Databázové systémy

Modifikácia DDL a DML

2026 Lukáš Miňo

Motivácia

- Keď navrhnete databázu na papieri ako semestrálnu prácu, vyzerá dokonale. Ale v momente, keď aplikáciu spustíte na produkcii pre reálnych ľudí, začína sa kolotoč zmien. Používatelia vám začnú meniť dáta pod rukami (DML) a produktový manažér príde s tým, že od zajtra musíme kvôli novému zákonu ukladať dodatočné informácie o faktúrach (DDL). Schopnosť bezpečne modifikovať dáta a štruktúru za behu bez toho, aby ste zhodili celý systém, je to, čo odlišuje teoretika od skutočného databázového špecialistu.

Motivácia

- **1. Svet dát sa mení každú milisekundu (DML)**
- **Používateľská aktivita:** Žiadna databáza nie je len na čítanie (**SELECT**). Používatelia neustále generujú nový stav – registrujú sa (**INSERT**), menia si heslá či adresy (**UPDATE**) alebo rušia neaktívne účty (**DELETE**).
  - **Biznis kontinuita:** Ak by sme nevedeli modifikovať dáta, e-shop by zamrzol v čase po prvom nahratí produktov.
- **2. Softvér sa vyvíja a s ním aj štruktúra (DDL)**
  - **Evolúcia aplikácie:** Predstavte si, že váš e-shop expanduje do zahraničia. Zrazu stĺpec telefon INT nestačí (potrebujete predvoľby ako +421) alebo musíte pridať úplne nový stĺpec pre IBAN.
  - **Legislatívne zmeny:** Príchod nariadení ako GDPR si vynucuje okamžitú zmenu metadát – napríklad pridanie stĺpca pre explicitný súhlas so spracovaním údajov alebo úplné odstránenie starých štruktúr.
- **3. Agilný vývoj a CI/CD (Continuous Integration / Continuous Delivery)**
  - Moderné IT firmy nasadzujú nové verzie aplikácií niekoľkokrát týždenne. Každá nová funkcia v aplikácii si často vyžaduje okamžitú, bezpečnú a automatizovanú úpravu existujúcej databázovej schémy pomocou príkazu **ALTER_TABLE**.

DML vs. DDL

- **Hlavná myšlienka:** Relačné systémy prísne oddeľujú dáta od ich štruktúry.
- **DML (Data Manipulation Language):** Slúži na prácu s inštanciou tabuľky (záznamami). Ovplyvňuje riadky.
- **DDL (Data Definition Language):** Slúži na definíciu databázových objektov. Ovplyvňuje databázové objekty.
- **Architektonický rozdiel:** Kým DML operácie podliehajú transakčnému spracovaniu (ACID) a dajú sa odvolať (ROLLBACK), DDL operácie v mnohých DBMS (napr. MySQL) spôsobujú **implicitný commit** a ich spätný chod nie je triviálny.
-

Data Manipulation Language

INSERT

- **Podstata:** Pridanie novej n-tice (riadku) do relácie.
- **Kontext e-shopu:** Registrácia nového zákazníka alebo vytvorenie transakcie.
- **Odborné prístupy k zápisu:**
  - **Pozičný zápis** (Implicitný – náchylný na chyby pri zmene schémy).
  - **Explicitný zápis stĺpcov** (Najlepší postup pre produkčný kód).
  - **Hromadný zápis** (Bulk INSERT pre optimalizáciu I/O operácií).

- **Pozičný** (neodporúča sa):
- INSERT INTO zakaznik VALUES (1, 'Ján Novák', 'jan@novak.sk‘);
- **Explicitný** (garantuje stabilitu kódu):
- INSERT INTO zakaznik (zakaznik_id, meno, email) VALUES (2, 'Eva Malá', 'eva@mala.sk‘);
- **Hromadný** (bulk):
- INSERT INTO zakaznik (zakaznik_id, meno, email) VALUES (3, 'A', 'a@a.sk'), (4, 'B', 'b@b.sk');

UPDATE

- **Podstata:** Zmena hodnôt atribútov v už existujúcich riadkoch.
- **Kritické riziko: Chýbajúca klauzula WHERE.** Bez nej dôjde k deštrukcii dát v celej tabuľke (zmena stavu všetkých objednávok).

```
-- Bezpečný update konkrétnej entity
UPDATE objednavka
SET stav = 'Zaplatená'
WHERE id = 4521;

-- Update s využitím aritmetického výrazu
UPDATE produkt
SET cena = cena * 1.10
WHERE kategoria = 'Elektronika'; -- Plošné zvýšenie o 10%
```

DELETE vs. TRUNCATE

- **DELETE FROM (DML):** Odstraňuje riadky na základe podmienky WHERE. Generuje log pre každý zmazaný riadok (náročné na transakčný log). Aktivuje ON DELETE triggre.
- **TRUNCATE TABLE (DDL/DML hybrid):** Okamžite uvoľní dátové bloky na disku. Vymaže *všetky* dáta naraz. Obchádza DML triggre. Je extrémne rýchly, ale spravidla sa nedá odvolať príkazom ROLLBACK.

```
-- Selektívne mazanie (DML)
DELETE FROM objednavka WHERE stav = 'Zrušená';

-- Kompletné vyčistenie tabuľky (rýchla operácia nad metadatami)
TRUNCATE TABLE objednavka;
```

Data Definition Language

CREATE

- **Podstata:** Vytvorenie nového objektu v databáze (tabuľka, index, pohľad, trigger).
- **Definícia fyzického modelu:** Mapovanie logických atribútov na fyzické dátové typy a priradenie integritných pravidiel.

```
CREATE TABLE faktura (
   id INT NOT NULL,
   objednavka_id INT NOT NULL,
   suma DECIMAL(10,2) NOT NULL,
   vyhotovena DATE DEFAULT CURRENT_DATE,

   CONSTRAINT pk_faktury PRIMARY KEY (id)
);
```

ALTER

- **Podstata:** Zmena štruktúry existujúceho objektu bez nutnosti jeho deštrukcie a opätovného vytvárania.
- **Operácie v praxi:** Pridanie stĺpca (ADD), úprava dátového typu (ALTER/MODIFY) a odstránenie stĺpca (DROP COLUMN).

```
-- 1. Pridanie nového atribútu
ALTER TABLE zakaznik ADD telefon VARCHAR(20);

-- 2. Zmena dátového typu / vlastnosti stĺpca (Rozdiely v syntaxi!)
ALTER TABLE zakaznik ALTER COLUMN meno TYPE VARCHAR(150); -- PostgreSQL
ALTER TABLE zakaznik MODIFY meno VARCHAR(150) NOT NULL;         -- MySQL / Oracle
ALTER TABLE zakaznik ALTER COLUMN meno VARCHAR(150) NOT NULL;     -- MS SQL

-- 3. Odstránenie stĺpca zo schémy
ALTER TABLE zakaznik DROP COLUMN telefon;
```

DROP

- **Podstata:** Kompletné odstránenie objektu a jeho metadát z databázy vrátane všetkých uložených dát.
- **Zápis a bezpečnosť:** Nebezpečná operácia, v produkcii sa ošetruje pomocou klauzuly **IF EXISTS**.

```
-- Bezpečné odstránenie pohľadu
DROP VIEW IF EXISTS mesacne_trzby_view;

-- Odstránenie tabuľky vrátane závislých objektov (cudzích kľúčov)
DROP TABLE zakaznik;
```

Ako modifikujú metadáta softvérové firmy?

Ako modifikujú metadáta softvérové firmy?

- Ručné zadávanie ALTER TABLE príkazov do produkčnej konzoly je zakázané (riziko ľudskej chyby, nekonzistentnosť prostredí).
- Používajú sa **migračné nástroje**.
- Každá zmena metadát (DDL) je zapísaná ako verziovaný SQL skript, ktorý prechádza cez Git (Code Review) a automaticky sa spúšťa pri nasadení novej verzie aplikácie.

Zhrnutie a rady

- **DML vždy s podmienkou**
  - Pri príkaze UPDATE a DELETE trikrát overte klauzulu WHERE, ideálne si ju najprv otestujte pomocou SELECT.
- **Preferujte explicitný INSERT**
  - Nikdy sa v kóde nespoliehajte na poradie stĺpcov v databáze.
- **DDL na živej databáze plánujte mimo špičky (počas minimálnej prevádzky systému)**
  - Zmena dátového typu alebo indexovanie miliónovej tabuľky môže uzamknúť databázu (Table Lock) a spôsobiť výpadok e-shopu.
- **Izolujte testovacie data**
  - Všetky deštruktívne príkazy testujte na staging/dev databáze s anonymizovanými dátami.

Zhrnutie a rady pre ALTER

- **Nikdy nespúšťajte ALTER ručne na produkcii**
  - Všetky zmeny schémy musia byť automatizované cez migračné nástroje a verziované v Gite.
- **Pred pridaním obmedzenia vyčistite data**
  - Ak idete pridať CHECK (cena > 0), najprv spravte SELECT a opravte všetky historické riadky, kde bola cena nulová. Inak ALTER zlyhá.
- **Pozor na dopad na aplikačný kód**
  - Ak pomocou ALTER zmažete stĺpec alebo zmenjte jeho typ, uistite sa, že vaša backend aplikácia (Java, Python…) už tento stlpec nikde v SQL dopytoch nepoužíva.
- **Náročné zmeny plánujte na nočné hodiny**
  - Ak operácia vyžaduje prepísanie celej tabuľky na disku, vykonajte ju v čase najnižšej prevádzky (maintenance window).