SELECT: GROUP BY, HAVING

Souhrnné informace pro celou tabulku

- Nyní už víte k čemu slouží agregační funce MIN (minimum), MAX (maximum), AVG (průměr), COUNT (počet), SUM (součet).
- Často ale nechceme získat jeden výsledek pro všechny řádky v tabulce, ale chceme výsledky pro určité skupiny řádků.
- Ke seskupení do skupin řádků slouží klauzule GROUP BY.

Příkaz SELECT – klauzule GROUP BY

- Umožňuje získat souhrnné informace pro určité skupiny řádků v tabulce – například pro jednotlivá oddělení se zaměstnanci.
- Klauzule GROUP BY („seskupit podle“) udává jména sloupců, podle kterých mají být vybrané řádky seskupeny – a pro tyto skupiny mohou být následně počítány min./max./průměry aj.
- Pokud chceme například získat průměrné platy zaměstnanců zvlášť pro jednotlivá oddělení, budeme řádky tabulky zamestnanec seskupovat podle identifikátoru oddělení (oddeleni_id).
- Seskupení se provede tak, že jsou nejdříve vyhledány řádky, které mají stejný identifikátor oddělení. Souhrnné informace (průměr, suma apd.) pak budou vypočítány pro takto vzniklé skupiny řádků. Ve výsledku dotazu se objeví pro každou skupinu řádků jeden výsledný řádek se souhrnnou informací.

Příkaz SELECT – klauzule GROUP BY

- Př.: `SELECT zamestnani_id, AVG(plat) AS prumerny_plat FROM zamestnanec GROUP BY zamestnani_id;`

```
ZAMESTNANI_ID PRUMERNY_PLAT
------------- -------------
            1         43000
            2        120930
```

- Seskupení lze provádět vícekrát po sobě – např. zaměstnance lze seskupovat podle oddělení a následně podle pohlaví – vzniknou skupiny zaměstnanců o stejném pohlaví ve stejném oddělení (GROUP BY zamestnani_id, pohlavi).
- Pokud použijeme GROUP BY ve složitějších dotazech, může být obtížné určit, podle jakých sloupců bychom měli seskupovat.
- Dobrá rada zní: Každopádně podle všech sloupců vybíraných SELECTem, které nejsou zrovna použity v agregační funkci, a podle dalších uvažovaných sloupců, které nemusí být SELECTem vybrány! Seskupovat by se mělo podle identifikátorů „skupin“ (např. zamestnani_id), nikoliv např. podle názvů (to nemusí být spolehlivé, pokud má více skupin stejný název).

Příkaz SELECT – klauzule HAVING

- Umožňuje omezit počet řádků vrácených SELECTem tím, že se v ní mohou uvést podmínky kladené na výsledky agregačních funkcí.
- Např. můžeme chtít vybrat pouze takové řádky odpovídající oddělením ve firmě, ve kterých je průměrný plat zaměstnanců větší něž 30000 Kč.
- Omezení uvedená v HAVING se aplikují až ve fázi zpracování dotazu, kdy už jsou řádky seskupeny podle hodnot sloupců uvedených v GROUP BY (tj. např. na výsledek dotazu v minulém slidu) – z výsledných řádků se souhrnnými informacemi můžeme chtít nechat zobrazit (vyfiltrovat) jen některé.
- Př.:

```
SELECT AVG(plat) AS prumerny_plat FROM zamestnanec
GROUP BY zamestnani_id
HAVING AVG(plat) > 23000;
```

```
PRUMERNY_PLAT
-------------
        24000
```

Nelze použít alias prumerny_plat!

ROLLUP, CUBE I.

- ROLLUP a CUBE modifikátory umožňují vytvářet agregace agregací (tzv. superagregace). Tyto superagregace jsou součástí výsledku a obsahují sumární výsledky výsledků agregačních funkcí.
- **Příklad použití ROLLUP:**
- Získejte počty aktivních a neaktivních zaměstnanců v jednotlivých odděleních:

```
SELECT ZAMESTNANI.NAZEV_POZICE, ZAMESTNANEC.AKTIVNI,
COUNT(ZAMESTNANEC.ZAMESTNANEC_ID) POCET

FROM ZAMESTNANEC INNER JOIN ZAMESTNANI

ON ZAMESTNANEC.ZAMESTNANI_ID = ZAMESTNANI.ZAMESTNANI_ID

GROUP BY ZAMESTNANI.NAZEV_POZICE, ZAMESTNANEC.AKTIVNI

ORDER BY ZAMESTNANI.NAZEV_POZICE;
```

ROLLUP, CUBE II.

- Vypište navíc pro každou pracovní pozici počet zaměstnanců, kteří ji zastávají nezávisle na tom, jestli jsou aktivní či neaktivní:

```
SELECT ZAMESTNANI.NAZEV_POZICE, ZAMESTNANEC.AKTIVNI,
COUNT(ZAMESTNANEC.ZAMESTNANEC_ID) POCET

FROM ZAMESTNANEC

INNER JOIN ZAMESTNANI

ON ZAMESTNANEC.ZAMESTNANI_ID = ZAMESTNANI.ZAMESTNANI_ID

GROUP BY ZAMESTNANI.NAZEV_POZICE,
ROLLUP(ZAMESTNANEC.AKTIVNI)

ORDER BY ZAMESTNANI.NAZEV_POZICE;
```

ROLLUP, CUBE III.

- Vypište navíc na konci celkový počet zaměstnanců:

```
SELECT ZAMESTNANI.NAZEV_POZICE, ZAMESTNANEC.AKTIVNI,
COUNT(ZAMESTNANEC.ZAMESTNANEC_ID) POCET

FROM ZAMESTNANEC

INNER JOIN ZAMESTNANI

ON ZAMESTNANEC.ZAMESTNANI_ID = ZAMESTNANI.ZAMESTNANI_ID

GROUP BY ROLLUP(ZAMESTNANI.NAZEV_POZICE,
ZAMESTNANEC.AKTIVNI)

ORDER BY ZAMESTNANI.NAZEV_POZICE;
```

- Tyto výpočty pouze nepatrně zvyšují časové nároky dotazů.
- Pořadí sloupců v ROLLUP modifikátoru je významné, protože určuje jakým způsobem se budou počítat mezivýsledky.

ROLLUP, CUBE III.

- CUBE modifikátor v GROUP BY klauzuli vytváří mezivýsledky pro všechny kombinace sgrupovaných sloupců.
- Vypište navíc na konci počet aktivních a neaktivních zaměstnanců:

```
SELECT ZAMESTNANI.NAZEV_POZICE, ZAMESTNANEC.AKTIVNI,
COUNT(ZAMESTNANEC.ZAMESTNANEC_ID) POCET FROM ZAMESTNANEC

INNER JOIN ZAMESTNANI

ON ZAMESTNANEC.ZAMESTNANI_ID = ZAMESTNANI.ZAMESTNANI_ID

GROUP BY CUBE(ZAMESTNANI.NAZEV_POZICE, ZAMESTNANEC.AKTIVNI)

ORDER BY ZAMESTNANI.NAZEV_POZICE;
```

GROUPING

- V předcházejících příkladech jsme viděli, že při použití ROLLUP a CUBE je problematické určování, který řádek je agregací a který superagregací. K rozlišení můžete použít funkci GROUPING:

```
SELECT ZAMESTNANI.NAZEV_POZICE, ZAMESTNANEC.AKTIVNI,
COUNT(ZAMESTNANEC.ZAMESTNANEC_ID) POCET,
GROUPING(ZAMESTNANI.NAZEV_POZICE), GROUPING(aktivni)

FROM ZAMESTNANEC

INNER JOIN ZAMESTNANI

ON ZAMESTNANEC.ZAMESTNANI_ID = ZAMESTNANI.ZAMESTNANI_ID

GROUP BY CUBE(ZAMESTNANI.NAZEV_POZICE, ZAMESTNANEC.AKTIVNI)

ORDER BY ZAMESTNANI.NAZEV_POZICE;
```

- **Tip:** K nahrazení jedniček a nul za výstižnější text můžete použít funkci DECODE nebo klauzuli CASE. **Tip 2:** Funkci GROUPING můžete použít v HAVING klauzuli pro vyfiltrování řádků.

GROUPING_ID

- GROUPING_ID funkce nabízí alternativní a více kompaktní způsob identifikování agregovaných řádek oproti GROUPING funkci. Dovnitř vstupuje seznam sloupců a vrací číslo, odpovídající úrovni agregace. Na pořadí sloupců uvnitř funkce záleží.

```
SELECT ZAMESTNANI.NAZEV_POZICE, ZAMESTNANEC.AKTIVNI,
  COUNT(ZAMESTNANEC.ZAMESTNANEC_ID) POCET,
  GROUPING(ZAMESTNANI.NAZEV_POZICE), GROUPING(AKTIVNI),
  GROUPING_ID(NAZEV_POZICE, AKTIVNI) UROVEN_AGREGACE
FROM ZAMESTNANEC
INNER JOIN ZAMESTNANI
ON ZAMESTNANEC.ZAMESTNANI_ID = ZAMESTNANI.ZAMESTNANI_ID
GROUP BY CUBE(ZAMESTNANI.NAZEV_POZICE, ZAMESTNANEC.AKTIVNI)
ORDER BY ZAMESTNANI.NAZEV_POZICE;
```

GROUPING SETS

- Spočítání všech agregací v CUBE, zejména když se používá více dimenzí, může být časově náročné. Pokud nepotřebujete všechny agregace, pak je možné pomocí klauzule GROUPING SETS jejich počet výrazně snížit, protože nad nimi máte větší kontrolu:

```
SELECT ZAMESTNANI.NAZEV_POZICE, ZAMESTNANEC.AKTIVNI,
  COUNT(ZAMESTNANEC.ZAMESTNANEC_ID) POCET,
  GROUPING(ZAMESTNANI.NAZEV_POZICE), GROUPING(AKTIVNI),
  GROUPING_ID(NAZEV_POZICE, AKTIVNI) UROVEN_AGREGACE
FROM ZAMESTNANEC
INNER JOIN ZAMESTNANI
ON ZAMESTNANEC.ZAMESTNANI_ID = ZAMESTNANI.ZAMESTNANI_ID
GROUP BY GROUPING SETS((ZAMESTNANI.NAZEV_POZICE,
ZAMESTNANEC.AKTIVNI), (ZAMESTNANEC.AKTIVNI,
ZAMESTNANI.NAZEV_POZICE))
ORDER BY ZAMESTNANI.NAZEV_POZICE;
```

GROUP_ID I.

- Můžete se dostat do situace, kdy napíšete dotaz, který Vám bude vracet duplicitní agregace (jako na předcházejícím snímku). Funkce GROUP_ID přiřadí prvnímu setu nulu a ostatním setům přiřadí vyšší číslo:

```
SELECT GROUP_ID(), ZAMESTNANI.NAZEV_POZICE,ZAMESTNANEC.AKTIVNI,
  COUNT(ZAMESTNANEC.ZAMESTNANEC_ID) POCET,
  GROUPING(ZAMESTNANI.NAZEV_POZICE), GROUPING(AKTIVNI),
  GROUPING_ID(NAZEV_POZICE, AKTIVNI) UROVEN_AGREGACE
FROM ZAMESTNANEC
INNER JOIN ZAMESTNANI
ON ZAMESTNANEC.ZAMESTNANI_ID = ZAMESTNANI.ZAMESTNANI_ID
GROUP BY GROUPING SETS((ZAMESTNANI.NAZEV_POZICE,
ZAMESTNANEC.AKTIVNI), (ZAMESTNANEC.AKTIVNI,
ZAMESTNANI.NAZEV_POZICE))
ORDER BY ZAMESTNANI.NAZEV_POZICE;
```

GROUP_ID II.

- GROUP_ID se používá k odstranění duplicitních agregací:

```
SELECT GROUP_ID(), ZAMESTNANI.NAZEV_POZICE,
  ZAMESTNANEC.AKTIVNI,
  COUNT(ZAMESTNANEC.ZAMESTNANEC_ID) POCET,
  GROUPING(ZAMESTNANI.NAZEV_POZICE),
  GROUPING(AKTIVNI),
  GROUPING_ID(NAZEV_POZICE, AKTIVNI) UROVEN_AGREGACE
FROM ZAMESTNANEC
INNER JOIN ZAMESTNANI
ON ZAMESTNANEC.ZAMESTNANI_ID = ZAMESTNANI.ZAMESTNANI_ID
GROUP BY GROUPING SETS((ZAMESTNANI.NAZEV_POZICE,
ZAMESTNANEC.AKTIVNI), (ZAMESTNANEC.AKTIVNI, ZAMESTNANI.NAZEV_POZICE))
HAVING GROUP_ID() = 0
ORDER BY ZAMESTNANI.NAZEV_POZICE;
```

LISTAGG I.

- Někdy můžete potřebovat převést sloupec záznamů na jeden řádek, kde budou hodnoty oddělené například středníkem. Od Oracle 11g k tomu slouží funkce LISTAGG.
  - Poznámka: Dříve se používaly funkce jako STRAGG nebo WM_CONCAT:
    - http://oraenablement.wordpress.com/2011/11/15/listagg-vs-stragg-vs-wm_concat/
- Příklad:

```
SELECT LISTAGG(last_name, '; ')
WITHIN GROUP (ORDER BY hire_date)
FROM HR.EMPLOYEES;
```

LISTAGG II.

- Tento SELECT vrátí seznam všech zaměstnanců a jejich nadřízených:

```
SELECT
b.jmeno || ' ' || b.prijmeni nadrizeny,
a.jmeno || ' ' || a.prijmeni zamestnanci
FROM zamestnanec a
LEFT JOIN zamestnanec b
ON a.nadrizeny = b.zamestnanec_id
```

LISTAGG III.

- Tento SELECT vrátí seznam všech nadřízených a jejich podřízených, kteří budou vždy na jednom řádku odděleni čárkou:

```
SELECT
b.jmeno || ' ' || b.prijmeni nadrizeny,
listagg(a.jmeno || ' ' || a.prijmeni, ',')
WITHIN GROUP (ORDER BY a.jmeno) zamestnanci
FROM zamestnanec a
LEFT JOIN zamestnanec b
ON a.nadrizeny = b.zamestnanec_id
group by b.jmeno, b.prijmeni;
```