Zápisování dat do databáze

Informační a znalostní systémy

Vkládání záznamů

- Pro vkládání záznamů do tabulky - příkaz INSERT

```
INSERT INTO tabulka
  VALUES ( výčet hodnot záznamu )
```

Vkládání záznamů

- Pro vkládání záznamů do tabulky - příkaz INSERT

```
INSERT INTO tabulka
  VALUES ( výčet hodnot záznamu )
```

- Příklad: Vložte do tabulky mereni záznam odpovídající aktualnímu času, senzoru s id = 1, veličině s id = 2, s náhodně vygenerovanou hodnotou

```
INSERT INTO mereni VALUES (now(),1,2,20);
```

  - pořadí hodnot odpovídá pořadí atributů v tabulce

Vkládání záznamů

- Pro vkládání záznamů do tabulky - příkaz INSERT

```
INSERT INTO tabulka
  VALUES ( výčet hodnot záznamu )
```

- Příklad: Vložte do tabulky mereni záznam odpovídající aktualnímu času, senzoru s id = 1, veličině s id = 2, s náhodně vygenerovanou hodnotou

```
INSERT INTO mereni VALUES (now(),1,2,20);
```

  - pořadí hodnot odpovídá pořadí atributů v tabulce
- tento zápis:
  - výhoda: krátký rychlý zápis
  - nevýhoda: přestane fungovat při změně struktury tabulky

Vkládání záznamů - výčet atributů

- tuto nevýhodu odstraňuje použití syntaxu

```
INSERT INTO tabulka ( výčet atributů )
    VALUES ( výčet hodnot );
```

Vkládání záznamů - výčet atributů

- tuto nevýhodu odstraňuje použití syntaxu

```
INSERT INTO tabulka ( výčet atributů )
    VALUES ( výčet hodnot );
```

- Příklad:

```
INSERT INTO mereni (cas,id_senzor,id_velicina,hodnota)
  VALUES (now(),1,2,20);
```

Vkládání záznamů - výčet atributů

- tuto nevýhodu odstraňuje použití syntaxu

```
INSERT INTO tabulka ( výčet atributů )
    VALUES ( výčet hodnot );
```

- Příklad:

```
INSERT INTO mereni (cas,id_senzor,id_velicina,hodnota)
  VALUES (now(),1,2,20);
```

- zjednodušte si zápis volbou vhodných DEFAULT hodnot

```
INSERT INTO mereni (id_senzor,id_velicina,hodnota)
  VALUES (1,2,20);
```

Vkládání záznamů - SELECT

- Vložení záznamu je možné svázat se SELECTem

```
INSERT INTO tabulka [(výčet atributů)]
  SELECT ....
```

- Příklad: Vygenerujte měření pro všechny přípustné kombinace senzor veličina

```
INSERT INTO mereni(id_senzor,id_velicina,hodnota)
  SELECT id_senzor,id_velicina,random() * 70
    FROM senzorvelicina;
```

Aktualizace záznamů

- Aktualizace záznamů v tabulce - UPDATE
- Syntax:

```
UPDATE tabulka
    SET atribut = hodnota [, atribut = hodnota ]*
    [ WHERE podmínka ];
```

Aktualizace záznamů

- Aktualizace záznamů v tabulce - UPDATE
- Syntax:

```
UPDATE tabulka
    SET atribut = hodnota [, atribut = hodnota ]*
    [ WHERE podmínka ];
```

- Příklad: Upravte všechny hodnoty změřené senzorem s id = 2 násobením konstantou 1.2

```
UPDATE mereni
    SET hodnota=hodnota * 1.2
    WHERE id_senzor = 2;
```

Aktualizace záznamů

- Aktualizace záznamů v tabulce - UPDATE
- Syntax:

```
UPDATE tabulka
    SET atribut = hodnota [, atribut = hodnota ]*
    [ WHERE podmínka ];
```

- Příklad: Upravte všechny hodnoty změřené senzorem s id = 2 násobením konstantou 1.2

```
UPDATE mereni
    SET hodnota=hodnota * 1.2
    WHERE id_senzor = 2;
```

- POZOR: není-li zadaná podmínku WHERE, aplikují se změny na všechny záznamy v tabulce

Mazání záznamů

- Mazání záznamů v tabulce - DELETE
- Syntax:

```
DELETE FROM tabulka
    [ WHERE podmínka ];
```

Mazání záznamů

- Mazání záznamů v tabulce - DELETE
- Syntax:

```
DELETE FROM tabulka
    [ WHERE podmínka ];
```

- Příklad: Smažte všechna měření odpovídající senzoru s id = 5

```
DELETE FROM mereni
    WHERE id_senzor = 5;
```

Mazání záznamů

- Mazání záznamů v tabulce - DELETE
- Syntax:

```
DELETE FROM tabulka
    [ WHERE podmínka ];
```

- Příklad: Smažte všechna měření odpovídající senzoru s id = 5

```
DELETE FROM mereni
    WHERE id_senzor = 5;
```

- POZOR:
  - Pokud není deklarována podmínka WHERE
    - jsou smazány všechny záznamy v tabulce
  - Referenční integrity mohou zajišťovat kaskádní mazání

    (smažu-li senzor, smažou se i všechny záznamy referencující tento senzor)
  - Skutečně chci data nevratně smazat, nebo je jen zneplatnit

    (cena za pořízení dat, anonymizace osobních údajů)

Transakce

- problém transakčního zpracování dat
  - mnohé operace nejsou atomické, skládají se z více operací
  - např bankovní převod
    - 2x UPDATE: odečtení části z jednoho účty, připsání částky na účet

Transakce

- problém transakčního zpracování dat
  - mnohé operace nejsou atomické, skládají se z více operací
  - např bankovní převod
    - 2x UPDATE: odečtení části z jednoho účty, připsání částky na účet

1. první odčítám, druhé přičítám - poškozuji klienta
2. první přičítám, druhé odčítám - poškozuji banku

Transakce

- problém transakčního zpracování dat
  - mnohé operace nejsou atomické, skládají se z více operací
  - např bankovní převod
    - 2x UPDATE: odečtení části z jednoho účty, připsání částky na účet

    1. první odčítám, druhé přičítám - poškozuji klienta
    2. první přičítám, druhé odčítám - poškozuji banku

    - problém při přerušení zpracování požadavku jiným požadavkem

      vícenásobný přístup k účtu
  - pokud chyba v průběhu zpracování, nutno vrátit do původního stavu.

Transakce

- proto transakční zpracování
  - zajišťuje, že "logicky atomická" operace nebude narušena zásahem jiného uživatele

    (např. v důsledku vícevláknové zpracování)
  - postup:
    1. začátek transakce - BEGIN
    2. provedení operací
    3. ukočení

       pokud vše ok $\rightsquigarrow$ COMMIT

       pokud nastala chyba $\rightsquigarrow$ ROLLBACK vrací ovlivněné tabulky do stavu před BEGIN
  - POZOR: po dobu transakce je limitován přístup ostatním uživatelům - proto minimalizace doby transakce