DUM 12 téma: Příkazy pro tvorbu databáze

ze sady: 3 **tematický okruh sady:** III. Databáze
ze šablony: 7 – Kancelářský software **určeno pro:** 4. ročník
vzdělávací obor: 18-20-M/01 Informační technologie
vzdělávací oblast: odborné vzdělávání
metodický list/anotace: viz VY_32_INOVACE_07312ml.pdf

Databázový server

Jak již bylo zmíněno, v relačních databázích slouží databázový server – stroj provozující SŘBD – k obsluze mnohdy velkého množství databází, často i pro nesouvisející účely či zákazníky. Samotný provoz takového serveru pak může být externě dodávanou službou, kdy je pouze věcí uživatele, jaké databáze a tabulky v rámci své přidělené kapacity vytvoří a využije. Druhým způsobem využití databázového serveru je externě dodaná aplikace, která si sama při prvním spuštění potřebné databáze a tabulky v nich vytvoří.

Pro tyto účely zahrnuje systém SQL i subjazyk skupiny DDL – Data Definition Language. Samozřejmě k jeho využití musí mít uživatel příslušná práva k databázovému serveru.

Struktura databáze a tabulek

Data jsou na databázovém serveru uložena způsobem znemožňujícím přímý přístup uživatele – z důvodu jak efektivity, tak hlavně bezpečnosti samotných dat. Obdobně probíhá i tvorba samotné databázové struktury, kdy databázovému severu pouze předáváme formalizovaným způsobem naši představu o vzhledu této struktury, a on pak již svými prostředky (soubory na disku, data v paměti…) sám data ukládá a prezentuje navenek. Z pohledu uživatele, pak rozlišujeme úrovně databáze-tabulka-datová věta-datová položka. Pro jednoduché základní aplikace si vystačíme s jednou databází a v ní umístěnými tabulkami.

Podstatný je vztah datového modelu ve formě E-R diagramu a virtuální databázové struktury, která jej v SŘBD reprezentuje. Samotnou fyzickou strukturu dat před námi SŘBD skrývá a je mimo rozsah tohoto dokumentu.

Zjednodušeně lze zavést vztahy mezi prvky E-R diagramu a reálné databáze:

E-R diagram	Databáze
Entita	Bazická tabulka
Relace M:N	Relační (vazební) tabulka
Relace 1:1 či 1:N	Pravidlo referenční integrity

Přičemž relace typu M:N většinou rozkládáme na dvojici 1:N již ve fázi návrhu.

Tvorba a změny tabulek

Jazyk SQL disponuje příkazy CREATE a ALTER pro vytváření a změny hlavních objektů jako databáze a tabulky.

Příkaz CREATE

V základní syntaxi lze zapsat jako:

CREATE TABLE *tabulka*

Který vytvoří prázdnou tabulku – bez jakékoliv vnitřní struktury. Takový přístup se v praxi používá například u generovaných databázových struktur – kdy jsou nejprve vytvořeny prázdné tabulky a až následně pomocí ALTER (viz dále) je jim doplněna struktura.

V naší praxi však spolu s požadavkem na tvorbu tabulky spojíme i naši představu o její vnitřní struktuře (reprezentované představou „sloupců“):

CREATE TABLE *tabulka* (*sloupec TYP, sloupec TYP…*)

Zde je velmi vhodné definovat názvy sloupců bez diakritiky, jednoznačně a jednoslovně (například s využitím znaku „_“ (datum_vystaveni)) a podobně. Datových typů podporují databáze založené na SQL celou řadu, uvedeme jen základní:

Označení	Typ	Popis
INT	celočíselný	4bajtový znaménkový typ cca -2 miliardy až +2miliardy
FLOAT	číselný	Desetinné číslo zvolené přesnosti
VARCHAR	znakový	Řetězec s danou maximální délkou
TEXT	znakový	Do 64 kByte
LONGTEXT	znakový	Do 4 GByte
DATETIME	Datum+čas	Kombinovaný údaj o datu a čase s přesností na sekundy

Poznámka:Přesný popis a řadu dalších datových typů lze dohledat v referenční příručce MySQL.

V praktickém příkladu, pak tabulku odpovídající entitě vytvoříme takto:

Entita:

auto
vin
barva
rok_vyroby
znacka
hmotnost

Bude převedena na tabulku:

CREATE TABLE auto (vin VARCHAR(17), barva VARCHAR(20), rok_vyroby INT, znacka VARCHAR(25), hmotnost INT)

Tento příkaz se v dokumentaci a literatuře pro přehlednost zapisuje rozdělený do řádek – interpretu SQL odřádkování nijak nevadí, za konec řádku považuje až znak „;“:

CREATE TABLE auto (vin VARCHAR(17),

barva VARCHAR(20),

rok_vyroby INT,

znacka VARCHAR(25),

hmotnost INT)

Zavedení primárního klíče do tabulky zajistíme rozšířením definice (popisy sloupců budeme v dalším textu již pro zkrácení vynechávat:

CREATE TABLE *tabulka* (*popisy_sloupců*, PRIMARY KEY (`*sloupec*`))

Systém pak již sám kontroluje dodržování pravidel pro primární klíč – unikátnost, nenulovost a podobně. Primární klíč lze alternativně zavést i jako vlastnost konkrétního sloupce „PRIMARY KEY“.

I pro ostatní sloupce pak lze nastavit některé specifické vlastnosti, které bude SŘBD následně sám kontrolovat:

NOT NULL – položka nesmí mít v řádné datově větě hodnotu NULL

AUTO_INCREMENT – při vytvoření nové datové věty se hodnota položky automaticky zvýší o 1 oproti naposledy vytvořené větě

Například:

```
CREATE TABLE zamestnanci (
    id_cloveka INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
    jmeno VARCHAR(20)
    )
```

Zvláštním způsobem tvorby tabulky je zkopírování existující, případně její klonování. Klonováním tabulky vytvoříme novou tabulku s identickou strukturou, ovšem bez dat:

CREATE TABLE *nova* LIKE *stara*

Kdežto kopírování vytvoří duplikát včetně obsažených dat:

CREATE TALBE *nova* SELECT * FROM *stara*

Takový postup využíváme často spolu s direktivou TEMPORARY, kdy je vytvořena tabulka pouze v paměťovém prostoru databázového serveru a odstraněna spolu s ukončením komunikace klienta se serverem:

CREATE TEMPORARY TABLE …

Používáme ho například pro testování potenciálně nebezpečných operací, před jejich aplikací na „ostré“ databázové tabulky, případně při konverzích živých dat. Výhodou je zde i možnost existence více tabulek stejného jména – pro každého klienta s jiným obsahem.

Změny struktury

Pro změny struktury existujících tabulek a databází disponuje SQL příkazem ALTER. Je velmi důležité upozornit, že jakékoliv manipulace nad existujícími tabulkami jsou kritickou operací, která velmi snadno může nenávratně data poškodit. Relativně „bezpečnou“ operací je přidávání sloupců nebo rozšiřování sloupců VARCHAR a podobně. Problémem již může být naopak zužování textových či číselných sloupců, změny formátů číselných položek a samozřejmě odebírání sloupců. SŘBD nedovolí provádět změny nad používanými sloupci s primárními klíči, nebo které jsou součástí referenčních integrit a podobně.

Příklad „bezpečné“ operace:

ALTER TABLE zamestnanci ADD vyska INT

Operace měnící sloupce:

ALTER TABLE zamestnanci MODIFY jmeno VARCHAR(30)

Zde záleží, jaká byla předchozí šířka sloupce „jmeno“ – zda provádíme rozšíření, či zužování. Poznámka: v některých implementacích SQL místo „MODIFY“ používáme „ALTER“.

Kritickým je vždy odstranění sloupce:

ALTER TABLE zamestnanci DROP titul;

Logicky nelze pomocí ALTER provádět operace, které nejsou logické. Například do existující tabulky doplnit sloupec s vlastností NOT NULL či UNIQUE, kdy by pro již existující řádky nebyly podmínky ihned po vytvoření sloupce splněny. Velmi obtížné je dodatečné vytvoření primárního klíče, kdy musí uživatel předem zajistit splnění podmínek pro kandidátský klíč v daném sloupci.

Úkoly pro samostatnou práci

Vytvořte tabulku „knihy“ se strukturou:

- Signatura – textový řetězec zvolené délky
- Název – text
- Rok vydání – číslo
- Autor – text, nebo jiný typ identifikátoru
- Počet stran – číslo

Kde signatura bude primární klíč a název musí být vyplněn

Doplňte tabulku o textový sloupec vydavatel

Vytvořte dočasnou kopii tabulky knihy včetně jejího obsahu s názvem „predavane_knihy“

Zdroje:

- Archiv autora