9. Gruparea datelor

- Care sunt funcțiile de grup și cum sunt ele folosite
- Cum se pot grupa datele într-o tabela
- Care sunt regulile de folosire a clauzei GROUP BY
- Cum se pot filtra grupurile folosind clauza HAVING
- Care este diferența dintre clauzele WHERE și HAVING
- În ce ordine se execută clauzele WHERE, GROUP BY și HAVING

9.1. Funcţii de grup

- Într-un capitol anterior am discutat despre funcţiile singulare, adică despre funcţiile care operează la un moment dat asupra unei singure înregistrări.
- Este acum momentul să discutăm despre funcţiile de grup, care returnează o singură valoare pentru un **grup sau set** de linii dintr-un tabel. Puteţi calcula cea mai mare valoare dintr-un set de valori, puteţi determina numărul de înregistrări ce respectă o anumită condiţie etc.
- Pentru exemplificarea acestor funcţii vom folosi tabela **VOTURI** şi tabela **JUDEŢE** care conţin următoarele date

Tabela **JUDETE**

Cod_judeţ	Judet	Număr_alegători
B	Bucureşti	1750192
IS	Iaşi	650029
SB	Sibiu	363380

Tabela **VOTURI**

Judeţ	Candidat	Număr_voturi
B	1	347016
B	2	1552
B	3	1374
IS	1	196508
IS	2	1038
IS	3	1267
SB	1	65084
SB	2	561
SB	3	533
B	4	96744
B	5	25656
B	6	13361
IS	4	35784
IS	5	5558
IS	6	4094
SB	4	19937
SB	5	4323
SB	6	2366
B	7	25937
B	8	4619
B	9	4323
IS	7	3682
IS	8	1291
IS	9	327
SB	7	4225
SB	8	765
SB	9	3797
B	10	2037
B	11	22687
B	12	514366
IS	10	1312
IS	11	3781
IS	12	12184
SB	10	660
SB	11	3768
SB	12	105993
SB	13	100
B	13	(null)
IS	13	(null)

- Principalele funcţii de grup.
- **COUNT(x)** – determină numărul de valori ale lui **x**. Funcţia, ca de altfel toate funcţiile de grup ignoră câmpurile completate cu **NULL**, adică va număra doar valorile nenule ale lui x.
- De exemplu, comanda

  SELECT COUNT(JUDET), COUNT(numar_voturi)

  FROM voturi
- va afişa numărul total de înregistrări din tabelă, **39** (câmpul **JUDET** nu are nici o valoare **NULL**) precum şi numărul de linii pentru care câmpul **numar_voturi** este nenul, adică **37**, ultimele două linii din tabel având valoare **null** în câmpul **numar_voturi**.

COUNT(JUDET)	COUNT(NUMAR_VOTURI)
39	37

- Funcţia **COUNT** poate fi folosită în combinaţie cu clauza **DISTINCT**, pentru a număra doar valorile distincte dintr-un domeniu. De exemplu dacă dorim să ştim pentru câte judeţe avem rezultatele votării în tabela noastră, vom folosi comanda:

SELECT count(distinct judet)

FROM voturi

- Se va obţine valoarea **3**, întrucât avem doar **3** judeţe înregistrate (Bucureşti, Iaşi, Sibiu).

COUNT(DISTINCTJUDET)
3

- Exemplu:

SELECT count(distinct candidat), count(candidat)

FROM voturi

- Evident primul apel de funcţie afişează valoarea **13** , deoarece există **13** candidaţi pentru care au fost exprimate voturi, iar a doua comandă afişează valoarea **39**, adică exact numărul de linii din tabel deoarece toate liniile au completat câmpul **candidat**.

COUNT(DISTINCTCANDIDAT)	COUNT(CANDIDAT)
13	39

- **MAX(x)** – determină valoarea maximă a valorilor expresiei **x**.
- Să vedem de exemplu cum putem afla care este cel mai număr de voturi exprimate pentru un candidat într-un judeţ.

SELECT MAX(numar_voturi)

FROM voturi

MAX(NUMAR_VOTURI)
514366

- Se poate observa pe tabelul cu datele din tabela voturi că acest maxim a fost obţinut în Bucureşti de către candidatul având codul **12**.
- Totuşi această informaţie nu este foarte relevantă pentru că şi populaţia din Bucureşti este mult mai mare decât în celelalte judeţe. Ar trebui să putem determina numărul de voturi primite de către un candidat raportat la numărul de alegători (persoane cu drept de vot). SQL ne permite să aplicăm funcţiile de grup nu doar pe câmpuri din baza de date ci şi pe expresii, ca în exemplul următor:

```
SELECT max(100*numar_voturi/numar_alegatori)
FROM voturi v, judete j
WHERE v.judet=j.cod_judet
```

MAX(100*NUMAR_VOTURI/NUMAR_ALEGATORI)
30.230651247867402838950262219070226097 6

- Prin această comandă am obţinut cel mai mare procent de voturi obţinut de către un candidat într-un judeţ. Acest procent a fost obţinut raportat la totalul persoanelor cu drept de vot şi a fost obţinut de către candidatul cu codul 1 în judeţul Iaşi:

```
SELECT 100*numar_voturi/numar_alegatori,
 j.judet, v.candidat
FROM voturi v, judete j
WHERE v.judet=j.cod_judet
```

100*NUMAR_VOTURI/NUMAR_ALEGATORI	JUDET	CANDIDAT
19.827310375090275809739731412325047766 2	Bucuresti	1
.088675985263331108815489957673215281523	Bucuresti	2
.078505672520500607933301032115333631967	Bucuresti	3
30.230651247867402838950262219070226097 6	Iasi	1
…	…	…

- În acest moment nu ştim încă să scriem o comandă pentru a afişa judeţul şi candidatul pentru care s-a obţinut valoarea maximă.
- **MIN(x)** – determină valoarea minimă a valorilor expresiei **x**.
- **SUM(x)** – determină suma valorilor expresiei **x**.

- Cum aflăm oare numărul total de voturi valabil exprimate în judeţul Sibiu? Foarte simplu:

SELECT sum(numar_voturi)

FROM voturi

WHERE judet='SB'

SUM(NUMAR_VOTURI)
212112

- **AVG(x)** – determină media valorilor expresiei **x**. De exemplu, putem afla procentul mediu obţinut un candidat în toate judeţele:

```
SELECT avg(100*numar_voturi/numar_alegatori)
FROM voturi v, judete j
WHERE (candidat=12) and (v.judet=j.cod_judet)
```

- Comanda afişează media procentelor obţinute în fiecare judeţ de către candidatul cu codul **12**:

AVG(100*NUMAR_VOTURI/NUMAR_ALEGATORI)
20.144045084597346892608799213577190663

- Am dori să afişăm un tabel cu procentele obţinute de toţi candidaţii, însă vom vedea cum realizăm acest lucru într-un paragraf următor.
- După cum am precizat la funcţia **COUNT**, funcţiile de grup, deci şi **AVG** ignoră valorile **NULL**. Aşadar dacă vom rula comanda:

```
SELECT avg(numar_voturi)
FROM voturi
WHERE candidat=13
```

- vom obţine valoarea **100**, deşi în baza de date există 3 linii pentru candidatul **13**, şi doar o linie are completat câmpul **numar_voturi** cu valoarea **100**.

- Dacă dorim să obţinem valoarea **33.333**, adică **100/3**, vom scrie:

SELECT AVG(NVL(numar_voturi,0))

FROM voturi

WHERE candidat=13

- adică înlocuim valorile **null** cu valoarea **0**, pentru ca acestea să intre în calculul mediei.

- **STDEV(x)** – funcţie statistică definită ca fiind abaterea pătratică a expresiei date. Cu cât valoarea funcţiei este mai mică cu atât valorile expresiei **x** sunt mai apropiate de medie.
- **VARIANCE(x)** – este o funcţie statistică care calculează dispersia expresiei **x**. Se defineşte ca pătratul abaterii medii pătratice.

- Funcţiile **COUNT**, **MIN**, **MAX** pot fi aplicate şi datelor de tip şir de caractere sau dată calendaristice, celelalte funcţii fiind aplicabile doar valorilor numerice.
- De exemplu comanda următoare va afişa data celei mai vechi angajări, data celei mai recente angajări, numărul de date de angajare, şi numărul de date distincte de angajare din tabela **employees**:

select min(hire_date), max(hire_date), count(distinct hire_date), count(hire_date)

from employees

MIN(HIRE_DATE)	MAX(HIRE_DATE)	COUNT(DISTINCT HIRE_DATE)	COUNT(HIRE_DATE)
17-JUN-87	29-JAN-00	19	20

9.2. Gruparea datelor. Clauza GROUP BY

- Uneori am putea dori să grupăm liniile dintr-o tabelă şi să obţinem anumite informaţii despre grupurile respective.
- De exemplu am dori să calculăm numărul total de voturi obţinut de fiecare candidat în toată ţara. Cu ceea ce am învăţat până acum, am putea rula o comandă de forma celei de mai jos pentru fiecare candidat în parte:

```
SELECT sum(numar_voturi)
FROM voturi
WHERE candidat=1
```

- însă această metodă nu este convenabilă, întrucât am dori să obţinem un tabel cu toate aceste date.

SUM(NUMAR_VOTURI)
608608

- O astfel de grupare a datelor se poate face folosind clauza **GROUP BY**.

SELECT candidat, sum(numar_voturi) AS "TOTAL VOTURI"

FROM voturi

GROUP BY candidat

CANDIDAT	TOTAL VOTURI
1	608608
2	3151
3	3174
4	152465
5	35537
6	19821
7	33844
8	6675
9	8447
10	4009
11	30236
12	632543
13	100

- Se observă că pentru fiecare grup de înregistrări s-a obţinut câte o singură valoare, adică pentru fiecare candidat am obţinut o sumă a tuturor voturilor primite.
- De exemplu candidatul cu codul 1 a obţinut în Bucureşti **347016** voturi, la Iaşi **196508** voturi iar la Sibiu **65084** voturi, în total **608608** voturi.

- Clauza **GROUP BY** poate fi folosită şi fără funcţii de grup, doar pentru a afişa liniile grupate după anumit criteriu:

SELECT candidat,numar_voturi FROM voturi

GROUP BY candidat, numar_voturi

- Procentul mediu obţinut de către fiecare candidat.

SELECT candidat,AVG(100*numar_voturi/numar_alegatori)

FROM voturi v, judete j WHERE v.judet=j.cod_judet

GROUP BY candidat

CANDIDAT	AVG(100*NUMAR_VOTURI/NUMAR_ALEGATORI)
1	22.656229561845598975692085315442433647 6
2	.13424833638246597421520567348011821838
3	.14003282051378316208662701165544554467
4	5.506383429113772239432943207791936674 12
5	1.17019960040031154685700409684022462069
6	.68144301477468349708451524754117789898
7	1.07036088696521333741348008106908880327
8	.22434794824565058705465479428445048096 1
9	.44740619415717432386337983296486539869 3
10	.16661747574531093409946880514800875407 6
11	.97161839160269742583080767121400220691
12	20.14404508459734689260879921357719066 63
13	.02751940117783037041113985359678573394 2

Reguli de folosire a clauzei GROUP BY

- În clauza **GROUP BY** nu se acceptă aliasele coloanelor, comanda următoare va genera o eroare

```
SELECT department_id As Departament,
    job_id, MAX(salary)
   FROM employees GROUP BY Departament,
   job_id
```

- Toate câmpurile care apar în select în afara funcţiilor de grup trebuie să apară în clauza **BROUP BY** ca în exemplele de mai jos:

SELECT department_id, job_id, MAX(salary)
FROM employees GROUP BY department_id, job_id

- sau

SELECT department_id, department_name, max(salary)
FROM employees NATURAL JOIN departments
GROUP BY department_id, department_name

- sau

SELECT upper(last_name), sum(salary)
FROM employees GROUP BY last_name

- Observaţi în ultimul exemplu că deşi în clauza **SELECT** câmpului **last_name** îi este aplicată o funcţie (simplă nu de grup), în clauza **GROUP BY**, **last_name** poate să apară fără funcţia respectivă. (*se pot confunda funcțiile singulare cu cele de grup*)

- Nu se pot folosi funcţii de grup în clauza **WHERE**. De aceea următoarea comandă nu va putea fi rulată ea generând o eroare:

```
SELECT * FROM voturi
WHERE numar_voturi=max(numar_voturi)
```

- Pentru a putea afla ce candidat/candidaţi au obţinut cele mai multe voturi vom folosi o subinterogare (asupra acestui subiect vom reveni în capitolul următor) astfel:

```
SELECT * FROM voturi
WHERE numar_voturi =
    (SELECT max(numar_voturi) from voturi)
```

- în clauza **GRUP BY** pot să apară şi alte coloane care nu apar în **SELECT**

SELECT MAX(salary) FROM employees

GROUP BY departments

- funcţiile de grup pot fi imbricatre ca în exemplul următor, în care am determinat cel mai mare număr total de voturi obţinut de către un candidat.

SELECT max(sum(numar_voturi)) FROM voturi

GROUP BY candidat

MAX(SUM(NUMAR_VOTURI))
632543

9.3. Selectarea grupurilor. Clauza HAVING

- De multe ori nu ne interesează să afişăm toate grupurile de obţinute prin folosirea clauzei **GROUP BY**. Pentru a filtra grupurile folosim clauza **HAVING**. Aşa cum am văzut în exemplele anterioare putem folosi clauza **GROUP BY** fără clauza **HAVING** însă clauza **HAVING** poate fi folosită doar atunci când este prezentă clauza **GROUP BY**.
- Presupunem că dorim să afişăm toţi candidaţii care au obţinut un procent în alegeri mai mare de 5% din numărul total de persoane cu drept de vot. Pentru aceasta procedăm astfel:
  - folosim clauza **GROUP BY** pentru a grupa liniile după candidaţi şi calculăm pentru fiecare candidat procentul obţinut:

```
SELECT candidat,
100*sum(numar_voturi)/sum(numar_alegatori)
FROM voturi v JOIN judete j
ON v.judet=j.cod_judet
GROUP BY candidat
```

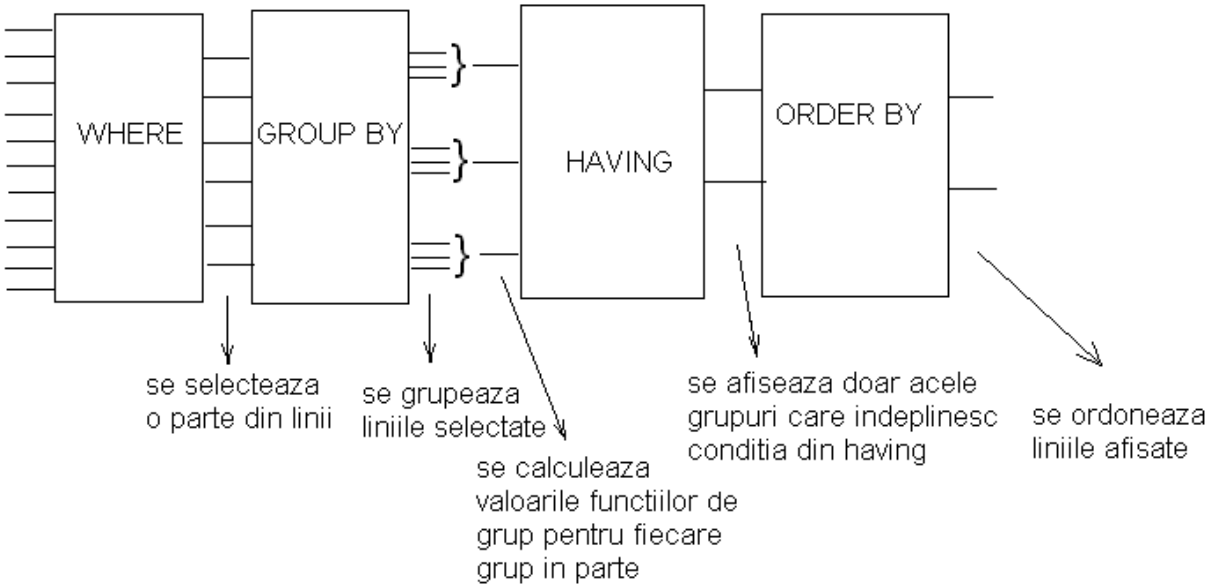
CANDIDAT	100*SUM(NUMAR_VOTURI)/SUM(NUMAR_ALEGATORI)
1	22.022281798276958215024527780964039309 6
2	.114017906347551618341432066351112190219
3	.114850153839139586358522811360974322994
4	5.516896252389545379380019040375220590 82
5	1.285894743850505192319730670237852714 63
6	.717216414381091915945898123499014510416
7	1.224634091534921285670398874511913984 69
8	.241532695928247239742640127862162446 75
9	.305651937454068080015892308621975458831
10	.145064356251137555674643336719012621576
11	1.094079789376252215858946352964845504 11
12	22.888361959631654497157874816227089221 6
13	.003618467354730295726481500042878838154

- Folosim clauza **HAVING** pentru a filtra grupurile care se vor afişa

```
SELECT candidat,
100*sum(numar_voturi)/sum(numar_alegatori)
FROM voturi v JOIN judete j
ON (v.judet=j.cod_judet)
GROUP BY candidat
HAVING 100*sum(numar_voturi)/sum(numar_alegatori)>5
```

CANDIDAT	100*SUM(NUMAR_VOTURI)/SUM(NUMAR_ALEGATORI)
1	22.022281798276958215024527780964039309 6
4	5.516896252389545379380019040375220590 82
12	22.888361959631654497157874816227089221 6

- Bineînţeles că putem folosi clauzele **WHERE**, **GROUP BY** şi **HAVING** împreună. În acest caz, clauza **WHERE** va filtra mai întâi liniile din tabelă, liniile rămase vor fi grupate apoi conform criteriului dat de clauza **GROUP BY** şi în final sunt afişate doar acele grupuri care respectă condiţia dată de clauza **HAVING**.
- Atenţie! Trebuie făcută distincţia clară dintre clauzele **WHERE** şi **HAVING**. Clauza **WHERE** acţionează asupra liniilor în timp ce **HAVING** acţionează la nivel de grup.



Ordinea de executare a clauzelor comenzii **SELECT**

- Să vedem de exemplu cum se evaluează comanda următoare

```
SELECT candidat,
       100*sum(numar_voturi)/sum(numar_alegatori)
FROM voturi v JOIN judete j
ON (v.judet=j.cod_judet)
WHERE numar_voturi>15000
GROUP BY candidat
HAVING
   100*sum(numar_voturi)/sum(numar_alegatori)>5
```

CANDIDAT	100*SUM(NUMAR_VOTURI)/SUM(NUMAR_ALEGATORI)
1	22.022817982769582150245277809640393096
4	5.516896252389545379380019040375220590 82
12	29.351212071318128741296724218526740 5132

- Observaţi însă mai întâi că prin adăugarea clauzei **WHERE**, rezultatele obţinute diferă puţin de cele abținute anterior, aceasta pentru că la calculul procentului obţinut de către candidatul 12 de exemplu nu mai este inclusă următoarea linie din tabelă

JUDET	CANDIDAT	NUMAR_VOTURI
IS	12	12184

- Comanda se evaluează astfel:
  - Mai întâi sunt filtrate liniile din tabelă

SELECT candidat, numar_voturi, numar_alegatori

FROM voturi v JOIN judete j

ON (v.judet=j.cod_judet)

WHERE numar_voturi>15000

  - au fost afişate doar 11 linii din totalul de 39 câte are tabela.

CANDIDAT	NUMAR_VOTURI	NUMAR_ALEGATORI
1	347016	1750192
1	196508	650029
1	65084	363380
4	96744	1750192
5	25656	1750192
4	35784	650029
4	19937	363380
7	25937	1750192
11	22687	1750192
12	514366	1750192
12	105993	363380

- Liniile obţinute la pasul anterior sunt grupate pe candidaţi şi se aplică funcţiile de grup

```
SELECT candidat,
       100*sum(numar_voturi)/sum(numar_alegatori)
FROM voturi v JOIN judete j
ON (v.judet=j.cod_judet)
WHERE numar_voturi>15000
GROUP BY candidat
```

CANDIDAT	100*SUM(NUMAR_VOTURI)/SUM(NUMAR_ALEGATORI)
1	22.022281798276958215024527780964039309 6
4	5.516896252389545379380019040375220590 82
5	1.465896313090220958614826259061862927 04
7	1.481951694442666861692888551656046879 43
11	1.296257782003345918619214349054275188 09
12	29.351212071318128741296724218526740513 2

- În final sunt afişate doar acele linii obţinute la pasul anterior care îndeplinesc condiţia din clauza **HAVING**.

```
SELECT candidat,
       100*sum(numar_voturi)/sum(numar_alegatori)
FROM voturi v JOIN judete j
ON (v.judet=j.cod_judet)
WHERE numar_voturi>15000
GROUP BY candidat
HAVING
   100*sum(numar_voturi)/sum(numar_alegatori)>5
```

CANDIDAT	100*SUM(NUMAR_VOTURI)/SUM(NUMAR_ALEGATORI)
1	22.022817982769582150245277809640393096
4	5.516896252389545379380019040375220590 82
12	29.351212071318128741296724218526740 5132

În această lecție am învățat despre:

- Care sunt funcțiile de grup și cum sunt ele folosite
- Cum se pot grupa datele într-o tabela
- Care sunt regulile de folosire a clauzei GROUP BY
- Cum se pot filtra grupurile folosind clauza HAVING
- Care este diferența dintre clauzele WHERE și HAVING
- În ce ordine se execută clauzele WHERE, GROUP BY și HAVING

SFÂRȘIT