Bazy danych i systemy zarządzania

Wykład IX

Elementy języka SQL

Część III

©Antoni Ligęza

Organizacja obiektów w SQL

Struktury Danych

Wiersze i kolumny: wiersze = rekordy; kolumny – odpowiadają poszczególnym rekordom,

Tablice: tabele z danymi; kolumny etykietowane są atrybutami, wiersze (rekordy) wybierane są za pomocą klucza; rodzaje tablic:
– tablice podstawowe zawierające dane,
– perspektywy – tablice wirtualne generowane przy pomocy zapytań,
– globalne tablice tymczasowe, zdefiniowane stale, zawartość nie jest przechowywane w bazie,
– lokalne tablice tymczasowe, zdefiniowane stale, zawartość nie jest przechowywane w bazie,
– zadeklarowane lokalne tablice tymczasowe, nie są zdefiniowane na stałe.

Schemat: grupa tablic znajdująca się pod kontrolą jednego użytkownika,

Katalog: grupa schematów,

Klaster: grupa katalogów; wszystkie tablice dostępne w czasie sesji (połączenia) muszą być w tym samym klastrze; tablice powiązane ze sobą muszą być w tym samym klastrze.

Kursor: obiekt w którym jest przechowywane wyjście zapytania do dalszego wykorzystywania w programie; kursory mogą być definiowane jako obiekty *tylko do odczytu, ang. read-only*, *niewrażliwe, ang. insensitive* (ignorujące zmiany danych podczas czytania), *przwijalne, ang. scroll* (zachowujące pewien porządek wierszy); kursory tego typu umożliwiają nawigację po rekordach; kursory mogą być statyczne, oraz *dynamiczne*, używane w dynamicznym SQL-u, gdy z góry nie wiadomo jakie zapytanie będzie w nim zawarte (zmienna łańcuchowa).

©Antoni Ligęza

Definiowanie tabel

Do definiowania tabel służy instrukcja `CREATE TABLE` o schemacie:

```
CREATE [{GLOBAL | LOCAL} TEMPORARY]  TABLE na-
zwa tablicy
   ({ definicja kolumny
   | [ograniczenie tablicy] }.,...
      [ON COMMIT {DELETE | PRESERVE} ROWS] );

definicja kolumny::=
   nazwa kolumny {nazwa domeny
                  | typ danych [rozmiar]  }
                  [ograniczenie kolumny...]
                  [DEFAULT wartosc domyslna ]
                  [COLLATE nazwa porownania ]
typ danych::= CHARACTER STRING | NATIONAL CHARAC-
TER |  BIT STRING |
               EXACT NUMERIC | APPROXIMATE NUME-
RIC | DATETIME | INTERVAL
```

Przykładowa, komenda definiująca strukturę tablicy:

```
CREATE TABLE KLIENCI
   (NumerKli INTEGER,
   ImieKli VARCHAR (15) NOT NULL,
   NazwiskoKli VARCHAR (20) NOT NULL,
   DataKli DATE NOT NULL DEFAULT CURRENT_DATE,
   UlicaKli VARCHAR (30),
   MiastoKli VARCHAR (2),
   KodPocztKli CHAR (5),
   TelefonKli CHAR (10)  NOT NULL,
   EmailKli VARCHAR (40),
   PRIMARY KEY (NumerKli));
```

spowoduje utworzenie tabeli opisującej klientów wg podanej specyfikacji.

©Antoni Ligęza

Perspektywy

Perspektywy (widoki, ang. *view*) tworzy się jako złączenia wybranych tabel za pomocą instrukcji `SELECT`. Perspektywa tworzy wirtualną tabelę dostosowaną do potrzeb konkretnego użytkownika. Przy tworzeniu perspektyw możliwe jest:

- złączenie dowolnie wybranych tabel,
- wykonanie operacji selekcji, projekcji i sortowania,
- wykonanie instrukcje grupowania,
- użycie podzapytania.

Standardowa składnia definicji instrukcji tworzenia perspektywy:

```
CREATE VIEW <nazwa perspektywy> [  (<nazwa kol> [, <nazwa kol> .
        AS
        (SELECT <instrukcja>
        [WITH [CASCADED|LOCAL] CHECK OPTION] );
```

Listy kolumn używa się gdy:

- jakiekolwiek dwie kolumny mają identyczne nazwy,
- koluny zawierają wartości wyliczalne,
- występują połączone kolumny o różnych nazwach.

Kasowanie perspektyw:

```
DROP VIEW <nazwa perspektywy> [CASCADE|RESTRICT];
```

`CASCADED` – sprawdza predykaty wszystkich warstw przy zmianach (`LOCAL` – nie),
`CHECK OPTION` – sprawdza, czy instrukcje `INSERT` lub `UPDATE` nie wstwiają do tablicy bazowej wierszy niezgodnych z definicją perspektywy.

©Antoni Ligęza

Przykłady tworzenia perspektyw

```
CREATE VIEW oaklanders
        AS
        SELECT au_fname, au_lname, title
        FROM authors, titles, titleauthors
        WHERE authors.au_id = titleauthors.au_id
          AND titles.title.id = titleauthors.au_id
          AND city = 'Oakland';

CREATE VIEW currentinfo (PUB, TYPE, IN-
COME, AVG_PRICE, AVG_SALES)
        AS
        SELECT pub_id, type, sum(price*ytd_sales), avg(price), a
        FROM titles
        GROUP BY pub_id, type;

CREATE VIEW cities (Author, Author-
city, Pub, Pubcity)
        AS
        SELECT au_lname, authors.city, pub_name, publishers.city
        FROM authors, publishers
        WHERE authors.city = publishers.city;
```

Dane w tablicach bazowych mogą być modyfikowane (wstawiane, usuwane) z poziomu perspektywy tylko przy spełnieniu określonych wymagań:

- zmiany muszą być określone jednoznacznie.
- perspektywa oparta jest na jednej tablicy bazowej; odniesienia są tylko do kolumn tej tablicy,
- zawiera tylko jedno zapytanie (bez `UNION`, `EXCEPT`, `INTERSECT`),
- w definicji zapytania nie występują funkcje agregujące (nie ma `GROUP BY`, `HAVING`, DISTINCT).

©Antoni Ligęza

Wyzwalacze

Wyzwalacz (trigger) jest zestawem instrukcji SQL definiujących akcję, która ma być wykonana automatycznie, po zajściu określonego zdarzenia. Wyzwalacze wykonują modyfikację danych (`UPDATE, INSERT, DELETE`).

```
CREATE TRIGGER <nazwa _wyzwalacza>
        ON <nazwa_tabeli>
        FOR {INSERT | UPDATE | DELETE}
           [,  {INSERT | UPDATE | DELETE}]...
        AS <instrukcje_SQL>
           [IF UPDATE (nazwa_kolumny)
           [{AND | OR} UPDATE (nazwa_kolumny)]...]

CREATE TRIGGER delcascadetrig    ON titles
        FOR DELETE
           [,  {INSERT | UPDATE | DELETE}]...   AS
           DELETE titleautors
           FROM titleautors, deleted
           WHERE titleautors.title_id = deleted.title_id
           [IF UPDATE (nazwa_kolumny)
           [{AND | OR} UPDATE (nazwa_kolumny)]...]
           /* Usuwa wiersze z titleautors zgod-
nie z usunietymi wierszami (titles) */
           DELETE salesdetails
           FROM salesdetails, deleted
           WHERE salesdetails.title_id = deleted.title_id
           /* Usuwa wiersze zlecenia sprzedazy zgod-
nie z usunietymi wierszami (titles) */
           delete royshed
           FROM royshed, deleted
           WHERE royshed.title_id = deleted.title_id
           /* Usuwa wiersze z royshed zgodnie z usu-
nietymi wierszami (titles)
```

©Antoni Ligęza

Elastyczne zarządzanie transakcjami

Transakcje

Transakcje: to grupy kolejnych instrukcji SQL realizowanych w formie sekwencji zakończonej sukcesem albo porażką. Porażka powoduje anulowanie transakcji (i jej efektów częściowych).

Zatwierdzenie transakacji dokonuje się instrukcją `COMMIT [WORK]`.

Anulowanie transakcji dokonuje się za pomocą instrukcji `ROLLBACK`.

Transkacja może skończyć się z powodu awariii systemu, zerwania połączenia, etc.

Przy przetwarzaniu transakcyjnym obowiązują następujące zasady:

- SZBD automatycznie rozpoczyna transakcję po wywołaniu jej instrukcją, gdy nie jest aktywna inna transakcja,
- Jeżeli transakcja nie może być zakończona z zachowaniem zmian, będzie ona wycofana,
- Transakcje mogą być tylko do odczytu (nie powodują potrzeby blokowania danych),
- W trakcie transakcji można opóźniać kontrolę ograniczeń (`SET CONSTRAINTS MODE`),
- Transakcje mogą określać poziom izolacji blokady nałożonej na dane (`SET TRANSACTION`); standard SQL-92 (ISO) określa cztery poziomy izolacji: `READ UNCOMMITTED`, `READ COMMITTED`, `READ REPETABLE`, `SERIALIZABLE`,
- Transakcje mogą określać rozmiar obszaru diagnostycznego dla swoich instrukcji.

©Antoni Ligęza

Zasady realizacji transakcji – ACID

Zasady ACID przetwarzania transakcyjnego

Niektóre (wielodostępne) SZBD umożliwiają jednoczesne przetwarzanie wielu transakcji (np. systemy rezerwacji miejsc, systemy bankowe, etc.). Poprawność i kompletność realizacji wszelkich operacji gwarantuje *moduł zarządzania transakcjami*. Poprawność opisana jest czterema poniższymi zasadami ACID:

- **Niepodzielność (Atomicity):** Wykonywana jest albo cała transakcja (od początku do końca, albo żaden jej element nie może zostać zeralizowany (nie jest dopuszczalna realizacja częściowa),
- **Spójność (Consistency):** Baza danych musi zachować spójność, dane po wykonaniu transakcji muszą być zgodne z nałożonymi ograniczeniami,
- **Isolacja (Isolation):** jeżeli dwie lub więcej transakcji jest przetwarzanych jednocześnie, nie mogą one wzajemnie na siebie oddziaływać; w wyniku jednoczesnego ich przetwarzania nie może zdarzyć się nic, co nie zdarzyłoby się, gdyby były one przetwarzane po kolei,
- **Trwałość (Durability):** Po zakończeniu transakcji jej wynik nie może zostać utracony (np. z powodu awarii systemu).

Blokady: Realizacja zasad ACID (zasadnicza idea) polega na blokowaniu dostępu do pewnych elementów bazy danych podczas realizacji transakcji.

Granulacja blokad: SZBD różnią się rozmiarami elementów danych, które są poddawane blokowaniu (np. blokady na poziomie rekordów, bloków na dysku, plików, relacji).

©Antoni Ligęza

Poziomy zgodności

Do definiowania poziomu izolacji transakcji służy instrukcja:

```
SET TRANSACTION { ISOLATION LEVEL
          {READ UNCOMMITTED | READ COMMITTED
           REPETABLE READ | SERIALIZABLE }
          | { READ ONLY | READ WRITE }
          | { DIAGNOSTICS SIZE ilosc warunkow }};
```

Istnieją następujące przypadki współdziałania pomiędzy transakcjami współbieżnymi:

- *Dirty read:* pierwsza transakcja modyfikuje rekord, a druga go czyta zanim zmiana została zachowana przez `COMMIT`; jeśli pierwsza transakcja zostanie wycofana, to druga z nich przeczytała wiersz, który nie istnieje.
- *Non-repetable read:* pierwsza transakcja czyta wiersz, a druga usuwa go lub modyfikuje i wykonuje `COMMIT`; teraz pierwsza transakcja czytając ponownie wiersz otrzyma inne wartości,
- *Phantom:* pierwsza transakcja odczytuje rekordy spełniające pewien predykat; druga transakcja wstawia lub modyfikuje rekordy, tak, że nowe rekordy spełniają też dany predykat – następne wykonanie tego samego zapytania przez pierwszą transakcję da inny wynik.

POZIOM IZOLACJI	Dirty read	Non-repetable	Phantom
READ UNCOMMITTED	TAK	TAK	TAK
READ COMMITTED	NIE	TAK	TAK
REPETABLE READ	NIE	NIE	TAK
SERIALIZABLE	NIE	NIE	NIE

©Antoni Ligęza

Spójność: domeny, ograniczenia i asercje

Domeny

W SQL 92 możliwe jest definiowanie *domen* jako dziedzin lub typów obiektów i atrybutów. Definicja domeny składa się z typu danych, definicji wartości domyślnej, ograniczenia wartości, sekwencji porządkującej, etc. Definiowane domeny mogą być modyfikowane i usuwane. Służą one do definiowaniu zbiorów ograniczeń, głównie dla definicji atrybutów..

Ograniczenia

Ograniczenia to reguły integralności danych. Dotyczyć one mogą pojedynczych kolumn tablic lub ich zbiorów (integralność na poziomie rekordu tablicy). W standardzie SQL 92 możliwe jest definiowanie *asercji*, tzn. ograniczeń istniejących niezależnie w schemacie; takie ograniczenia mogą odnosić się do wielu tablic. Asercje pozwalają na zdefiniowanie ogólnych zasad, które muszą spełniać dane. Asercje można tworzyć i usuwać.

Opóźnianie ograniczeń

Istnieje możliwość opóźniania sprawdzania ograniczeń lub asercji, co może być użyteczne, jeżeli chcemy sprawdzać ograniczenia po zakończeniu całej transakcji. Ograniczenia można sprawdzać:

- po każdej instrukcji odnoszącej się do tablicy,
- na końcu każdej transakcji zawierającej przynajmniej jedną instrukcję modyfikacji w tabeli,
- zawsze, gdy użytkownik uzna to za konieczne.

©Antoni Ligęza

Grupy instrukcji SQL

`ALLOCATE CURSOR, ALLOCATE DESCRIPTOR` – tworzenie powiązań,
`ALTER DOMAIN, ALTER TABLE` – definiowanie modyfikacji,
`CLOSE` – zamykanie kursora,
`COMMIT WORK` – zatwierdzanie transakcji,
`CONNECT` – połaczenie w architekturze klient-serwer,
`CREATE` (`ASSERTION, CHARACTER SET, COLLATION, DOMAIN, SCHEMA, TABLE, VIEW`) – tworzenie obiektów bazy danych,
`DEALLOCATE DESCRIPTOR, DEALLOCATE PREPARE` – niszczy element,
`DECLARE CURSOR, DECLARE LOCAL TEMPORARY TABLE` – tworzenie wybranych elementów,
`DELETE` – usuwa rekordy z tablicy,
`DESCRIBE` – dostarcza informacji o instrukcji,
`DISCONNET` – kończy połączenie,
`DROP` (`ASSERTION, CHARACTER SET, COLLATION, DOMAIN, SCHEMA, TRANSLATION, VIEW`) – usuwa wybrany element,
`EXECUTE [IMMEDIATE]` – wykonuje przygotowaną instrukcję,
`FETCH` – pobiera wiersze z otwartego kursora,
`GET` (`DESCRIPTOR, DIAGNOSTICS`) – pobiera/zwraca infomnacje,
`GRANT` – nadaje uprawnienia użytkownikom,
`INSERT` – wstawia wiersze do tablicy,
`OPEN` – przygotowuje kursor do użycia,
`PREPARE` – tworzy instrukcję SQL,
`REVOKE` – odwołuje uprawnienia,
`ROLLBACK` – unieważnia transakcję,
`SELECT` – wyszukuje zadane rekordy,
`SET` (`CATALOG, CONNECTION, CONSTRAINT MODE, DESCRIPTOR, NAMES, SCHEMA, SESSION AUTHORIZATION, TIME ZOPNE, TRANSACTION` – określa domyślny element,
`UPDATE` – zmienia wartości w tabel.

©Antoni Ligęza

Podsumowanie

Zalety SQL

SQL jest ukierunkowany, ale i ograniczony do zastosowań w relacyjnych bazach danych. Pozwala w praktyce realizować wszystkie operacje algebry relacji, operacje grupowania danych, obliczenia, etc. Zalety SQL obejmują:

- deklaratywny charakter (brak konieczności definiowania *jak*, a tylko *co* użytkownik chce uzyskać,
- wysoki poziom abstrakcj ale i selektywny dostę do danyc,h
- efektywne mechanizmy realizacji zapytań,
- czytelność i przejrzystość,
- standaryzację,
- dobre podstawy matematyczne (algebra relacji).

Problemy i ograniczenia SQL

Istotniejsze ograniczenia SQL wynikają z jego założeń i obejmują:

- operowanie jedynie na strukturach tablicowych; brak możliwości reprezentacji i przetwarzania innych struktur, a w tym,
- brak możliwości definiowania i przetwarzania termów i list,
- ograniczenie do danych atomicznych,
- brak rekurencji, brak iteracji,
- ograniczone możliwości sterowania przetwarzaniem danych,
- brak możliwości dedukcji.

©Antoni Ligęza