Vysoká škola báňská – Technická univerzita Ostrava
Fakulta strojní, Katedra automatizační techniky a řízení

5

Informační systémy

2008/2009

Radim Farana

Obsah

- Jazyk SQL,
  - Spojení tabulek,
  - agregační dotazy,
  - jednoduché a složené dotazy,
  - dotazy DML.
- Doporučená literatura:
  Gruber, M. *Mistrovství v SQL*. Svazek 1. Praha : SoftPress s.r.o., 2004. ISBN 80-86497-62-3

Dotazy přes více tabulek

- Z principu optimální databáze vyplývá, že data jsou rozdělena do více tabulek.
- Je třeba položit SQL dotaz na více tabulek současně.
- Chceme vypsat informace jako jméno, příjmení a ne ID
- Musíme definovat spojení mezi tabulkami
- Spojení pomocí WHERE
- Spojení pomocí JOIN
- U názvů sloupců musíme uvádět ze které tabulky pocházejí.
- tZakaznik.Jmeno, tObjednavka.Cena

Dotaz pomocí WHERE

- Výběr jména a příjmení zákazníka spolu s cenou jeho objednávky

```
SELECT tZakaznik.Jmeno, tZakaznik.Prijmeni, tObjednavka.Cena
    FROM tZakaznik as z, tObjednavky as o
    WHERE o.IDZak = z.ID;
```

- Můžeme použít i rozšiřující kritéria spojená pomocí logických funkcí

```
SELECT tZakaznik.Jmeno, tZakaznik.Prijmeni, tObjednavka.Cena
    FROM tZakaznik, tObjednavky
    WHERE (tObjednavky.IDZak = tZakaznik.ID) AND
    (tZakaznik.Jmeno = 'Petr');
```

Spojení tabulek pomocí JOIN

- Slouží k položení dotazu přes více tabulek
- Syntaxe

```
SELECT [seznam_poli] FROM
    leva_tab <LEFT | RIGHT> [OUTER] JOIN prava_tab
    ON podminka;
```

Spojení tabulek pomocí JOIN

- INNER JOIN
  - Obsahuje jen hodnoty, které vyhovují kriteriu
  - Obdobný výsledek jako použití WHERE
  - Syntaxe
  - **SELECT seznam_poli**
    FROM Tab1 INNER JOIN Tab2
    ON podminka;

```
SELECT tZakaznik.Jmeno, tZakaznik.Prijmeni, tObjednavka.Cena
    FROM tZakaznik INNER JOIN tObjednavky
    ON tObjednavky.IDZak = tZakaznik.ID;
```

Spojení mimo rovnost

1
2
3

1
2
3

- SELECT Osoby.* FROM Osoby INNER JOIN Osoby As TMP On Osoby.ID < TMP.ID;

1, 2
1, 3
2, 3

- SELECT Osoby.* FROM Osoby INNER JOIN Osoby As TMP On Osoby.ID <> TMP.ID;

1, 2
1, 3
2, 1
2, 3
3, 1
3, 2

Spojení tabulek pomocí JOIN

- LEFT JOIN
  - Z levé tabulky se vyberou všechny záznamy
  - Z pravé jen ty, které vyhovují podmínce
  - Zbytek bude mít hodnotu NULL
  - Objednávky jsou přiřazeny k zákazníkům
  - RIGHT JOIN – Zákazníci k objednávkám

```
SELECT tZakaznik.Jmeno, tZakaznik.Prijmeni, tObjednavka.Cena
   FROM tZakaznik LEFT JOIN tObjednavky
   ON tObjednavky.IDZak = tZakaznik.ID;
```

Záznamy bez podřízených záznamů

- Využití výsledku spojení:
  - **SELECT Hlavni.***
    FROM Hlavni LEFT JOIN Vedlejsi
    ON Hlavni.PrimarniKlic = Vedlejsi.CiziKlic
    WHERE Vedlejsi.PrimarniKlic IS NULL;

Spojení tabulek pomocí JOIN

- FULL JOIN
  - Vypíše všechny záznamy z obou tabulek,
  - U záznamů, které nevyhovují podmínce je opět NULL,
  - Uděláme v podstatě LEFT i RIGHT JOIN současně.

Spojení tabulek pomocí JOIN

- CROSS JOIN
  - Vypíše kartézský součin množin záznamů.
  - Kombinace každý s každým.
  - Vhodné pro generování testovacích množin, rozpisu každý s každým.
  - Nepíše se podmínka.

```
SELECT tZakaznik.Jmeno, tZakaznik.Prijmeni,
   tObjednavka.Cena FROM tZakaznik CROSS JOIN
   tObjednavky;
```

Spojení tabulek pomocí JOIN

- OUTER JOIN
  - Vypíše i ty záznamy, které nevyhovují podmínce
  - Dá se kombinovat s LEFT a RIGHT

```
SELECT tZakaznik.Jmeno, tZakaznik.Prijmeni,
   tObjednavka.Cena FROM tZakaznik LEFT OUTER JOIN
   tObjednavky ON tObjednavky.IDZak = tZakaznik.ID;
```

Agregační funkce

- Vestavěné funkce používané pro souhrny
- Vrací jedinou hodnotu jako výsledek

AVG	-	Aritmetický průměr
COUNT	-	Počet hodnot ve výrazu
MAX	-	Nejvyšší hodnota výrazu
MIN	-	Nejnižší hodnota výrazu
SUM	-	Součet všech hodnot výrazu

- SELECT SUM(Cena) FROM Objednavky;

Vrátí celkovou cenu všech objednávek.

Seskupení hodnot

- Používá se spolu s agregačními funkcemi
- Pro množinu záznamů GROUP BY

SELECT IDZak, SUM(Cena) FROM Objednavky GROUP BY IDZak;

- V klauzili GROUP BY musí být uvedeny všechny neagregované sloupce

SELECT IDZak, Datum, SUM(Cena) FROM Objednavky GROUP BY IDZak, Datum;

- Můžeme využít i WHERE

SELECT IDZak, Datum, SUM(Cena) FROM Objednavky WHERE IDZak=3 GROUP BY IDZak, Datum;

Seskupení hodnot

- Pokud chceme vložit omezující podmínku přidáme HAVING <podmínka>
- Použití HAVING jen s GROUP BY a agregační funkcí
- Jiné použití nemá smysl

SELECT IDZak, SUM(Cena) FROM Objednavky GROUP BY IDZak HAVING (SUM(Cena) > 300);

Složené dotazy

- Výsledek jednoho dotazu je argumentem jiného dotazu.
- Nejčastěji v rámci klauzulí
  - SELECT,
  - WHERE.

Skalární hodnoty

- V rámci klauzule SELECT
- SELECT tblSteps.*, stpPrice/(SELECT Sum(stpPrice) FROM tblSteps) AS stpPerc FROM tblSteps;
- V rámci klauzule WHERE
- SELECT tblSteps.* FROM tblSteps WHERE tblSteps.stpPrice>(SELECT Avg(stpPrice) as stpPriceAvg FROM tblSteps);

Propojení vloženého dotazu

- SELECT tblSteps.* FROM tblSteps WHERE tblSteps.stpPrice)> (SELECT Avg(stpPrice) as stpPriceAvg FROM tblSteps as Pom WHERE tblSteps.stpOperation=Pom.stpOperation);

Položka nadřízené tabulky

Položka podřízené tabulky

Vektor hodnot

- Testování existence seznamu (EXISTS, NOT EXISTS).
- Porovnání hodnoty se seznamem vrácených hodnot:
  - IN - nachází se v seznamu,
  - ANY - podmínka platí alespoň pro jednu hodnotu v seznamu,
  - ALL - podmínka platí pro všechny hodnoty v seznamu.

Vektor hodnot

- WHERE Adresy.DatumNarozeni IN
  (SELECT Adresy.DatumNarozeni FROM Adresy WHERE Adresy.Prijmeni="Novak")
- vrátí všechny osoby, které se narodily stejný den jako některý z Nováků
- WHERE Adresy.DatumNarozeni < ANY
  (SELECT Adresy.DatumNarozeni FROM Adresy WHERE Adresy.Prijmeni="Novak")
- vrátí všechny osoby, které se narodily dříve než některý z Nováků
- WHERE Adresy.DatumNarozeni < ALL
  (SELECT Adresy.DatumNarozeni FROM Adresy WHERE Adresy.Prijmeni="Novak")
- vrátí jen osoby, které se narodily dříve než všichni Nováci

Hledání duplicitních záznamů

- SELECT * FROM tblSteps
  WHERE stpOperation)
  In (SELECT stpOperation FROM tblSteps As Tmp GROUP BY stpOperation HAVING Count(*)>1)
  ORDER BY tblSteps.stpOperation;

O duplicitě rozhoduje jedna položka

Hledání duplicitních záznamů

- SELECT Adresy.* FROM Adresy
  WHERE Adresy.Prijmeni In (SELECT [Prijmeni] FROM [Adresy] As Tmp
  GROUP BY [Prijmeni],[Jmeno]
  HAVING Count(*)>1 And [Jmeno] = [Adresy].[Jmeno])
  ORDER BY Adresy.Prijmeni, Adresy.Jmeno;

Propojení ostatních kontrolovaných položek

Vkládání nových záznamů

- Určení jednotlivých hodnot
  INSERT INTO tabulka (seznam položek)
  VALUES (seznam hodnot);

- INSERT INTO Pracovnici
  (pr_OSC, pr_Jmeno)
  VALUES (122, 'Jan Kvákal');

Vkládání nových záznamů

- Převzetí záznamů z jiné tabulky
  INSERT INTO tabulka (seznam položek)
  SELECT seznam položek FROM zdroj;

- INSERT INTO PracovniciByvali
  (pr_OSC, pr_Jmeno)
  SELECT pr_OSC, pr_Jmeno FROM
  Pracovnici WHERE pr_OSC<100;

Úprava záznamů

- UPDATE Tabulka
  SET položka=hodnota
  WHERE primární klíč=hodnota;

- UPDATE Pracovnici
  SET pr_Jmeno='Jan Kvákal'
  WHERE pr_OSC=122;

Odstranění záznamů

- DELETE FROM tabulka WHERE
  podmínka;

- DELETE FROM Pracovnici
  WHERE pr_OSC=122;

Spojení záznamů z více zdrojů

- Zdroj1 UNION Zdroj2;

- SELECT pr_OSC, pr_Jmeno
  FROM Pracovnici
  UNION
  SELECT p_pracovnik, p_typ
  FROM prace;

Další operace podle standardu SQL

- Rozdíl záznamů ze dvou zdrojů
  - Zdroj1 EXCEPT Zdroj2; (také MINUS)
  - vrátí záznamy Zdroje1, které nejsou ve Zdroji2.
- Shoda záznamů dvou zdrojů
  - Zdroj1 INTERSECT Zdroj2;
  - Vrátí záznamy nacházející se v obou zdrojích.
- Rozšíření ALL, závisí na implementaci.

MS SQL Server nepodporuje

Spojení záznamů z více zdrojů

- Požadavky na shodu datových typů:

Data type of *ith* column	Data type of *ith* column of results table
Not data type-compatible (data conversion not handled implicitly by Microsoft® SQL Server™).	Error returned by SQL Server.
Both fixed-length char with lengths L1 and L2.	Fixed-length char with length equal to the greater of L1 and L2.
Both fixed-length binary with lengths L1 and L2.	Fixed-length binary with length equal to the greater of L1 and L2.
Either or both variable-length char.	Variable-length char with length equal to the maximum of the lengths specified for the *ith* columns.
Either or both variable-length binary.	Variable-length binary with length equal to the maximum of the lengths specified for the *ith* columns.
Both numeric data types (for example, smallint, int, float, money).	Data type equal to the maximum precision of the two columns. For example, if the *ith* column of table A is of type int and the *ith* column of table B is of type float, then the data type of the *ith* column of the results table is float because float is more precise than int.
Both columns' descriptions specify NOT NULL.	Specifies NOT NULL.