Databázové systémy

6. agregované funkcie, zoskupovanie riadkov, množinové operácie

KKUI
Katedra kybernetiky
a umelej inteligencie

Agregované funkcie

- vracajú hodnoty zo skupín riadkov
- ak nie je definované inak spracujú všetky riadky v tabuľke ako jednu skupinu
- pri argumente agregovanej funkcie môžu byť použité klauzuly DISTINCT alebo ALL
  - DISTINCT – berie do úvahy iba rôzne hodnoty v stĺpci
  - ALL – berie do úvahy všetky hodnoty, implicitné nastavenie

Funkcia AVG()

- vracia priemernú hodnotu všetkých (rôznych) hodnôt daného stĺpca
- typ stĺpca: čísla alebo nenumerický typ,ktorý sa dá konvertovať na čísla
- ignoruje prázdne hodnoty

Príklad:

vypočítajte priemernú mzdu zamestnanca

```
SELECT avg(salary) FROM employees;
```

Funkcia COUNT()

- vracia počet riadkov, ktoré vyhovujú danému kritériu
- COUNT(*) vracia počet všetkých riadkov vrátane prázdnych hodnôt
- typ stĺpca: text, čísla, dátum

Príklad:

- počet všetkých zamestnancov

```
SELECT count(*) FROM employees;
```

- počet oddelení, ktorí majú aspoň jedného zamestnanca

```
SELECT count(distinct department_id) FROM
  employees;
```

Funkcia SUM()

- vracia súčet hodnôt daného výrazu
- typ stĺpca: čísla

Príklad:

- koľko zarábajú zamestnanci, pracujúci pre oddelenie Marketing (id oddelenia je 20)

```
SELECT sum(salary) FROM employees
  WHERE department_id=20;
```

Funkcie MAX() a MIN()

- vracajú maximálnu resp. minimálnu hodnotu výrazu z daného stĺpca
- typ stĺpca: text, čísla, dátum

Príklad:

- zobrazenie maximálneho platu

```
SELECT max(salary) FROM employees;
```

zobrazenie zamestnanca, ktorý najdlhšie pracuje na aktuálnej pozícii

```
SELECT min(hire_date) FROM employees;
```

Funkcia STDDEV()

- štatistická funkcia
- vracia štandardnú odchýlku hodnôt výrazu pre daný stĺpec
- ignoruje prázdne hodnoty
- typ stĺpca: čísla

Príklad:

```
SELECT stddev(salary) FROM employees;
```

Funkcia VARIANCE()

- štatistická funkcia vracajúca rozptyl výrazu resp. hodnôt stĺpca
- ignoruje prázdne hodnoty
- typ stĺpca:čísla

Príklad:

- zobrazenie rozptylu ročných platov zamestnancov, pracujúcich pre oddelenie ľudských zdrojov

```
SELECT variance(salary) FROM employees;
```

Zoskupovanie riadkov

Úloha:

- počet zamestnancov v jednotlivých oddeleniach
- minimálny plat zamestnancov podľa regiónu, v ktorom pracujú/žijú
- zobrazenie počtu zamestnancov pre jednotlivé pozície a oddelenia
- zobrazenie oddelení, v ktorých je priemerná mzda väčšia ako 5000

Zoskupovanie riadkov

- GROUP BY
- umožňuje zoskupovanie riadkov s rovnakou hodnotou atribútu
- nad týmito skupinami môžu byť vykonávané agregované funkcie

Syntax:

```
SELECT <meno stĺpca>, <agregovaná
  funkcia>(<meno stĺpca>) FROM
  <meno tabuľky>
[WHERE <podmienka>]
GROUP BY <meno stĺpca>
[ORDER BY 1,2];
```

Zoskupovanie riadkov

Príklady:

- počet zamestnancov v jednotlivých oddeleniach

```
SELECT department_id, count(*)
FROM employees GROUP BY department_id
ORDER BY 1;
```

- minimálny plat zamestnancov v jednotlivých oddeleniach

```
SELECT department_id, min(salary)
FROM employees GROUP BY department_id
ORDER BY 1;
```

- zobrazenie počtu zamestnancov pre jednotlivé pozície a oddelenia

```
SELECT department_id,job_id,count(*)
FROM  employees GROUP BY department_id,job_id
ORDER BY 1;
```

?ako by sme to urobili , ak by sme chceli zobrazit aj nazvy oddeleni, prac.pozicii?

Klauzula HAVING

- realizuje výber skupín na základe určitej podmienky
- WHERE vs HAVING
  - WHERE: vyberá záznamy z celej tabuľky (pred zoskupením) na základe určitej podmienky(môže obsahovať aj stĺpce mimo GROUP BY klauzuly)
  - HAVING : vyberá skupiny na základe určitého kritéria, teda výber prebieha po zoskupení

Príklad:

- zobrazenie oddelení, v ktorých je priemerná mzda väčšia ako 5000

```
SELECT department_id, avg(salary)
FROM  employees
GROUP BY department_id HAVING avg(salary)>5000;
```

- zobrazenie oddelení, kde pracujú aspoň jeden obchodný zástupca

```
SELECT department_id,job_id,count(*)
FROM  employees WHERE job_id='SA_REP'
GROUP BY department_id,job_id  HAVING count(*)>=1;
```

Klauzula ROLLUP a CUBE

- používajú sa v kombinácii s GROUP BY na zahrnutie medzisúčtov
- ROLLUP
  - na zahrnutie medzisúčtov pre zoskupovacie stĺpce sprava do ľava
- CUBE
  - zahrnutie medzisúčtov pre všetky kombinácie zoskupovacích stĺpcov

https://www.youtube.com/watch?v=CCm4IY-Ntfw

Množinové operácie

- operácie nad matematickými množinami
- kombinujú viacero dopytov
- vyžadujú, aby dopyty mali rovnaký počet argumentov a aby prislúchajúce argumenty boli rovnakého(resp. kombinovateľného) typu
- je možné zjednotiť tabuľky s rôznym počtom stĺpcov a to pridaním stĺpca obsahujúceho konštantu napr. NULL
- ak sú typy rôzne
  - kombinovateľné : automatická konverzia
  - nekombinovateľné: musíme prekonvertovať
- ORDER BY sa dá použiť len na výsledný výber
- ORACLE:
  zjednotenie: UNION, prienik: INTERSECT, rozdiel: MINUS

UNION

- vytvorí zjednotenie výsledkov dvoch alebo viacerých dopytov
- eliminuje duplikáty
- výsledok je usporiadaný

Syntax:

```
SELECT <zoznam stĺpcov> FROM <meno tabuľky1>
UNION
SELECT <zoznam stĺpcov> FROM <meno tabuľky2>;
```

Príklad:

- zobrazeniezamestnancov a dobrovoľníkov, ktorí pracujú pre firmu
- volunteers- tabuľka obsahujúca údaje o ľudoch, ktorí s firmou spolupracujú ako dobrovoľnéci na dobročinných projektoch

```
SELECT first_name,last_name,email FROM employees UNION
   SELECT first_name,last_name,email FROM volunteers;
```

UNION ALL

- vytvorí zjednotenie výsledkov dvoch alebo viacerých dopytov
- vracia všetky riadky t.j. neeliminuje duplikáty
- výsledok nie je utriedený
- rýchlejší ako UNION
- Syntax:

```
SELECT <zoznam stĺpcov> FROM <meno tabuľky1>
UNION
SELECT <zoznam stĺpcov> FROM <meno tabuľky2>;
```

```
SELECT first_name,last_name,email FROM employees
UNION ALL
SELECT first_name,last_name,email FROM volunteers;
```

-rozdiel s predchadzajúcim príkladom?

UNION ALL

- kedy použiť UNION ALL?
  - nezáleží, či výsledok obsahuje duplikáty
  - duplikáty je potrebné zobraziť
  - duplikáty sa vo výsledkoch dopytov nenachádzajú (napr. atribúty s obmedzeniami PRIMARY KEY)

INTERSECT

- vytvorí prienik výsledkov dvoch dopytov t.j. zobrazí riadky, ktoré sa nachádzajú vo výsledkoch oboch dopytov
- môžeme nahradiť pomocou INNER JOIN

Syntax:

```
SELECT <zoznam stĺpcov> FROM <meno tabuľky1>
INTERSECT
SELECT <zoznam stĺpcov> FROM <meno tabuľky2>;
```

Príklad:

- zobrazeniezamestnancov, ktorí pracujú ako aj dobrovoľníci
- volunteers- tabuľka obsahujúca údaje o ľudoch, ktorí s firmou spolupracujú ako dobrovoľnéci na dobročinných projektoch

```
SELECT first_name,last_name,email FROM employees
INTERSECT
SELECT first_name,last_name,email FROM volunteers;
```

MINUS

- vytvorí rozdiel výsledkov dvoch dopytov t.j. zobrazí riadky, ktoré sa nachádzajú vo výsledku prvého dopytu, ale nie vo výsledku druhého dopytu
- nahraditeľné pomocou OUTER JOIN

! záleží na poradí: A-B≠B-A

Syntax:

```
SELECT <zoznam stĺpcov> FROM <meno tabuľky1>
MINUS
SELECT <zoznam stĺpcov> FROM <meno tabuľky2>;
```

Príklad:

- Zobrazenie zamestnancov, ktorí nepracujú ako dobrovoľníci
- volunteers- tabuľka obsahujúca údaje o ľudoch, ktorí s firmou spolupracujú ako dobrovoľnéci na dobročinných projektoch

```
SELECT first_name,last_name,email FROM employees
MINUS
SELECT first_name,last_name,email FROM volunteers;
```