

Databáze

aneb Co jste se mohli naučit

SQL – historie

- Od 70. let 20. století, původně SEQUEL (Structured English Query Language)
- 1986 SQL-86 – standard podle jazyka z tohoto roku
- 1992 SQL-92 – opravená verze SQL-86, opravena integrita dat
- 1999 SQL-99 – podlesní verze jazyka podporující práci s objekty

SQL – úvod

- Každý příkaz jazyka SQL se skládá z klíčových slov a identifikátorů
- Např. `SELECT id, jmeno FROM interpret`

SELECT

- Používá se pro výpis dat z tabulky
- Syntaxe: SELECT [TOP(pocet) [PERCENT]]
[DISTINCT] [* | jmeno_sloupce[,dalsi_sloupce]
[AS alias]] FROM [jmeno_databaze!]tabulka
[spojeni_s_dalsimi_tabulkami] [WHERE
podminky] [GROUP BY
sloupec_pro_seskupeni] [ORDER BY sloupec1[,
sloupec2] [ASC|DESC]]

SELECT – vysvětlení

- SELECT vypíše prvních *pocet (procent)* záznamů z tabulek *databáze!tabulka*, které vyhovují podmínce *podminka*, seskupených podle *sloupec_pro_seskupeni*, seřazených podle *sloupec1* vzestupně | sestupně
- DISTINCT oznamuje databázovému enginu, že chceme pouze jedinečné názvy (nebude vypsáno vícekrát např. album Best of)
- AS přejmenuje sloupec ve výpisu (i při odkazování) na *alias*.

Agregační funkce

- COUNT(sloupec), MAX(sloupec), MIN(sloupec), AVG(sloupec), SUM(sloupec)
- Vypíše počet záznamů, největší prvek, nejmenší sloupec, průměr, součet prvků ze sloupce *sloupec*.

INSERT

- Vkládání do tabulky
- Syntaxe: `INSERT INTO [databaze!]tabulka (sloupec1[, sloupec2, ...]) VALUES (hodnota1[, hodnota2, ...])`
- Vloží do tabulky *databaze!tabulka* hodnoty *sloupec1=hodnota1, sloupec2=hodnota2, ...*

UPDATE

- Aktualizuje data v tabulce
- Syntaxe: UPDATE [databaze!]tabulka SET sloupec1=hodnota1[, sloupec2=hodnota2] [WHERE podminka]
- Změní v tabulce *tabulka* hodnotu v *sloupec1* na *hodnota1*, pokud vyhovuje podmínce *podminka*

DELETE

- Vymaže záznamy z tabulky
- Syntaxe: DELETE FROM [databaze!]tabulka
[WHERE podmínka]
- Vymaže z tabulky *tabulka* záznamy vyhovující
podmínce *podmínka*

Propojení tabulek – I.

- Tabulky propojujeme příkazem JOIN. Jsou tři typy spojení: INNER, LEFT a RIGHT
- INNER JOIN – výsledkem spojení je pouze propojení odpovídajících si záznamů (žádný není navíc)
- LEFT JOIN – výsledkem je celá levá tabulka a odpovídající záznamy z pravé tabulky
- RIGHT JOIN – jako LEFT JOIN, ale celá tabulka je pravá

Propojení tabulek – II.

- Syntaxe: [databaze!]tabulka1
[INNER | LEFT | RIGHT] JOIN
[databaze!]tabulka2 ON podminka_spojeni
- Například: skladba INNER JOIN interpret ON
interpret.id=skladba.interpret
- Je možno propojit více než 2 tabulky:
následuje další příkaz pro propojení

MS Visual Studio 2005

- K prohlížení tabulek slouží komponenta DataGridView
- Jako zdroj dat pro zobrazení se bere vlastnost DataSource
- Vlastnost Columns slouží pro výběr požadovaných sloupců pro zobrazení

Dynamické zobrazení dat (SQL příkazu)

```
dataGridView1.AutoGenerateColumns = true; // nutné pro správnou interpretaci výsledných dat
```

```
string connectionString = „k nalezení ve vlastnostech spojení“;
```

```
// vytvoříme si SQL spojení
```

```
SqlConnection conn = new SqlConnection(connectionString);
```

```
// vytvoříme si SQL příkaz
```

```
SqlCommand command = new SqlCommand(txtSql.Text, conn);
```

```
SqlDataAdapter adapter = new SqlDataAdapter(); // vytvoříme si adaptér – bude obsahovat data
```

```
adapter.SelectCommand = command; // nastavíme mu SQL příkaz
```

```
DataTable dt = new DataTable(); // reprezentuje paměťový obraz tabulky
```

```
try
```

```
{
```

```
    adapter.Fill(dt); // Naplníme datovou tabulku z adaptéru
```

```
}
```

```
catch (SqlException se) // odchytíme případné výjimky
```

```
{
```

```
    MessageBox.Show("SqlException: " + se.Message, "Error", MessageBoxButtons.OK,  
        MessageBoxIcon.Error);
```

```
}
```

```
dataGridView1.DataSource = dt; // nastavíme zdroj dat pro DataGridView
```

```
dataGridView1.Refresh(); // obnovíme ho, aby se aktualizovala data
```