Metody databáze (pohledy, uložené procedury, triggery)

Metody databáze

Databáze disponuje různými metodami zobrazení nebo možnostmi *předpřipravení* příkazů pro jednoduché budoucí (zejména opakované) použití.

Pohledy

Pohled je ve své podstatě pojmenovaný SELECT. Slouží k zjednodušení zápisu dotazů. Například v případě e-shopu, který chce zobrazovat jenom nearchivované produkty, ale archivované (ty, jenž už nejsou nabízeny k prodeji) si chce v databázi ponechat. Vytvoří se tedy následující pohled:

```
CREATE VIEW current_products AS SELECT * FROM products WHERE archived = 0;
```

Tento pohled se následně bude používat v dotazech namísto tabulky products. Ze všech dotazů tak zmizí část `WHERE archived = 0`. Příklad načítání produktů z aplikace:

```
SELECT * FROM current_products;
```

Pohledy je možné vytvářet jen uvnitř jednoho databázového serveru, může se však odkazovat na tabulky v jiné lokální databázi. Nelze je editovat, k tomuto nejsou pohledy ani určeny. Úprava dat v pohledu by znamenala data upravit v původní databázi, na jejíž data pohled pohlíží.

Uložené procedury

Uložená procedura je sada příkazů SQL, které jsou uložené na databázovém serveru, vykonávaná tak, že je zavolána prostřednictvím dotazu názvem, který jí byl přiřazen (jde tedy o jakousi obdobu funkce). Ulehčuje správu databázových aplikací.

Mezi výhody patří možnost volat shodnou proceduru (vykonat stejný kód) jejím jednoduchým zavoláním a případně předáním parametrů. Je-li potřeba výkonný kód upravit, může se to provést na jediném místě – není potřeba jakýmkoliv způsobem zasahovat do aplikací, které s databází komunikují. Uložené procedury mohou přispívat k zabezpečení serveru – mohou být nastaveny tak, že je smí spouštět pouze uživatel s oprávněním a procedury samy mohou kontrolovat počet, typ, velikost a další charakteristiky parametrů, které jsou jim posílány, což může umožnit vyhnout se SQL injection.

Uložené procedury jsou obecně schopné běžet rychleji, než kdyby se měl příslušný kód vykonávat z klientské aplikace. Důvodem je fakt, že uložené procedury jsou na serveru zkompilovány[1].

[1] zkompilováním je myšleno, že si server pro uloženou proceduru vytvoří a spravuje tzv. *prováděcí plán*, díky kterému je subsystém serveru zvaný optimalizátor obvykle schopen najít nejrychlejší způsob, jak uloženou proceduru provést a potom tento způsob opakovaně používat při jednotlivých voláních této uložené procedury

Vytvoření a volání procedury

Procedura pro vrácení veškerého uloženého softwaru. Níže je pak zavolána.

```
CREATE PROCEDURE vratSoftware()
BEGIN
    SELECT * FROM software;
END //

CALL vratSoftware();
```

Procedura pro vrácení konkrétního záznamu (např. podle kódu produktu). Níze je pak zavolána.

```
CREATE PROCEDURE vratProdukt(IN kodProduktu CHAR(8))
BEGIN
    SELECT * FROM produkty WHERE kod = kodProduktu;
END //

CALL vratProdukt("ABC12345");
```

Triggery

Trigger je pak nějaká uložená procedura, která se spouští automaticky v souvislosti s provedením nějakého akčního dotazu v tabulce (každý trigger patří právě jedné tabulce). Dále se od uložených procedur liší nemožností předávat nějaké vstupní parametry nebo něco vracet.

V triggeru se nesmí objevit následující příkazy: `ALTER`, `START TRANSACTION`, `ROLLBACK`, `COMMIT`, `GRANT`, `REVOKE`, `CALL` a některé další.

Výhodou však je, že triggery mají přístup k datům, která se právě mění. Tzn. Trigger, který se spouští před aktualizací nějaké tabulky, má přístup k hodnotám těch řádků, které mají být změněny. Před úpravou nějakého záznamu je tak například možné automaticky uložit původní záznam do tabulky *archiv*.

Ukázka triggeru, který před odstraněním záznamu záznam uloží do tabulky *zaloha*:

```
CREATE TRIGGER backupDeleted
BEFORE DELETE
ON user
FOR EACH ROW
BEGIN
    INSERT INTO zaloha(id, username, password, email, delete_time, delete_user)
    VALUES (old.id, old.username, old.password, old.email, now(), user());
END;
```