Bazy danych i systemy zarządzania

Wykład V

Elementy języka SQL

Część II

Realizacja operacji algebry relacji,
zaawansowane techniki programowania,
przegląd konstrukcji języka,
wybrane problemy

Suma relacji zgodnych – operator `UNION` wewnątrz klauzuli `SELECT`

```
SELECT kolumnaA1, kolumnaA2,..., kolumnaAk
   FROM tabelaA
   WHERE predykatA
UNION
SELECT kolumnaB1, kolumnaB2,..., kolumnaBk
   FROM tabelaB
   WHERE predykatB;
```

Wskazane kolumny muszą mieć identyczne typy i rozmiary (niekoniecznie nazwy) oraz musi być zachowana zgodna kolejność tych kolumn. Zastosowanie operatora `UNION` automatycznie usuwa duplikaty.

```
SELECT Identyfikator, Nazwisko, Imie, Wiek
   FROM Pracownicy
   WHERE Wiek < 28
UNION
SELECT Identyfikator, Nazwisko, Imie, Wiek
   FROM Studenci
   WHERE Wiek < 28;
```

Opcja `CORRESPONDING BY` jest określona dla SQL-92 (gdy wybrane kolumny są zgodne, również co do nazwy; struktura tabel nie musi być identyczna).

```
SELECT *
   FROM Pracownicy
   WHERE Wiek < 28
UNION CORRESPONDING BY (Identyfikator, Nazwisko, Imie, Wiek)
SELECT *
   FROM Studenci
   WHERE Wiek < 28;
```

Kolumny nie wykazane w opcji `CORRESPONDING BY` są pomijane.

Suma relacji zgodnych – operator OR w opcji WHERE klauzuli SELECT

```
SELECT kolumna1, kolumna2,..., kolumnak
   FROM Tabela
   WHERE PredykatA
UNION
SELECT kolumna1, kolumna2,..., kolumnak
   FROM Tabela
   WHERE PredykatB;
```

Dla realizacji sumy elementów tej samej tabeli można wykorzystać spójnik OR:

```
SELECT kolumna1, kolumna2,..., kolumnak
   FROM Tabela
   WHERE PredykatA OR PredykatB;
```

```
SELECT Identyfikator, Nazwisko, Imie, Wiek
   FROM Pracownicy
   WHERE Dzial = 'Kadry'
UNION
SELECT Identyfikator, Nazwisko, Imie, Wiek
   FROM Pracownicy
   WHERE Dzial = 'Place';
```

może być zastąpione przez:

```
SELECT Identyfikator, Nazwisko, Imie, Wiek
   FROM Pracownicy
   WHERE Dzial = 'Kadry' OR Dzial = "Place';
```

Zapytanie z UNION dwukrotnie przeszukuje tabelę; OR nie usuwa duplikatów.

Suma relacji – polecenie `INSERT`

Polecenie `INSERT INTO` może być wykorzystane do dopisania pojedynczego wiersza do tabeli oraz skopiowania jednego lub wielu wierszy z innej tabeli; ma ono ogólną postać:

```
INSERT INTO Tabela
   [(kolumna1, kolumna2,...,kolumnak)]
   VALUES (lista wartosci);
```

lub

```
INSERT INTO TabelaA
   [(kolumnaA1, kolumnaA2,...,kolumnaAk)]
   SELECT kolumnaB1, kolumnaB2,...,kolumnaBk
      FROM TableaB
      WHERE WarunekWyboruWierszy;;
```

Przykładowe zastosowania:

```
INSERT INTO Ksiazki
   VALUES ('83-87102-55-5','Harrington','SQL dla kazdego',
           'EDU-Mikom', 1998, 'Warszawa');

INSERT INTO Ksiazki
   (ISBN,Autor,Tytul)
   VALUES ('83-87102-55-5','Harrington','SQL dla kazdego');

INSERT INTO Ksiazki
   SELECT ISBN, Autor, Tytul, Wydawnictwo, Rok, Miejsce
      FROM ZamowioneKsiazki
      WHERE Status = 'Dostarczone';
```

Uwaga: rekordy generowane poleceniem `SELECT` muszą być zgodne co do struktury (liczba, typ) z tabelą docelową.

Różnica (odejmowanie) tabel/relacji

```
SELECT kolumna1, kolumna2,..., kolumnak
   FROM TabelaA
   WHERE (kol1, kol2,...,kolj) NOT IN (SELECT kol1, kol2,...,kolj
                                          FROM TabelaB
                                          WHERE Warunek);
```

Przykłady:

```
SELECT Autor, Tytul
   FROM Ksiazki
   WHERE ISBN NOT IN (SELECT ISBN
                         FROM ZamowioneKsiazki);
```

```
SELECT NazwaWydawcy
   FROM Wydawcy
   WHERE NazwaWydawcy NOT IN (SELECT NazwaWydawcy
                                 FROM Ksiazki
                                 WHERE ISBN IN (SELECT ISBN
                                                  FROM ZamowioneKsiazki)
      );
```

```
SELECT NazwaKlienta, NrKlienta
   FROM KLienci
   WHERE NrKlienta  NOT IN (SELECT NrKlienta
                                 FROM Zamowienia
                                 WHERE Data >= '2000-01-01');
```

```
SELECT IdFaktury, TypElementu, Odbiorca
 FROM RejestrSprzedazy
 WHERE (IdFaktury, TypElementu NOT IN (SELECT IdFaktury, TypElementu
                                          FROM Zaplacone);
```

Różnica z wykorzystaniem EXCEPT

```
SELECT kolumna1, kolumna2,...,kolumnak
   FROM TabelaA
   EXCEPT
   SELECT kolumna1, kolumna2,...,kolumnak
      FROM TabelaB;
```

```
SELECT *
   FROM TabelaA
   EXCEPT CORRESPONDING BY (kol1, kol2,...,kolk)
   SELECT *
      FROM TabelaB;
```

Przykład:

```
SELECT Autor, Tytul, RokWydania
   FROM Ksiazki
   EXCEPT
   SELECT Autor, Tytul, RokWydania
      FROM KsiazkiDoKasacji;
```

Różnicę można też zrealizować za pomocą złączenia `LEFT OUTER JOIN`.

```
SELECT kolumna1, kolumna2,...,kolumnak
   FROM (TabelaA LEFT OUTER JOIN TabelaB)
   ON TabelaA.Atrybut = TabelaB.Atrybut
   WHERE TabelaB.Atrybut IS NULL;
```

Przykład:

```
SELECT DISTINCT Ksiazki.*
   FROM (Ksiazki LEFT OUTER JOIN KsiazkiDoKasacji)
   ON TabelaA.Atrybut = TabelaB.Atrybut
   WHERE TabelaB.Atrybut IS NULL;
```

Iloczyn zwykły tabel/relacji

```
SELECT kolumna1, kolumna2,...,kolumnak
   FROM TabelaA
   INTERSECT
   SELECT kolumna1, kolumna2,...,kolumnak
   FROM TabelaB;
```

Przykłady:

```
SELECT Autor, Tytul, RokWydania
   FROM ZamowieniaA
   INTERSECT
   SELECT Autor, Tytul, RokWydania
   FROM ZamowieniaB;

SELECT *
   FROM ZamowieniaA
   WHERE RokWydania > '1995'
   INTERSECT CORRESPONDING  BY (Autor, Tytul, RokWydania)
   SELECT *
      FROM ZamowieniaB
      WHERE Cena < 1000;
```

Inne możliwości realizacji iloczynu: $R \cap S = R \setminus (R \setminus S)$ oraz `JOIN`.

```
SELECT Autor, Tytul, RokWydania
   FROM ZamowieniaA JOIN ZamowieniaB ON (Autor, Tytul, RokWydania)
   WHERE ZamowieniaA.RokWydania > '1995' AND
         ZamowieniaB.Cena < 100;
```

Usuwanie i modyfikacja rekordów

Kasowanie rekordów

```
DELETE FROM Tabela
   WHERE Warunek;
```

```
DELETE FROM Ksiazki
   WHERE RokWydania < 1937 AND Cena < 123;
```

```
DELETE FROM Ksiazki
   WHERE Sygnatura IN (SELECT Sygnatura
                           FROM KsiazkiZagubione
                           WHERE StatusStraty = 'pokryta');
```

Modyfikacja rekordów

```
UPDATE   Tabela
   SET kolumna1 = nowa_wartosc,
       kolumna2 = nowa_wartosc,...,
       kolumnak = nowa_wartosc
   WHERE Warunek;
```

```
UPDATE  Czytelnicy
   SET Ulica = 'Krolewska',
       Limit =  15
   WHERE IdCzytelnika = 'PU1010';
```

```
UPDATE  Ksiazki
   SET Cena = Cena * 1.2,
   WHERE RokWydania > '1998';
```

Inne operacje

Projekcja na atrybuty `kolumna1, kolumna2,...,kolumnak`:

```
SELECT kolumna1, kolumna2,...,kolumnak
   FROM Tabela;
```

Selekcja dla warunku `KryteriumSelekcji`:

```
SELECT *
   FROM Tabela
   WHERE KryteriumSelekcji;
```

Iloczyn kartezjański:

```
SELECT TabelaA.*, TabelaB.*
   FROM TabelaA, TabelaB;
```

```
SELECT TabelaA.*, TabelaB.*
   FROM TabelaA CROSS JOIN TabelaB;
```

Różnica lewa (prawa):

```
SELECT TabelaA.*  (TabelaB.*)
   FROM TabelaA UNION JOIN TabelaB
   WHERE TabelaB.AtrybutB IS NULL;
  (WHERE TabelaA.AtrybutA IS NULL;)
```

Dopełnienie dla wybranego atrybutu z innej tabeli:

```
SELECT DISTINCT WybranyAtrybut
     FROM TabelaOdniesienia
     EXCEPT
     SELECT WybranyAtrybut
        FROM TabelaDana;
```

Organizacja sbiektów 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).

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.

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 zerealizowany (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).

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

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.

Definiowanie tabel

Do definiowania tabel służy instrukcja `CREATE TABLE`. Ma ona następukącą postać ogólną:

```
CREATE [{GLOBAL | LOCAL} TEMPORARY]  TABLE nazwa 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 CHARACTER |  BIT STRING |
              EXACT NUMERIC | APPROXIMATE NUMERIC | 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),
   StanKli CHAR (2),
   KodPocztKli CHAR (5),
   TelefonKli CHAR (10)  NOT NULL,
   EmailKli VARCHAR (40),
   PRIMARY KEY (NumerKli));
```

spowoduje utworzenie tabeli opisującej klientów wg podanej specyfikacji.

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.

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.