Baze de Date

Cap. 3. SQL. Proiecție. Selecție. Join

Textbook: Ramakrishnan, Gehrke, “Database Management Systems”, McGraw Hill, Cap. 5

2023 UPT Conf.Dr. Dan Pescaru

Operatorii algebrei relaționale

1. Operatorii algebrei relaționale

- Proiecție (Π) – selectează doar atributele specificate în lista de proiecție
- Selecție (σ) – selectează rândurile care satisfac condiția de selecție
- Produs Cartezian ($\times$) – combină două relații R_1 și R_2 incluzând toate perechile ($t_1 \in R_1$ și $t_2 \in R_2$)
- Join ($\bullet$) – $\times$ cu selecție perechi $\sigma_{cond\_join}(R_1 \times R_2)$
- Reuniune ($\cup$) – toate înregistrările din R_1 și din R_2 dar fără duplicate
- Intersecție ($\cap$) – înregistrările comune din R_1 și R_2
- Diferență (/) – înregistrările din R_1 care nu sunt și în R_2

BD folosită în exemple (Port)

Tabelă Marinar

mid	nume	rang	varsta
22	Ion	7	45
31	Horatiu	1	33
58	Oana	8	54
71	Constantin	9	55

Tabelă Barca

bid	nume	culoare
101	Cleo	Albastra
102	Triton	Rosie
103	Poseidon	Verde

Tabelă Rezervare

rid	Mid	Bid	dată
22	31	101	10/03/2022
231	58	103	18/10/2022
71	31	101	22/10/2022

SQL – Proiecție

1. Selectează doar atributele (coloanele) specificate în lista de proiecție:

   SELECT [DISTINCT] listă_proiecție

   FROM Tabela;

   - listă_proiecție: listă de atribute separată prin virgulă sau markerul '*' pentru proiecție 1:1
   - DISTINCT: implicit nu șterge duplicatele!
     - Când este necesar?

2. Ex.: SELECT mid, nume FROM Marinar;

   SELECT DISTINCT rang FROM Marinar;

Proiecție – redenumire câmpuri

1. Aliasurile sunt utile pentru a evita ambiguitatea:
   - pentru câmpuri calculate
     - poate conține apeluri de funcții de bibliotecă (funcțiile sunt dependente de sistem) + funcții de agregare
   - când listă_proiecție include câmpuri cu același nume din tabele diferite (ex. la Join)

     SELECT [DISTINCT] $câmp_1$ AS $alias_1$, $câmp_2$ AS $alias_2$,…, exp_1 AS $alias_{e1}$, …
     FROM Tabela;
   - E.g.: SELECT bid AS ID_barca,
     year(data) AS Anul_rezervarii
     FROM Rezervare; // -- syntaxă MySQL

Oracle SQL – Proiecție

1. Sintaxă dialect Oracle SQL:

SELECT [ALL | DISTINCT | UNIQUE]
câmp$_1$ AS alias$_1$, ..., exp$_1$ AS nume_exp$_1$, ...
FROM Tabela;

- Se poate folosi ROWNUM pentru numerotare rânduri în rezultat

2. Ex.:

SELECT ROWNUM AS nrcrt, bid, nume AS denumire
FROM Barca;

MySQL SQL – Proiecție

1. Sintaxă MySQL:

```
SELECT [ALL | DISTINCT | DISTINCTROW]
        câmp1 AS alias1, ..., exp1 AS nume_exp1, ...
        FROM Tabela;
```

- Pentru numerotare înregistrări:

```
SET @r=0;
SELECT @r:=@r+1 AS nrcrt, bid, nume
        FROM Barca;
```

SQL – Selecție

1. Selectează un subset de înregistrări bazată pe o condiție:

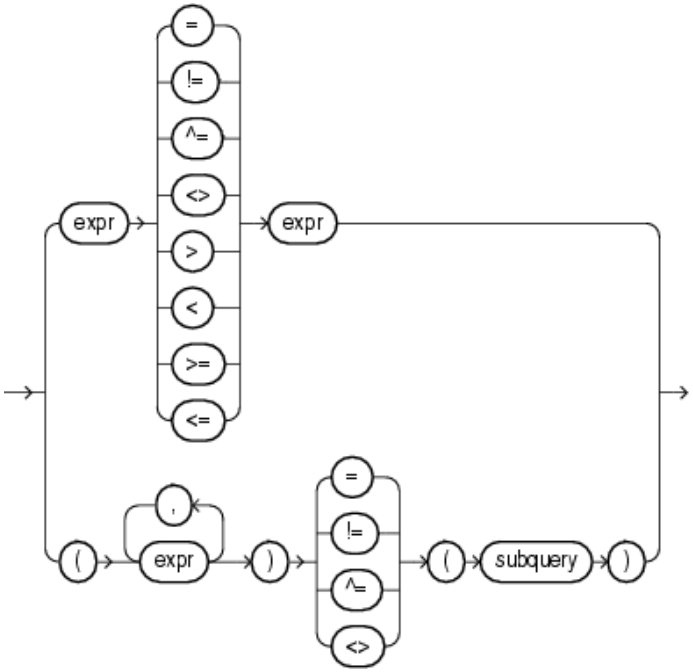
   SELECT listă_proiecție FROM tabla
   WHERE condiție;

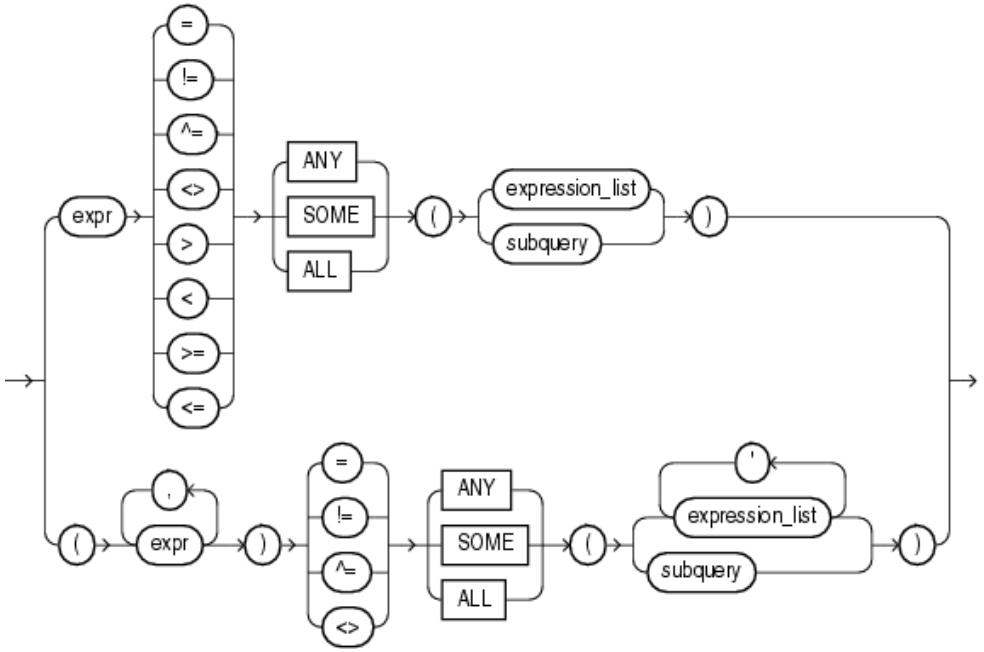
   - Sau * pentru selecție pură (fără proiecție)
   - Pot exista duplicate?
   - Condiție selecție: expresii logice cuprinzând operatori AND, OR, NOT și funcții SQL

   Ex.: SELECT mid, nume FROM Marinar
   WHERE varsta>30;

Oracle SQL – Selecție

1. Operatori de comparare





Oracle SQL – condiții de selecție

1. Ex.

```
SELECT * FROM Marinar
    - WHERE rang=7;
    - WHERE rang<3 AND varsta>25 OR vasta>=50;
    - WHERE rang IN (1,2,7);
    - WHERE rang = ANY (2,5,6);
    - WHERE rang NOT IN (1,3);
    - WHERE varsta > ANY (18, 13, 42, 27);
    - WHERE varsta > ALL (18, 13, 42, 27);
    - WHERE nume LIKE 'M_ _ _%'
    - WHERE varsta IS NULL
```

Oracle SQL – funcții (I)

1. Funcții de bibliotecă

- abs(*n*) – valoarea absolută (fără semn)
- round(*n*) – rotunjește număr
- trunc(*n*) – trunchiază număr
- concat(*sir1*,*sir2*) – concatenează șiruri (sau '||')
- lower/upper(*sir*) – modifică *șir* în litere mici/mari
- substr(*sir*, *pos*, [*lng*]) – extrage un subșir dintr-un *șir* de la poziția *pos* eventual de lungime *lng*
- instr(*subsir*, *sir*[, *startpos*, *aparN*]) – caută *subsir*
- trim(*sir*) – șterge spațiile din față/spate
- length(*sir*) – returnează lungimea șirului

Oracle SQL – funcții (II)

1. Funcții de bibliotecă

- sysdate, current_timestamp – data curentă

Ex. SELECT sysdate FROM Dual;

- extract(*spec*, *d*) – extrage o parte din dată (spec ∈{year, month, day, hour, minute, second})
- to_char(data, format) – convertește la șir (format include Y, M, D, HH24, Mi, SS)
- nvl(*expr1*,*expr2*) returnează *exp2* dacă *exp1* este NULL

Ex. SELECT nvl(nume, 'necunoscut') FROM Marinar;

- case when *exp1* then *exp2* else *exp3* end

MySQL SQL – condiții selecție

1. Ex.

```
SELECT * FROM Marinar
    - WHERE rang=7;
    - WHERE rang=7 AND varsta>=25;
    - WHERE rang IN (1,2,7);
    - WHERE rang BETWEEN 3 AND 7;
    - WHERE varsta IS NULL;
    - WHERE varsta<=>NULL; //egalitate NULL-safe
SELECT 1 <=> 1, NULL<=>NULL, 1 <=> NULL;
SELECT 1 = 1, NULL = NULL, 1 = NULL;
```

MySQL SQL – funcții (I)

1. Funcții de bibliotecă:
   - abs(*n*) – valoarea absolută
   - concat(*str1*,*str2*,…) – concatenare șiruri
   - curtime() – data curentă pe server
   - day() – ziua din lună (0-31)
   - dayofyear() – ziua din an (1-366)
   - format(*n*, *decPlaces*) – formatare grupe
   - if(*exp1*,*exp2*,*exp3*) – ca și C: exp1?exp2:exp3
   - ifnull(*exp1*,*exp2*) – dacă *exp1*=NULL atunci *exp2*
   - left(*str*, *len*) – subșirul din stânga
   - length(*str*) – lungimea șirului

MySQL SQL – funcții (II)

1. Funcții de bibliotecă:
   - locate(*subsir*, *sir*, [*s_pos*]) – poziția primei apariției subșirului în șir
   - lower(*sir*) - transformă în litere mici
   - mid(*sir*, *pos*, *len*) – un substring începând cu poziția specificată
   - month(*d*) – luna din data specificată
   - now() – data curentă, include timpul
   - password(*sir*) – criptare ireversibilă a sir
   - rand() – număr aleator $\in$[0.0…1.0]
   - trim(*sir*) - șterge spațiile din față/spate din sir

SQL Exemple selecție

- Extrageți rangul tuturor marinarilor care au numele în prima jumătate a alfabetului. Clasificați marinarii în juniori și seniori dacă vârsta este sub/peste 31 de ani
  - Soluție Oracle SQL:

```
SELECT  ('Marinarul ' ||
    case when varsta<31 then 'junior' else
    'senior ' end || nume || ' are rangul ' ||
    to_char(rang) || '.') AS marinar,
    nvl(varsta, 0) AS varsta
    FROM Marinar
    WHERE nume < 'L';
```

SQL – Ordonarea rezultatului

1. ORDER BY este utilizat pentru ordonarea rezultatului după unul sau mai multe atribute/expresii
2. Ordinea ascendentă (ASC) este implicită

   SELECT *coloană1*,*coloană2*,…

   FROM *Tabelă*

   ORDER BY $coloană_i$, $coloană_j$ [ASC|DESC];

3. Ex.: SELECT * FROM Marinar

   ORDER BY rang, varsta DESC;

SQL – Produs cartezian

1. Nu foarte întâlnit în aplicații
2. Sintaxa:
   - Tabele multiple în FROM (utile: aliasuri pentru tabele)

     SELECT t_1.*, t_2.*

     FROM *Tabela*$_1$ t_1, *Tabela*$_2$ t_2;
3. Ex.

   SELECT m.*, r.*

   FROM Marinar m, Rezervare r;

SQL – JOIN

1. SQL JOIN este folosit pentru a combina rânduri din două sau mai multe tabele pe baza legăturilor logice. De obicei: t_1.PK=t_2.FK
2. Două sintaxe posibile:
   - JOIN implicit: tabele multiple în FROM; condiția de join înglobată în clauza WHERE

     SELECT t_1.*col*$_1$,…,t_2.*col*$_1$,…

     FROM *tabela*$_1$ t_1, *tabela*$_2$ t_2 …

     WHERE t_1.*col*$_i$=t_2.c_j [AND cond_selectie];
   - JOIN explicit: tabelele și condițiile ambele în FROM

Tipuri de JOIN

1. INNER JOIN - returnează toate rândurile cu corespondent în ambele tabele. Sintaxa implicită implică INNER JOIN
2. LEFT (OUTER) JOIN - returnează toate rândurile din tabelul din stânga și cele cu corespondent din tabelul din dreapta (cu NULL în câmpurile fără corespondent)
3. RIGHT (OUTER) JOIN – invers fată de LEFT
4. FULL (OUTER) JOIN - returnează toate rândurile din ambele tabele indiferent dacă există sau nu potrivire (LEFT $\cup$ RIGHT)

INNER JOIN

1. INNER JOIN – sintaxa explicită

```
SELECT list_proiectie
    FROM tabela1 [INNER] JOIN tabela2
    ON tabela1.PK=tabela2.FK …
    [WHERE conditie_selectie…];
```

2. Ex.

```
SELECT m.nume, m.rang, r.data, b.nume AS barca
  FROM ((Marinar m INNER JOIN Rezervare r ON
    m.mid=r.mid) INNER JOIN Barca b ON r.bid=b.bid)
  WHERE m.varsta>25;
```

LEFT JOIN

1. LEFT [OUTER] JOIN – sintaxă

```
SELECT list_proiectie
    FROM tabela1 LEFT JOIN tabela2
    ON tabela1.PK=tabela2.FK …
    [WHERE conditie_selectie …];
```

2. Ex.

```
SELECT m.nume, m.rang, r.data, b.nume AS barca
  FROM ((Marinar m LEFT JOIN Rezervare r ON
    m.mid=r.mid) LEFT JOIN Barca b ON r.bid=b.bid)
  WHERE m.varsta>25;
```

RIGHT JOIN

1. RIGHT [OUTER] JOIN – sintaxă

```
SELECT list_proiectie
    FROM tabela1 RIGHT JOIN tabela2
    ON tabela1.PK=tabela2.FK …
    [WHERE conditie_selectie …];
```

2. Ex.

```
SELECT m.nume, m.rang, r.data, b.nume AS barca
  FROM ((Marinar m LEFT JOIN Rezervare r ON
   m.mid=r.mid) RIGHT JOIN Barca b ON r.bid=b.bid)
   // WHERE s.age>25; //contradicție – de ce ?
```

FULL JOIN

1. FULL [OUTER] JOIN – sintaxă

```
SELECT list_proiectie
    FROM tabela1 FULL JOIN tabela2
    ON tabela1.PK=tabela2.FK …
    [WHERE conditie_selectie …];
```

2. Obs: nu există în toate DBMS (ex. MySQL)
3. Ex.

```
SELECT m.nume, m.rang, r.data, b.nume AS barca
  FROM ((Marinar m FULL JOIN Rezervare r ON
   m.mid=r.mid) FULL JOIN Barca b ON r.bid=b.bid)
   // WHERE s.age>25; // la fel ?
```

SELF JOIN

1. O singură tabelă implicată (legături recursive). Poate fi de tip INNER, LEFT, RIGHT or FULL JOIN, dar …
2. Ex.

Persoana

id	nume	id_tata	id_mama
B012	John	B026	NULL
B026	Horace	NULL	G018
G114	Anna	B026	NULL
G018	Sara	NULL	NULL

```
SELECT p.nume, t.nume AS tata, m.nume AS mama
      FROM ((Persoana p LEFT JOIN Persoana t
            ON p.id_tata=t.id) LEFT JOIN Persoana m
            ON p.id_mama=m.id);
// Obs: dar INNER+INNER JOIN sau LEFT+RIGHT JOIN ?
```

Oracle SQL JOIN

1. Implementează FULL OUTER JOIN
2. Definește un operator pentru outer join (+) care poate fi folosit în condiția de WHERE pentru a specifica LEFT/RIGHT JOIN in loc de INNER JOIN. Nu poate înlocui FULL JOIN
3. Ex. Pentru LEFT JOIN

```
SELECT m.nume, m.rang, r.data
    FROM Marinar m, Rezervare r
    WHERE m.mid = r.mid(+);
```

Obs: pentru RIGHT JOIN: m.mid(+) = r.mid

MySQL JOIN

1. Nu implementează FULL OUTER JOIN. Poate fi obținut cu UNION între LEFT și RIGHT JOIN
2. USING poate înlocui ON dacă PK și FK au același nume
   - Ex. SELECT m.*,r.*

     FROM Marinar m INNER JOIN Rezervare r

     USING (mid);
3. STRAIGHT_JOIN este similar cu JOIN, dar tabela din stânga ca fi citită înaintea celei din dreapta. Acest lucru poate corecta ordinea pentru optimizarea execuției