lui STUD, dar nu şi împreună cu acesta.
- De exemplu, cererea:

```
SELECT A.NUME, B.NUME
FROM STUD A, SPEC B
WHERE ST.CODS = B.CODS;
```

va semnala o eroare deoarece în contextul clauzei FROM conţinute, pentru coloanele din STUD se poate folosi pentru prefixare fie STUD fie A.

EXEMPLU – cont.

- În schimb cererea:

```
SELECT ST.NUME, SPEC.NUME
FROM ST, SPEC
WHERE ST.CODS = SPEC.CODS;
```

va returna rezultatul dorit: numele studenţilor şi numele specializărilor acestora.

SINONIM - STERGERE

- Ştergerea unui sinonim se face prin cererea

```
DROP [PUBLIC] SYNONYM nume_sinonim;
```

- PUBLIC se foloseşte pentru ştergerea sinonimelor create cu această opţiune. Fiind o comandă DDL, ştergerea unui sinonim nu poate fi revocată cu ROLLBACK.

Sfarsitul capitolului

ALTE OBIECTE ALE BAZEI DE DATE