Baze de Date

Cap. 5. SQL – Agregare Date

Textbook: Ramakrishnan, Gehrke, “Database Management Systems”, McGraw Hill, C15

2023 UPT Conf.Dr. Dan Pescaru

Funcții de agregare

1. Funcțiile de agregare au ca intrare o mulțime de valori și returnează o singură valoare scalară
2. Sunt folosite pentru a calcula date statistice despre informațiile din tabelă
3. Majoritatea funcțiilor de agregare (toate cu excepția MIN, MAX și COUNT) primesc o singură expresie numerică ca parametru de intrare

Proprietățile funcțiilor de agregare

1. Principalele proprietăți ale funcțiilor de agregare sunt:
   - Operează pe o singură coloană sau expresie dintr-un grup de coloane
   - Returnează o singură valoare pentru un grup de înregistrări
   - Utilizabile doar în lista de proiecție, în subinterogări și în clauza HAVING. NU se pot utiliza în clauza WHERE !!!
   - Rezultatul este anonim – se va redenumi corespunzător scopului printr-un alias

Limitări la utilizarea funcțiilor de agregare

1. Lista de proiecție va conține doar funcții de agregare și valori distincte (care includ chei):
   - Expresii care generează o singură valoare pentru grupul de înregistrări din intrare (aceste expresii trebuie incluse în clauza GROUP BY)

Ex. SELECT nume, ~~COUNT(*) FROM~~ Marinar;

- Obs. Nu se verifică de unele versiuni de MySQL, dar nu are semnificație logică!

Controlul setului de intrare

1. Pentru a controla setul de intrare se pot folosi:
   - DISTINCT: ia în considerare numai valori distincte din setul de intrare
   - ALL (implicit): ia în considerare toate valorile, inclusiv duplicate
2. Ex.: SELECT COUNT(DISTINCT col) AS Nr
   FROM Tabelă

Funcții de agregare

A. COUNT: întoarce numărul de înregistrări
B. SUM: returnează suma valorilor expresiilor/coloanelor din setul de intrare
C. AVG: returnează media valorilor expresiilor/coloanelor din setul de intrare
D. MIN, MAX: returnează minimul/maximul expresiilor/coloanelor din setul de intrare
E. VARIANCE: măsoară împrăștierea unui set de numere în jurul mediei lor
F. STDDEV: returnează deviația standard a unui set de valori numerice

COUNT()

1. Returnează numărul de valori/valori distincte din setul de intrare
2. Parametrii de intrare
   - [DISTINCT/ALL] coloană/expresie: nu ia în calcul valorile Null și ia/nu ia în calcul duplicatele
   - * : ia în calcul toate înregistrările din intrare indiferent dacă sunt Null sau duplicate

```
SELECT COUNT(DISTINCT ClientID) AS NrDeClienti
        FROM Factura;
```

SUM()

1. Returnează suma valorilor coloanei/expresiei de intrare
2. Ignoră valorile Null (tratate ca 0). Dacă toate valorile sunt Null returnează Null

- Ex.
```
SELECT SUM(salar) AS buget
        FROM Angajat
        WHERE id_departament=101;
```

AVG()

1. Returnează media aritmetică a valorilor din coloana/expresia de intrare
2. Ignoră valorile Null (nu sunt tratate ca 0). Dacă toate sunt Null, returnează Null
3. Atenție la utilizarea DISTINCT și ALL în AVG

- Ex.

```
SELECT SUM(vârsta)/COUNT(*) AS vmSUM,
                    AVG(vârsta)  AS vmAVG
        FROM Marinar;
```

MIN(), MAX()

1. MIN() - returnează cea mai mică valoare
2. MAX() - returnează cea mai mare valoare
3. Ignoră valorile Null (nu sunt tratate ca 0). Dacă toate sunt Null, întoarce Null
4. La șiruri de caractere: comparația este case sensitive / insensitive la Oracle/MysQL
5. Obs: DISTINCT nu are nici un efect

- Ex.

```
SELECT MIN(vârsta) AS vmin, MAX(vârsta)  AS vmax
        FROM Marinar;
```

VARIANCE()

1. Returnează varianța unui set de numere (măsoară cât de departe setul de numere este împrăștiat în jurul valorii medii). Se calculează ca media pătratelor diferențelor valorilor față de valoarea medie

$$\sigma = \text{SUM}(d^2)/\text{COUNT}(d) - (\text{SUM}(d)/\text{COUNT}(d))^2$$

1. Returnează 0 dacă setul conține doar o înregistrare
2. Ignoră valorile Null (nu sunt tratate ca 0). Dacă toate sunt Null, returnează Null
3. Obs: DISTINCT are efect.

```
SELECT VARIANCE(vârsta) AS variV FROM Marinar;
```

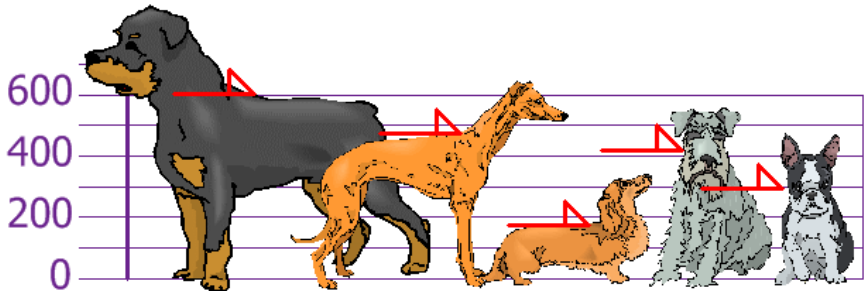
STDDEV()

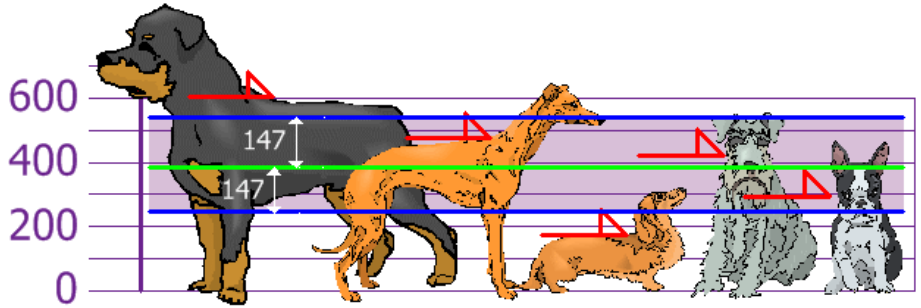
1. Returnează deviația standard a unui set de valori (definește un interval de ”Normalitate” în jurul mediei valorilor)
2. Se calculează ca și rădăcină pătrată din Varianță
3. Returnează 0 dacă intrarea conține un singur rând
4. Ignoră valorile Null, DISTINCT are efect

- Ex.

```
SELECT STDDEV(vârsta) AS devStdVarsta
                        FROM Marinar;
```

Deviația standard. Interpretare





1. Media = 394 mm
2. Varianța = 21704
3. Deviația standard = 147 mm
4. Înălțime standard (”normală”) = [247.. 541] mm
5. Foarte-mari: 600 mm, Foarte-mici: 170 mm

(Ref: http://www.mathsisfun.com/)

Gruparea datelor (I)

1. De ce: Calculați count(*) pentru fiecare rang (dar pentru fiecare vârstă?)
2. GROUP BY: grupează datele de intrare și produce o singură valoare a funcției de agregare pentru fiecare grup
3. Ex.

```
SELECT   idDep AS departament,
                    MAX(salar) AS salMax
      FROM Angajat
      GROUP BY idDep;
```

Gruparea datelor (II)

1. Lista de proiecție pot să conțină doar funcții de agregare și eventual expresii cu valori unice pentru întregul grup
2. GROUP BY poate conține mai mult de un singur criteriu (însă nu ierarhic)
3. Lista de proiecție trebuie să conțină toate expresiile GROUP BY care nu sunt agregări

- Ex. SELECT vârsta, rang, COUNT(*) AS nrMar
  FROM Marinar
  GROUP BY rang, vârsta; // sau vârsta, rang?

Gruparea datelor: ordonare grupuri

- ORDER BY poate fi utilizată pentru ordonarea grupurilor (ierarhizare)

```
SELECT rang, vârsta, COUNT(*) AS nrMar
    FROM Marinar GROUP BY rang, vârsta
                    ORDER BY rang, vârsta;
```

vs.

```
SELECT rang, vârsta, COUNT(*) AS nrMar
    FROM Marinar  GROUP BY rang, vârsta
                    ORDER BY vârsta, rang;
```

Gruparea datelor: adăugare de câmpuri

- Toate expresiile unice care se adaugă în lista de proiecție se vor adăuga și în GROUP BY

```
SELECT d.idDep, d.nume, AVG(a.salar) AS salMed
    FROM (Angajat a INNER JOIN Departament d
                ON a.idDep=d.idDep)
    GROUP BY d.idDep;
```

```
SELECT d.idDep, d.nume, AVG(a.salar) AS salMed
    FROM (Angajat a INNER JOIN Departament d
                ON a.idDep=d.idDep)
    GROUP BY d.idDep, d.nume;
```

HAVING (I)

1. Se utilizează cu GROUP BY pentru a restricționa grupurile din rezultat
2. Câteodată înlocuibil cu WHERE

```
SELECT rang, COUNT(*) AS nrMar
    FROM Marinar
    GROUP BY rang HAVING rang>3;

SELECT rang, COUNT(*) AS nrMar
    FROM Marinar WHERE rang>3
    GROUP BY rang;
```

HAVING (II)

1. De obicei mai eficient decât WHERE
2. Nu se poate înlocui în orice situație cu WHERE (ex. când conține o agregare)

```
SELECT rang, COUNT(*) AS nrMar
    FROM Marinar
    GROUP BY rang
    HAVING COUNT(*)>1;
```

~~(WHERE COUNT(*)>1)~~

Combinarea funcțiilor de agregare

1. Funcțiile de agregare pot fi combinate în expresii aritmetice
2. Ex.

```
SELECT idDep AS departament
        (MAX(salar) - MIN(salar)) AS
                                intervalSal
    FROM Angajat
    GROUP BY idDep;
```

Funcții de agregare imbricate

1. Funcțiile de agregare pot fi aplicate imbricat (suportat doar de unele sisteme ex. Oracle)
2. Ex.

```
SELECT idDep, AVG(MAX(salar)) AS medieSalDep
    FROM Angajat
    GROUP BY idDep; // rezultă o singură valoare
```

- Se efectuează prima dată agregarea internă (MAX(salar)) pentru fiecare grup generat de GROUP BY (fiecare idDep), apoi se aplică cea de a doua agregare peste rezultate

Alternativă la imbricarea agregărilor

1. Pentru MySQL se pot folosi subinterogări
2. Ex.

```
SELECT AVG(depsal.maxsal)
   FROM
        ( SELECT MAX(salar) AS maxsal
                FROM Angajat
                GROUP BY idDep
         ) depsal;
```

Agregare și subinterogări (I)

1. Funcțiile de agregare pot fi folosite în subinterogări
2. Ex. Toți marinarii care au vârsta mai mare decât media vârstelor marinarilor

```
SELECT *
    FROM Marinar
    WHERE vârsta >
        (SELECT AVG(vârsta) FROM Marinar);
```

Agregare și subinterogări (II)

1. Ex. Cei mai bătrâni marinari (vârsta lor nu o cunoaștem în prealabil)

```
SELECT *
    FROM Marinar
    WHERE vârsta =
        (SELECT MAX(vârsta) FROM Marinar);
```

Agregare directă utilizând OVER

1. Clauza OVER permite definirea unei ferestre. Agregarea se va aplica peste grupurile definite de parametrul PARTITION BY din clauza OVER. Pentru toata tabela se poate folosi PARTITION BY 1
2. Generează un sumar pentru fiecare rând (în loc de fiecare grup definit prin GROUP BY)

```
SELECT mid, nume, varsta, rang,
    AVG(varsta) OVER (PARTITION BY rang)
        AS varstaMediePerRang
    FROM Marinar
    ORDER BY nume;
```

Ierarhizarea utilizând OVER

1. Funcția ROW_NUMBER() împreună cu clauza OVER permite ierarhizarea și ordonarea rezultatelor după poziția înregistrării curente în cadrul grupului considerat (ferestrei)

```
SELECT rang, nume, varsta,
        ROW_NUMBER() OVER (PARTITION BY
                rang order by varsta) as nrrang,
        AVG(varsta) OVER (PARTITION BY
                rang) as vmedrang
    FROM Marinar
```

Exemple (I)

1. Problemă:

   - Să se listeze toți marinarii împreună cu numărul total de rezervări făcute de fiecare în parte.

Exemple (I)

Răspuns 1:

```
SELECT m.*,
    (SELECT count(*) FROM Rezervare r
            WHERE r.mid=m.mid) AS nrRez
     FROM Marinar m ORDER BY nrRez;
```

Răspuns 2:

```
SELECT m.mid, m.nume, COUNT(*) AS nrRez
    FROM Marinar m INNER JOIN  Rezervare r
                        ON m.mid=r.mid
    GROUP BY m.mid,m.nume ORDER BY nrRez;
```

Obs: greșit pentru 0 rezervări (De ce LEFT JOIN este de asemenea greșit? Soluție (rezolvare NULL)?

Exemple (II)

1. Problemă :

   - Găsiți vârsta celui mai tânăr marinar peste de 20 de ani pentru fiecare rang care are cel puțin 3 marinari de orice vârsta.

Exemple (II)

1. Răspuns

```
SELECT m1.rang, MIN(m1.varsta) AS tanar
    FROM Marinar m1
    WHERE m1.varsta > 20
    GROUP BY  m1.rang
    HAVING 1< (SELECT COUNT(*)
                    FROM Marinar m2
                    WHERE m1.rang=m2.rang);
```

Obs: De ce nu este corect HAVING COUNT(*)>1?