Laboratorul 2

De la BD

Obiective:

- Înțelegerea modelului relațional al bazelor de date
- Realizarea unei imagini de ansamblu asupra limbajului SQL
- Fraza SELECT: Operatori, clauza WHERE si ORDER BY

Cuprins

Modelul relațional

- este un model simplu, cu implementări foarte eficiente şi un limbaj de interogare (SQL) descriptiv simplu si expresiv;
- propus in anii '70 de catre Edgar Frank Codd, se regaseste in cele mai populare sisteme de gestiune de

baze de date: http://db-engines.com/en/ranking_trend

Elemente:

- Baza de date constă dintr-un set de tabele/relaţii
- Fiecare tabel/relaţie are un nume unic in baza de date şi conţine un set de coloane/atribute/câmpuri
- Fiecare atribut are un nume unic in tabelul la care aparţine si are asociat tip/domeniu din care poate lua valori
- Fiecare linie/înregistrare/tuplu din tabel conţine valori pentru toate atributele
- Ordinea liniilor/tuplelor nu are semnificaţie

Schema bazei de date relaţionale reprezintă descrierea structurală a tabelelor în baza de date.

Instanţa bazei de date reprezintă conţinutul bazei de date la un anumit moment

NULL – este o valoare specială utilizată când nu se cunoaşte (sau e nedefinită) valoarea unui atribut pentru o înregistrare.

Cheie candidat – un atribut care are valori unice pentru fiecare înregistrare - sau un set de atribute ale căror valori combinate sunt unice, minimală în raport cu această proprietate (nici o submulţime nu e cheie); dacă nu este îndeplinită proprietatea de mulţime minimală avem de a face cu o supercheie

Cheie primară – una dintre cheile candidat aleasă pentru a identifica în mod unic înregistrările din tabel; nici un atribut implicat nu poate avea valoarea NULL

Cheie alternativă – cheie candidat care nu a fost aleasă drept cheie primară

Cheie străină – un atribut sau un set de atribute dintr-o relaţie care are corespondent o cheie candidat a altei relaţii (sau chiar a relaţiei curente, caz în care o numim cheie străină recursivă)

Obţinerea de informaţii din baza de date se realizează cu ajutorul interogărilor SQL. Rezultatul unei interogări este un nou tabel.

Algebra relaţională

- Un limbaj formal
- Operatorii de bază din algebra relaţională: selecţie, proiecţie, produs cartezian, intersecţie, diferenţă, redenumire
- Bazată pe mulţimi (duplicatele se elimină)

Limbajul SQL

SQL - Structured Query Language

- limbajul utilizat în manipularea bazelor de date relaţionale
- un limbaj esenţial declarativ; exprimă ce se doreşte de la baza de date, nu şi cum se obţine rezultatul
- un limbaj implementat cu fundamente în algebra relaţională (semantica este definită cu ajutorul algebrei relaţionale), bazat pe multi-seturi (nu se elimină duplicatele)
- deşi considerat un limbaj de programare non-procedural, majoritatea SGBD-urilor relaţionale oferă şi extensii procedurale.

Limbajul SQL a fost pentru prima dată standardizat de către Organizaţia Internaţională pentru Standardizare (ISO) în anul 1987. De atunci limbajul a suferit modificări/îmbunătăţiri în mai multe rânduri, ultima versiune stabilă fiind SQL:2008 (denumită ISO/IEC 9075:2008).

Deşi standardizat la nivel internaţional, fiecare dezvoltator de baze de date a modificat şi a extins versiunea standard pentru a exploata avantajele oferite de implementarea particulară a produsului propriu. Datorită dimensiunii mari şi a complexităţii standardului cei mai mulţi dezvoltatori nu l-au implementat în totalitate. Vice-versa, standardul nu specifică comportamentul bazei de date în anumite situaţii, decizille ramânând la latitudinea dezvoltatorilor. Se vorbeşte astfel de dialecte SQL. Rezultatul este apariţia de incompatibilităţi între SGBD-uri relaţionale diferite. Unele incompatibilităţi sunt intenţionat păstrate de dezvoltatori cu scopul de fidelizare a clienţilor.

Declaraţia de conformitate Oracle cu standardul ISO SQL:2008 poate fi studiată la adresa http://download.oracle.com/docs/cd/E14072_01/server.112/e10592/ap_standard_sql003.htm

În continuare sunt listate în forma Backus-Naur cele mai uzuale comenzi SQL. Convenţiile de sintaxă în acest format sunt următoarele:

- <> reprezintă o variabilă de substituţie
- | reprezintă conjuncţia
- […] reprezintă elemente opţionale
- {…|…|…} reprezintă exact un element din listă

Comenzi DDL (de definire a datelor)

Crearea tabelelor

```
CREATE [{GLOBAL | LOCAL} TEMPORARY] TABLE
       <nume_tabel>
        (<nume_coloană1> {data_type | domain_name} [constrângeri]
        [,<nume_coloana2> {data_type | domain_name} [constrângeri]]
        […]
        )
       [ ON COMMIT { DELETE | PRESERVE } ROWS ]
```

Opţiunea TEMPORARY specifică crearea unui tabel temporar, care îşi păstrează conţinutul doar pe durata unei tranzacţii (până la apariţia comenzii COMMIT) sau pe parcursul unei sesiuni (opţiunea ON COMMIT PRESERVE ROWS). În Oracle există doar varianta GLOBAL a tabelelor temporare: definiţia tabelului temporar este vizibilă din toate sesiunile de lucru însă accesul la date este privat (datele inserate intr-o sesiune pot fi accesate doar din sesiunea respectivă).

Ştergerea unui tabel

```
DROP TABLE <nume_tabel>  [RESTRICT | CASCADE]
```

CASCADE permite ştergerea unui tabel care e referenţiat de un VIEW sau de chei străine şi şterge view-ul şi restricţia de tip cheie străină.

În Oracle sintaxa pentru ştergerea tabelelor este:

```
DROP TABLE <nume_tabel>  [CASCADE CONSTRAINTS] [PURGE]
```

În Oracle tabelele şterse sunt reciclate dacă nu este specificată opţiunea PURGE, ele putând fi recuperate cu comanda:

```
FLASHBACK TABLE <nume_tabel>   TO
     { { SCN | TIMESTAMP } expr
     [ { ENABLE | DISABLE } TRIGGERS ] |
     BEFORE DROP [ RENAME TO <nume_nou> ]}
```

Comenzi DML (de manipulare a datelor)

Inserarea înregistrărilor

```
INSERT INTO <nume_tabel>
            [ (lista_coloane) ]
            VALUES (lista_valori)
```

Actualizarea valorilor

```
UPDATE <nume_tabel>
       SET <nume_coloana1> = valoare1
           [, <nume_coloana2> = valoare2...]
       [WHERE conditie]
```

Ştergerea înregistrărilor

```
DELETE FROM <nume_tabel>
      [WHERE conditie]
```

Interogarea bazei de date

```
SELECT [DISTINCT | ALL] {* | [expresie_coloana [AS <nume_nou>]] [,...] }
       FROM <nume_tabel> [alias] [, ...]
       [WHERE conditie]
       [GROUP BY lista_coloane] [HAVING conditie]
       [ORDER BY exp1 [ASC/DESC] [,exp2...]]
```

Tipuri de date (în Oracle)

Tip	Proprietăţi

CHAR(n)	Şiruri de lungime fixă n (1 caracter e stocat pe 1 octet). Dacă valorile introduse au mai puţine caractere se adaugă blanks (spaţii) la sfarşit ; dacă valorile introduse conţin mai multe caractere apare eroare (VALUE_ERROR)
VARCHAR2(n)	Şiruri de lungime variabilă, n este lungimea maximă. Este indicat a fi folosit cand valorile pentru un atribut diferă ca lungime fiindcă se consumă un număr de octeţi egal cu numărul de caractere din şir
NUMBER	Corespunde numerelor în virgulă fixă şi mobilă. Numere în intervalul (-10126, 10126), infinit (pozitiv şi negativ)
NUMBER(n1, n2)	Numere ce conţin în total n1 zecimale semnificative, n2 precizând numărul de cifre de la punctual zecimal până la cea mai puţin semnificativă cifră. n2 poate lua valori negative, caz în care se face rotunjirea pe numărul de poziţii specificate în stânga punctului zecimal. Ex. pentru tipul NUMBER(5,-2), introducerea valorii 12345 generează valoarea 12300 iar pentru 12399 generează 12400
DATE	Reprezintă un moment de timp stocând anul, luna, ziua, orele minutele şi secundele
LOB (large object): BLOB, CLOB, NCLOB, BFILE	Permit stocarea în format binar sau caracter a datelor nestructurate precum text, imagini, videoclipuri…

Pentru compatibilitate cu alte sisteme, Oracle suportă şi tipuri precum: decimal, integer, smallint, real; reprezentarea lor internă este însă definită de formatul tipului number.

O listă completă găsiţi la adresa: http://download.oracle.com/docs/cd/B19306_01/server.102/b14200 /sql_elements001.htm

NULL este o valoare specială care semnifică lipsa informaţiei. Ambele teste NULL=NULL şi NULL<>NULL returnează FALSE. Verificarea (in)egalităţii cu NULL se face utilizând IS [NOT] NULL.

Operatori

Operatori aritmetici uzuali

Operator	Descriere	Exemplu
*	înmulţire - pt. tipurile numerice	SELECT valoare / 10 FROM note;
/	împărţire - pt. tipurile numerice	SELECT valoare * 5 FROM note;
-	scădere - pt. tipurile numerice şi DATE	SELECT valoare-1FROM note;

Dacă unul dintre operanzi e NULL, rezultatul e NULL

Operatori pentru şiruri

|| - concatenare

Ex: `SELECT 'Numele studentului este: ' || nume FROM studenti;`

Dacă exact unul dintre operanzi e NULL, rezultatul e celălalt operand; dacă ambii operanzi sunt NULL rezultatul e NULL.

Operatori de comparare

- returnează TRUE sau FALSE, sunt aplicabili pentru toate tipurile de date

Operator	Descriere	Exemplu
= > >= < <= <>	Operatori binari, semnificaţia uzuală	SELECT nume AS "Student" FROM studenti WHERE an = 2
ANY/ SOME	Operator aplicat unei liste sau rezultatului unei interogări în conjuncţie cu unul din operatorii de comparare uzuali de mai sus, cu semnificaţia următoare: operatorul uzual primeşte ca al doilea operand fiecare din valorile din listă; returnează TRUE dacă pt. cel puţin o valoare din listă rezultatul e TRUE, altfel returnează FALSE.	SELECT * FROM studenti WHERE an = SOME (2,3);
ALL	Ca şi ANY/SOME, cu diferenţa rezultatul este TRUE doar dacă operatorul uzual returnează TRUE pentru toate valorile din listă.	SELECT * FROM studenti WHERE an >= ALL (1, 2)
IN	Verifică apartenenţa valorii primului operand la mulţimea specificată de al doilea operand. Este echivalent cu „=ANY”	SELECT * FROM studenti WHERE nume IN ('Adrian', 'Alex');
NOT IN	Returnează FALSE dacă valoarea primului operand nu face parte din lista specificată de al doilea operand. Este echivalent cu „<>ANY”	SELECT * FROM studenti WHERE an NOT IN (1,2);
BETWEEN x AND y	Îl putem considera ca pe un operator ternar: returnează TRUE dacă primul operand satisface simultan condiţiile >=x şi <=y, unde x şi y reprezentă alţi doi operanzi	SELECT nume, prenume FROM studenti WHERE an BETWEEN 1 AND 3
LIKE	Operator binar, verifică dacă primul operand este în conformitate cu un şablon specificat de al doilea operand. Şablonul este un şir de caractere, ce poate conţine unul din caracterele speciale: **%** semnifică orice şir de caractere, chiar şirul vid (de lungime 0) _ suplineşte un singur caracter Dacă cele două simboluri speciale trebuiesc	SELECT * FROM sudenti WHERE nume LIKE '%andr%';

	interpretate ca atare se utilizează caracterul ESCAPE: \% şi _	
IS [NOT] NULL	Operator unar. Singurul mod de test pentru null	SELECT * FROM studenti WHERE bursa IS NOT NULL AND bursa > 200;
EXISTS	Operator unar, verifică dacă o sub-interogare returnează măcar o linie	SELECT * FROM studenti WHERE NOT EXISTS (SELECT * FROM note WHERE studenti.nr_matricol = note.nr_matricol); SELECT * FROM studenti WHERE EXISTS(SELECT* FROM note WHERE valoare=10 AND studenti.nr_matricol=note.nr_matricol);

Operatori logici

Operator	Descriere	Exemplu
NOT	Operator unar reprezentând negaţia	SELECT * FROM studenti WHERE NOT (bursa IS NULL);
AND	Operator binar reprezentând ŞI-ul logic	SELECT * FROM studenti WHERE an='3' AND bursa IS NOT NULL;
OR	Operator binar reprezentând SAU-ul (V) logic	SELECT * FROM studenti WHERE an='3' OR bursa IS NOT NULL;

Operatori pentru mulţimi

-operanzii sunt interogări (rezultatele interogărilor – deci tabele/relaţii), semnificaţia şi restricţiile de aplicare sunt cele specificate de algebra relaţională.

Operator	Descriere	Exemplu
UNION [ALL]	Reuniunea specifică mulţimilor. Dacă ALL este specificat duplicatele nu sunt eliminate.	SELECT * FROM (SELECT nume FROM studenti WHERE an = '2' UNION SELECT nume FROM studenti WHERE an = '3');
INTERSECT [ALL]	Intersecţia specifică mulţimilor, duplicatele sunt eliminate.	SELECT * FROM (SELECT nume FROM studenti WHERE an = 2 INTERSECT SELECT nume FROM studenti WHERE bursa IS NOT NULL);
MINUS	Înregistrările distincte selectate de prima interogare care nu există în a doua interogare.	SELECT * FROM (SELECT nume FROM studenti WHERE an = 3 MINUS SELECT nume FROM studenti WHERE

		bursa IS NULL);

Schema bazei de date - studiu de caz

Fiecare facultate a unei universităţi are nevoie să-şi facă managementul intern al studenţilor pe care îi are. Spre exemplu, un aspect important în cadrul secretariatului este cunoaşterea notelor pe care studenţii le-au luat la diversele materii pe care le-au făcut. Pe baza acestor informaţii, se poate decide dacă studentul va avea sau nu bursă, dacă va primi sau nu cazare în anul ce urmează sau daca va trebui să repete o anumită materie. Deşi aceste informaţii sunt disponibile în format clasic, în cataloage, este mult mai uşor de lucrat cu un număr mare de studenţi atunci când aceştia sunt stocaţi într-o bază de date.

În afara informaţiilor referitoare la studenţi (nume, prenume, anul în care sunt înmatriculaţi, etc.) ce ar putea forma o tabelă într-o bază de date, şi alte informaţii precum notele sau numele cursurilor la care au participat trebuie să fie stocate. Nu este natural ca aceste informaţii să fie stocate împreună cu informaţiile personale ale studentului (de exemplu din cauză ca participarea la doua cursuri diferite ar presupune introducerea de doua ori - ca şi înregistrari diferite în tabelă - a aceloraşi informaţii privitoare la student) şi din acest motiv sunt necesare tabele suplimentare.

În cadrul laboratorului de baze de date, vom folosi tabele studenti, cursuri, note şi profesori, ce au urmatoarele structuri:

Tabela studenti:

- nr_matricol - câmp alfanumeric de 6 caractere ce nu poate fi nul;
- nume - câmp alfanumeric, variabil ca dimensiune dar având maxim 10 caractere ce nu poate fi nul;
- prenume - câmp alfanumeric, variabil ca dimensiune dar având maxim 10 caractere;
- an - câmp numeric ce poate reţine numere formate doar dintr-o singură cifră;
- grupa - câmp alfanumeric de 2 caractere;
- bursa - câmp numeric ce poate ţine minte valori reale formate din 4 cifre înainte de virgulă şi două cifre după;
- data_nastere - câmp de tip dată calendaristică.

Tabela cursuri:

- id_curs - câmp alfanumeric de 4 caractere ce nu poate fi nul;
- titlu_curs - câmp alfanumeric, variabil ca dimensiune dar având maxim 10 caractere ce nu poate fi nul;
- an - câmp numeric ce poate reţine numere formate doar dintr-o singură cifră;
- semestru - câmp numeric ce poate reţine o cifra;
- credite - câmp numeric ce poate reţine numere formate din două cifre.

Tabela note:

- nr_matricol - câmp alfanumeric de 6 caractere ce nu poate fi nul;

- id_curs - câmp alfanumeric de 4 caractere ce nu poate fi nul;
- valoare - câmp numeric ce poate reţine numere formate din două cifre;
- data_notare - câmp de tip dată calendaristică.

Tabela profesori:

- id_prof - câmp alfanumeric de 4 caractere ce nu poate fi nul;
- nume - câmp alfanumeric, fix ca dimensiune având 10 caractere ce nu poate fi nul;
- prenume - câmp alfanumeric, fix ca dimensiune având 10 caractere;
- grad_didactic - câmp alfanumeric, variabil ca dimensiune dar având maxim 5 caractere.

Tabela didactic

- id_prof - câmp alfanumeric de 4 caractere ce nu poate fi nul;
- id_curs - câmp alfanumeric de 4 caractere ce nu poate fi nul.

Creaţi aceste tabele utilizând contul student (creat laboratorul anterior).

Inseraţi în baza de date nou creată un student Popescu Ionut, proaspat înmatriculat în anul 2 care a luat nota 10 la materia Logica pentru informatica predata de Cristian Masalagiu în primul semestru din anul 1.

Executaţi comanda @<intrebati profesorul> pentru a rula un script de creare (similar cu cel rulat în primul laborator) care are rolul de a crea toate tabelele de mai sus şi de a le popula cu date.:

```
@"http://thor.info.uaic.ro/~vcosmin/BD/facultate.sql"
```

Pentru a vedea care este structura unei tabele (în cazul în care aţi uitat spre exemplu din cate litere poate fi format numele de familie al unui student), puteti executa comanda SQL*Plus describe:

```
DESCRIBE nume_tabel
```

Comanda SELECT (si clauzele WHERE, ORDER BY)

Selectarea datelor dintr-un tabel

Atunci când se execută o comandă de tip SELECT, pot fi preluate din baza de date un număr de linii (anumite linii), un număr de coloane (anumite coloane) sau o combinaţie între cele două.

Pentru a selecta toate liniile şi toate coloanele dintr-un tabel, poate fi executată comanda

```
SELECT * FROM nume_tabel;
```

Exerciţiu: selectaţi toate liniile şi toate coloanele din tabelul studenti.

Aceasta comanda furnizează, de obicei, mai multe informaţii decât sunt necesare. Pentru a prelua doar anumite coloane, simbolul asterisc va fi inlocuit cu numele coloanelor ce se doresc a fi preluate, despărţite prin virgulă. Un exemplu este preluarea doar a numelor şi prenumelor studenţilor înscrişi la facultate, executând comanda:

```
SELECT nume, prenume FROM studenti;
```

Atunci când, dintr-un anumit motiv, denumirea coloanei nu este reprezentativă la afișare, aceasta poate fi modificată în timpul exprimării comenzii SELECT prin definirea unui alias. Pentru aceasta se va folosi cuvântul cheie "AS" după numele coloanei ce trebuie redenumită, urmat de numele nou al coloanei. În cazul în care noua denumire este formată din mai multe cuvinte, aceasta trebuie să fie incadrată între simbolurile ghilimele. Spre exemplu, următoarea comandă va redenumi numele coloanei "nume" din tabelul profesori în "Nume Profesor":

```
SELECT nume AS "Nume Profesor" FROM profesori;
```

Observaţie: Cuvântul AS este optional, în cazul în care acesta lipseste se va insera doar un spaţiu între numele vechi al coloanei ce trebuie selectat şi noul nume. Dacă numele nou al coloanei este format doar dintr-un singur cuvânt, ghilimelele pot fi omise (în fapt, ghilimelele sunt utilizate atunci când noul nume conţine spaţii, caractere speciale sau se doreşte a fi afişat case-sensitive - predefinit este afişat cu litere mari). Ca şi exemplu, încercaţi să executaţi comanda:

```
SELECT nume profesor FROM profesori;
```

Daca sunt mai multe înregistrări similare, toate acestea vor fi afişate, de exemplu dacă se doreşte aflarea anilor de studiu în care sunt înscrişi studenţii executaţi interogarea:

```
SELECT an FROM studenti;
```

Atunci când avem mai mulţi studenţi înmatriculaţi, acest an de studiu va fi afişat pentru fiecare student în parte. Cum nu ne interesează să fie afişat de mai multe ori anul 1 (doar pentru ca avem mai mulţi studenţi înscrişi în primul an), vom utiliza clauza "DISTINCT" în cadrul comenzii SELECT tocmai pentru a restricţiona preluarea din baza de date doar a datelor unice (duplicatele vor fi eliminate). Executaţi aşadar comanda:

```
SELECT DISTINCT an FROM studenti;
```

Criteriul DISTINCT poate fi utilizat şi pentru mai multe câmpuri selectate simultan. Spre exemplu, următoarea interogare ne va returna doar combinaţiile <an,grupa> unice în tabel:

```
SELECT DISTINCT an, grupa FROM studenti;
```

Observaţie: Deşi comenzile SQL au fost scrise cu litere mari, limbajul SQL nu este unul case-sensitive. Comenzile poti fi aşadar scrise şi cu litere mici. În cadrul laboratoarelor de baze de date (pe acest WIKI) vom scrie totuşi (sau cel puţin vom încerca) cuvintele cheie cu litere mari tocmai pentru ca voi să le puteţi diferenţia de restul obiectelor (nume de tabele, nume de câmpuri) care nu fac parte din limbaj. Singurul loc în care SQL va face totuşi diferenţa între literele mari şi cele mici este în cadrul şirurilor de caractere din diverse câmpuri ale unui tabel. De exemplu, dacă veţi încerca să executaţi comanda `SELECT * FROM cursuri WHERE titlu_curs='LOGICA';` , probabil nu va funcţiona (decât dacă aveţi în tabelă un curs cu denumirea "LOGICA" - scris cu litere mari).

Observaţie: O comanda SQL poate fi scrisă pe mai multe rânduri. Dacă o comandă nu a fost terminată (încă) la apăsarea tastei enter, SQL*Plus va aştepta continuarea comenzii pe rândul următor. Comanda va fi considerată

completă la întâlnirea simbolului punct şi virgulă.

În unele cazuri, datele existente în tabele trebuie să fie modificate înainte de afişare. O serie de operatori simpli v-au fost prezentaţi în secţiunile anterioare (spre exemplu suma, produsul etc.). Dacă, spre exemplu, dorim să afişam numele studenţilor împreună cu bursa pe care aceştia la care au fost adăugaţi 10ron (pentru că au primit o sponsorizare suplimentară), comanda ce va afişa sumele finale pe care studenţii le vor primi este:

```
SELECT nume, prenume, bursa + 10 FROM studenti;
```

Observaţie: Precedenţa operatorilor aritmetici este cea cunoscută din matematica de gimnaziu: înmulţirea, împărţirea, adunarea, scăderea. Pentru a schimba ordinea de executare a operaţiilor, ca şi în alte limbaje cunoscute, utilizaţi parantezele rotunde.

Exerciţiu: Afişaţi jumătate din valoarea bursei studenţilor la care a fost adăugat în prealabil o sumă de 100 ron. Ţineţi cont de precedenţa operatorilor.

Valori NULL

Uneori, anumite câmpuri pot fi lăsate fără valoare (de exemplu pentru studenţii care nu au bursă, în loc să trecem valoarea 0, nu va fi trecută nici o valoare - uneori chiar şi valoarea 0 poate avea o semnificaţie interpretabilă: spre exemplu, aceasta ar putea însemna că studentul ia de fapt bursă dar că această bursă nu poate fi acoperită din punct de vedere financiar [insert sad student image here]). Neinserarea unei valori într-un câmp va avea ca efect trecerea în acel câmp a valorii NULL (sau ” - şirul vid ce este reprezentat prin două apostroafe). Dacă valoarea unui câmp este nulă şi acest câmp este utilizat într-o expresie aritmetică, rezultatul va fi de asemenea o valoare nulă (se poate observa pentru cazul studenţilor ce nu aveau bursă.

Concatenarea şirurilor de caractere

Pentru concatenarea a două şiruri de caractere se va utiliza operatorul || (două bare verticale). Prin intermediul acestuia pot fi concatenate două şiruri de caractere oarecare, un şir de caractere cu un număr, un şir de caractere cu valoarea dintr-o coloană, etc. Rezultatul este o nouă coloană. In cazul tipurilor de date de tip char, în baza de date sunt reţinute şi spaţiile necesare ajungerii la un număr de caractere egal cu dimensiunea câmpului (de exemplu dacă într-un câmp de tip char de 10 caractere s-au introdus doar 3 caractere, automat sunt stocate 7 spaţii pentru a completa cele 10 caractere). In cazul datelor de tip varchar2 nu sunt introduse spaţii suplimentare. Să vedem câteva exemple:

```
SELECT 'Studentul '||nume||' '||prenume||' este inmatriculat in anul '|| an ||'.' AS info
FROM studenti;
```

Observaţie: Toate şirurile de caractere din SQL sunt scrise între simbolurile apostrof. Singurul loc în care se folosesc ghilimelele, pentru a grupa mai multe caractere, este atunci când dorim să creăm un alias pentru numele unei coloane.

Afişarea anumitor rânduri

Deja am filtrat (într-un exemplu anterior) anumite rânduri - duplicatele - utilizând cuvantul cheie DISTINCT. Pentru a restricţiona şi mai mult datele ce vor fi afişate, în cadrul operaţiilor de tip SELECT, se poate utiliza clauza WHERE.

Clauza WHERE este opţională şi, atunci când va fi utilizată, va fi mereu scrisă după numele tabelului din care se face selecţia şi va fi urmată de o condiţie. Vor fi selectate doar acele rânduri din tabel care satisfac condiţia impusă de clauza WHERE. Spre exemplu, am putea selecta doar studenţii din anul 1 executând comanda:

```
SELECT nume, prenume FROM studenti WHERE an = 1;
```

Condiţiile ce sunt utilizate în clauza WHERE sunt de fapt expresii booleene. Oricare dintre operatorii de comparare descrişi în secţiunile anterioare pot fi utilizaţi. Următorul exemplu va afişa toţi studenţii bursieri.

```
SELECT nume, prenume FROM studenti WHERE bursa IS NOT NULL;
```

Condiţia ce urmează după clauza WHERE poate fi una simplă (ca cele date până acum ca exemplu) sau pot fi compuse din mai multe condiţii simple interconectate cu operatorii logici descrişi în secţiunea anterioară. Spre exemplu, dacă dorm să îi găsim pe acei studenţi din anul 1 care iau bursă, putem executa comanda:

```
SELECT nume, prenume FROM studenti WHERE bursa IS NOT NULL AND an = 1;
```

Observaţie: în evaluarea unei expresii logice compuse, precedenţa operatorilor logici este: NOT, AND, OR.

Atunci când în clauza WHERE se doreşte lucru cu şiruri de caractere, un operator foarte util este operatorul LIKE. Prin intermediul acestuia putem forma expresii regulate care să ne permită selectarea anumitor informaţii chiar dacă acestea nu sunt cunoscute în întregime. În cadrul acestor expresii regulate, caracterul "%" (la sută) are semnificaţia unui grup de caractere, iar caractereul "_" (underscore) semnifică un singur caracter. Spre exemplu, dacă dorim să selectăm toţi studenţii a căror nume se termină în "escu", putem executa comanda:

```
SELECT nume, prenume FROM studenti WHERE nume LIKE '%escu';
```

Observaţie: atunci când doriţi să utilizaţi efectiv caracterul % sau caracterul _, acestea vor fi prefixate (escaped) cu caracterul \ (backslash).

Ordonarea rezultatelor

Pentru a ordona datele selectate, se va utiliza clauza ORDER BY urmată de numele (sau numarul) câmpului după care se doreşte a fi făcută ordonarea (sau de către o funcţie aplicată unui câmp). Optional, după numele câmpului (funcţiei) poate fi adăugat unul dintre cuvintele cheie ASC, DESC pentru a forţa sortarea în mod ascendent sau descendent (predefinit, sortarea se realizează în mod ascendent).

Spre exemplu, dacă dorim să sortăm studenţii în ordinea anilor de studiu, se poate executa următoarea comandă:

```
SELECT nume, prenume, an FROM studenti ORDER BY an ASC;
```

Atunci când sunt mai mulţi studenţi în acelaşi an, am putea dori să îi ordonăm şi după grupa din care fac parte. După clauza ORDER BY putem introduce oricâte câmpuri împreună cu opţiunea ce indică tipul sortării, despărţite prin virgulă. Spre exemplu:

```
SELECT nume, prenume, an FROM studenti ORDER BY an ASC, grupa DESC;
```

Putem construi interogări care să conţină ambele clauze învăţate azi: WHERE şi ORDER BY. Prima care va fi scrisă va fi clauza WHERE urmată de ORDER BY. Spre exemplu, pentru a selecta toţi studenţii din anul 1 în

ordinea grupelor vom executa comanda:

```
SELECT nume, prenume, grupa FROM studenți WHERE an=1 ORDER BY grupa ASC;
```

Deoarece numele de familie al studenţilor este al doilea câmp din tabel, ordonarea lor după acesta poate fi facută prin comanda:

```
SELECT * FROM studenți ORDER BY 2;
```

Exerciții

1. Scrieți o interogare pentru a afișa numele, prenumele, anul de studiu si data nașterii pentru toți studenții. Editați în SQL*Plus și executați. Salvați apoi interogarea intr-un fișier numit p1.sql.
2. Scrieți și executați o interogare pentru a afișa în mod unic valorile burselor.
3. Încărcați fișierul p1.sql în buffer. Dați fiecărei coloane din clauza SELECT un alias. Executați înterogarea.
4. Afișați numele concatenat cu prenumele urmat de virgulă și anul de studiu. Ordonați crescător după anul de studiu. Denumiți coloana “Studenți pe ani de studiu”.
5. Afișați numele, prenumele și data de naștere a studenților născuți între 1 ianuarie 1995 si 10 iunie 1997. Ordonați descendent după anul de studiu.
6. Afișați numele și prenumele precum și anii de studiu pentru toți studenții născuți în 1995.
7. Afișați studenții (toate informațiile pentru aceștia) care nu iau bursă.
8. Afișați studenții (nume și prenume) care iau bursă și sunt în anii 2 și 3 de studiu. Ordonați alfabetic ascendent după nume și descendent după prenume.
9. Afișați studenții care iau bursă, precum și valoarea bursei dacă aceasta ar fi mărită cu 15%.
10. Afișați studenții al căror nume începe cu litera P și sunt în anul 1 de studiu.
11. Afișați toate informațiile despre studenții care au două apariții ale literei “a” în prenume.
12. Afișați toate informațiile despre studenții al căror prenume este “Alexandru”, “Ioana” sau “Marius”.
13. Afișați studenții bursieri din semianul A.
14. Afișaţi numele şi prenumele profesorilor a căror prenume se termină cu litera "n" (întrebare capcană).

Adus de la „http://85.122.23.37/BD/index.php?title=Laboratorul_2&oldid=224”
