Bazy danych

Wykład 2_1

Temat: Relacyjny model danych

Sławomir Świętoniowski

slawomir-swietoniowski@wp.pl

Plan wykładu

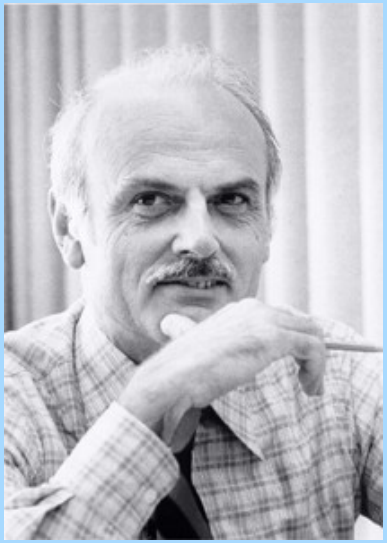
1. **Założenia relacyjnego modelu danych.**
2. Definicja podstawowych pojęć.
3. Algebra relacyjna i język SQL.
4. Reguły Codda dla systemów relacyjnych baz danych.

Model relacyjny - historia

Główny twórca:

Edgar F. Codd

(23 VIII 1923 - 18 IV 2003)



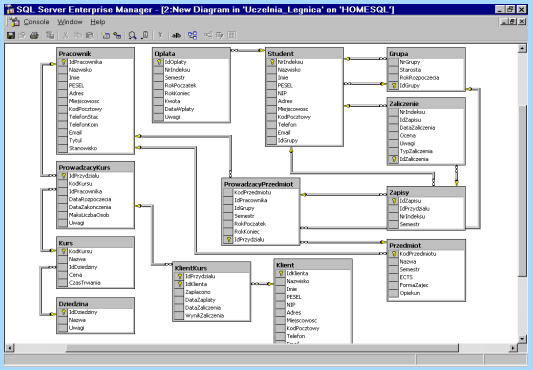
Źródło: Wikipedia (http://en.wikipedia.org).

Ważniejsze publikacje:

- "A Relational Model of Data for Large Shared Data Banks", 1970, CACM 13, No. 6.
- „Extending the Relational Database Model to Capture More Meaning", ACM Transactions on Database Systems, 1979, Vol. 4, No. 4, pp. 397-434.
- „Relational Database: A Practical Foundation for Productivity", Communications of ACM, 1982, Vol. 25, No. 2.
- „The Relational Model for Database Management: Version 2", Reading, Mass., Addison-Wesley, 1990.

Model relacyjny – podstawowe założenia

- Każda tabela w bazie danych ma jednoznaczną nazwę.
- Każda kolumna ma jednoznaczną nazwę w ramach jednej tabeli.
- Wszystkie wartości kolumny muszą być tego samego typu – zdefiniowane na tej samej dziedzinie
- Porządek kolumn w tabeli nie jest istotny.
- Każdy wiersz w tabeli musi być różny - powtórzenia nie są dozwolone.
- Porządek wierszy nie jest istotny.
- Każda wartość pola tabeli (na przecięciu kolumna/wiersz) powinna być atomowa – nie może być ciągiem, ani zbiorem.



IdPracownika	Nazwisko	Imie	NIP	PESEL	Adres	Miejscowosc
1	Nowakowski	Andrzej	612-412-54-64	57121943212	ul. Świerkowa 6	Wrocław
2	Kowalski	Jan	663-654-76-87	72013142337	ul. Strzelecka 15/8	Wrocław
3	Janicki	Bogdan	432-543-654-6	49042343259	ul. Pastelowa 58/9	Wrocław
4	Marcinkowski	Piotr	789-098-23-45	76110309878	ul. Lakiernicza 98A/	Wrocław
5	Andrzejewski	Grzegorz	980-432-23-12	52082898152	ul. Nobla 8/3	Wrocław
6	Piotrowski	Bartlomiej	780-678-66-11	65102189131	ul. Nadrzeczna 16/	Legnica
7	Bogdanska	Ewa	430-543-55-22	79031487924	ul. Wrocławska 23/	Legnica
8	Grzegorzewski	Pawel	250-532-99-90	76061898778	ul. Karkonoska 58	Wrocław
9	Styczen	Tomasz	610-220-96-52	68071678156	ul. Janowska 43/63	Legnica
10	Romanowski	Janusz	616-420-90-19	74090209835	ul. Góralska 43	Legnica

Plan wykładu

1. Założenia relacyjnego modelu danych.
2. **Definicja podstawowych pojęć.**
3. Algebra relacyjna i język SQL.
4. Reguły Codda dla systemów relacyjnych baz danych.

Def. 1 Relacja matematyczna

Niech dane będą zbiory $D_1, D_2, ..., D_n$.

Relacją matematyczną R nad tymi zbiorami nazywamy dowolny podzbiór iloczynu kartezjańskiego nad tymi zbiorami, tzn.

$R \subseteq D_1 \times D_2 \times ... \times D_n = \{(d_1, d_2, ..., d_n): d_i \in D_i, i=1,2, ...,n\}$.

Przykład:

R: > (relacja większości)

$D_1 = \{3, 4\}$

$D_2 = \{1, 2, 3\}$

$D_1 \times D_2 = \{(3, 1), (3, 2), (3, 3), (4, 1), (4, 2), (4, 3)\}$

$R \subseteq D_1 \times D_2 = \{(3, 1), (3, 2), (4, 1), (4, 2), (4, 3)\}$

Na podstawie: PANKOWSKI T., *Podstawy baz danych*, Wydawnictwo Naukowe PWN, Warszawa, 1992.

Def. 2. Wiersz

Zbiór atrybutów (kolumn): $U = \{A_1, A_2, ...\}$. Dla każdego $A \in U$, $DOM(A)$ jest zbiorem wartości **dziedziną** (domeną) atrybutu A.

Wierszem typu U nazywamy dowolną funkcję:

$$f: U \rightarrow \cup \{DOM(A): A \in U\}$$

taką, że dla dowolnego $A \in U$, $f(A) \in DOM(A)$.

Zbiór wszystkich wierszy typu U oznaczamy jako: $WIERSZ(U)$.

Na podstawie: PANKOWSKI T., *Podstawy baz danych*, Wydawnictwo Naukowe PWN, Warszawa, 1992.

Wiersz - przykład

Zbiór atrybutów: $U = \{imię, nazwisko, wiek\}$.

$DOM(imię) = \{Jan, Andrzej\}$

$DOM(nazwisko) = \{Nowak, Kowalski, Jabłoński\}$

$DOM(wiek) = \mathbb{N} \cap [1; 130]$

Przykładowy wiersz typu U:

$r(U) = \{(imię, Andrzej), (nazwisko, Nowak), (wiek, 22)\}$

$WIERSZ(U) = \{$
$\{(imię, Jan), (nazwisko, Nowak), (wiek, 1)\},$
$\{(imię, Jan), (nazwisko, Nowak), (wiek, 2)\},$
$...,$
$\{(imię, Andrzej), (nazwisko, Jabłoński), (wiek, 130)\}\ \}$

Def. 3. Relacja (tabela)

Relacją (ang. *relation*) typu U nazywamy dowolny, skończony podzbiór zbioru *WIERSZ* (U).

Zbiór wszystkich relacji (tabel) typu U oznaczamy: *REL* (U).

Oznaczenia:

- Relacje typu U: $R(U)$, $S(U)$, $T(U)$,... lub R, S, T, ...
- Wiersze typu U: $r(U)$, $s(U)$, $t(U)$,... lub r, s, t, ...
- Podzbiory U: X, Y, Z, ...
- Wiersz $r(U)$: $r\ (U) = \{(A_1, a_1), (A_2, a_2), ..., (A_n, a_n)\}$
  lub w uproszczeniu: $r\ (U) = (a_1, a_2, ..., a_n)$
  np. $r\ (U) = (Andrzej, Nowak, 22)$

Relacja (tabela) - przykład

Tabela: Osoby

Imię	Nazwisko	Wiek
Jan	Kowalski	18
Andrzej	Jabłoński	37
Andrzej	Nowak	25

$$WIERSZ\,(U) = \{$$
$$\{(imię,\ Jan),\ (nazwisko,\ Nowak),\ (wiek,\ 1)\},$$
$$\{(imię,\ Jan),\ (nazwisko,\ Nowak),\ (wiek,\ 2)\},$$
$$\ldots,$$
$$\{(imię,\ Andrzej),\ (nazwisko,\ Jabłoński),\ (wiek,\ 130)\}\ \}$$

Def. 4. Zależność funkcyjna

Niech dana będzie tabela $R(U)$ i niech $X, Y \subseteq U$ będą zbiorami atrybutów. Mówimy, że w R spełniona jest zależność funkcyjna $X \rightarrow Y$, jeśli dla wszystkich wierszy w relacji R wartości atrybutów ze zbioru Y zależą od wartości atrybutów ze zbioru X. Mówimy wówczas, że Y zależy funkcyjnie od X lub, że X determinuje funkcyjnie Y.

Tabela *R*

Nr_indeksu	Nazwisko	Przedmiot	Ocena
1000	Kowalski	Bazy danych	4.5
1000	Kowalski	Akademia CISCO	4.0
1003	Morawski	Bazy danych	5.0
1006	Nowak	Bazy danych	3.0
1006	Nowak	Akademia CISCO	4.5

W tabeli R występują poniższe zależności funkcyjne:

$Nr\_indeksu \rightarrow Nazwisko$ oraz $\{Nr\_indeksu, Przedmiot\} \rightarrow Ocena$

Klucze główne i obce

- **Klucz główny** (ang. *primary key*, PK):
  - zbiór atrybutów, który identyfikuje jednoznacznie wiersze tabeli;
  - w tabeli może być kilka kluczy kandydujących, spośród których wybieramy jeden klucz główny (np. w tabeli [Osoba] kluczami kandydującymi mogą być kolumny [NIP], [PESEL], [IdOsoby]).
- **Klucz obcy** (ang. *foreign key*, FK):
  - pozwala na łączenie danych z różnych tabel;
  - zbiór atrybutów w tabeli, który czerpie swoje wartości z tej samej dziedziny, co klucz główny tabeli powiązanej.

Integralność encji

- Każda tabela musi mieć klucz główny, który jednoznacznie identyfikuje wiersze tej tabeli.
- Klucz główny nie może zawierać wartości pustych (null).
- Zabronione są powtórzenia wierszy w ramach jednej tabeli.

Integralność referencyjna

- Klucz obcy może przyjmować jedną z dwóch wartości:
  - klucz główny z tabeli powiązanej;
  - wartość NULL (jeżeli nie koliduje to z innymi regułami integralności).
- Niedozwolone są wskazania poprzez klucz obcy na wiersz, który nie istnieje.
- Kaskadowa aktualizacja i usuwanie zależą od konkretnego zastosowania (np. jeśli usuwamy fakturę VAT z tabeli [Faktura], to usuwamy także wszystkie jej pozycje z powiązanej tabeli [Pozycja]; natomiast jeśli usuwamy grupę studentów z tabeli [Grupa], to raczej nie usuwamy z bazy wszystkich studentów z tej grupy, zapisanych w powiązanej tabeli [Student]).

Plan wykładu

1. Założenia relacyjnego modelu danych.
2. Definicja podstawowych pojęć.
3. **Algebra relacyjna i język SQL.**
4. Reguły Codda dla systemów relacyjnych baz danych.

Algebra relacyjna

Algebra relacyjna do zbiór operacji na tabelach i wierszach. Większość tych operacji ma bardzo duże znaczenie praktyczne i stanowi podstawę działania komend języka SQL. (Operacje o znaczeniu praktycznym są pogrubione).

- **projekcja**,
- **selekcja**,
- **złączenie**,
- **suma**,
- **różnica**,
- **przekrój**,
- *dopełnienie*,
- *podzielenie*.

Projekcja tabeli

Projekcja tabeli *R* jest to tabela *T*, w której są tylko wybrane kolumny z tabeli *R*.

Tabela *R*

Imie	Nazwisko	Wiek
Jan	Kowalski	18
Andrzej	Jabłoński	37
Andrzej	Nowak	25
Krzysztof	Nowak	25
Leszek	Morawski	59

```
-- Odpowiednik w języku SQL:
SELECT DISTINCT Nazwisko, Wiek
FROM R
```

Projekcja tabeli *R* na kolumny [Nazwisko, Wiek]

Nazwisko	Wiek
Kowalski	18
Jabłoński	37
Nowak	25
Morawski	59

Selekcja tabeli

Selekcja tabeli *R* jest to tabela *T*, w której są tylko wybrane wiersze z tabeli *R*, zgodnie z nałożonym warunkiem selekcji.

Tabela *R*

Imie	Nazwisko	Wiek
Jan	Kowalski	18
Andrzej	Jabłoński	37
Andrzej	Nowak	25
Krzysztof	Nowak	25
Leszek	Morawski	59

Selekcja tabeli *R* - warunek: Wiek > 30.

Imie	Nazwisko	Wiek
Andrzej	Jabłoński	37
Leszek	Morawski	59

```
-- Odpowiednik w języku SQL:
SELECT *
FROM R
WHERE Wiek > 30 -- Warunek selekcji.
```

Złączenie tabel

Złączeniem tabel *R* i *S* jest tabela *T*, w której wiersze mają połączone kolumny z tabel *R* i *S*. W tabeli *T* są tylko te wiersze, w których nastąpiło dopasowanie wartości określonych kolumn z obu tabel.

Tabela *R*

Imie	Nazwisko	KodPrzedmiotu
Jan	Kowalski	INF507
Andrzej	Jabłoński	INF517
Krzysztof	Nowak	INF517
Leszek	Morawski	INF517

N — 1

Tabela *S*

KodPrzedmiotu	Nazwa	ECTS
INF407	Bazy danych	4
INF507	Sieciowe bazy danych	4
INF517	Grafika komputerowa	5

Złączone tabele *R* i *S*

Imie	Nazwisko	KodPrzedmiotu	Nazwa	ECTS
Jan	Kowalski	INF507	Sieciowe bazy danych	4
Andrzej	Jabłoński	INF517	Grafika komputerowa	5
Krzysztof	Nowak	INF517	Grafika komputerowa	5
Leszek	Morawski	INF517	Grafika komputerowa	5

```
-- Odpowiednik w języku SQL:
SELECT Imie, Nazwisko, KodPrzedmiotu, Nazwa, ECTS
FROM R INNER JOIN S ON R.KodPrzedmiotu = S.KodPrzedmiotu
```

Suma tabel

Sumą tabel *R* i *S* jest tabela *T*, w której są połączone wiersze z tabel *R* i *S* (bez powtórzeń).

Tabela *R*

Imie	Nazwisko	Wiek
Jan	Kowalski	18
Andrzej	Jabłoński	37
Andrzej	Nowak	25

Tabela *S*

Imie	Nazwisko	Wiek
Krzysztof	Nowicki	23
Leszek	Morawski	59

Suma tabel *R* i *S*

Imie	Nazwisko	Wiek
Jan	Kowalski	18
Andrzej	Jabłoński	37
Andrzej	Nowak	25
Krzysztof	Nowicki	23
Leszek	Morawski	59

```
-- Odpowiednik w języku SQL:
SELECT *
FROM R
 UNION -- Operator sumy.
SELECT *
FROM S
```

Różnica tabel

Różnicą tabel *R* i *S* jest tabela *T*, w której są wiersze z tabeli R, które nie występują w tabeli *S*.

Tabela *R*

Imie	Nazwisko	Wiek
Jan	Kowalski	18
Andrzej	Jabłoński	37
Andrzej	Nowak	25

Tabela *S*

Imie	Nazwisko	Wiek
Andrzej	Nowak	25
Leszek	Morawski	59

Różnica tabel: *R – S*

Imie	Nazwisko	Wiek
Jan	Kowalski	18
Andrzej	Jabłoński	37

```
-- Odpowiednik w języku SQL:
SELECT *
FROM R
WHERE NOT EXISTS
    (SELECT *
     FROM S
     WHERE R.Imie = S.Imie AND
     R.Nazwisko = S.Nazwisko AND
     R.Wiek = S.Wiek)
```

Przekrój tabel

Przekrojem tabel *R* i *S* jest tabela *T*, której wiersze jednocześnie występują w tabeli R i w tabeli *S*.

Tabela *R*

Imie	Nazwisko	Wiek
Jan	Kowalski	18
Andrzej	Jabłoński	37
Andrzej	Nowak	25

Tabela *S*

Imie	Nazwisko	Wiek
Andrzej	Nowak	25
Leszek	Morawski	59

Przekrój tabel *R i S*

Imie	Nazwisko	Wiek
Andrzej	Nowak	25

```
-- Odpowiednik w języku SQL:
SELECT *
FROM R
WHERE EXISTS
    (SELECT *
     FROM S
     WHERE R.Imie = S.Imie AND
     R.Nazwisko = S.Nazwisko AND
     R.Wiek = S.Wiek)
```

Co to jest język SQL?

SQL – Structured Query Language

(wymowa: `*es-`kju-`el* lub `*si:kuel*):

- Najbardziej rozpowszechniony, ustandaryzowany język baz danych.
- Pozwala na:
  - **definiowanie struktury bazy danych** (pozbiór DDL – Data Definition Language), np. CREATE TABLE, ALTER VIEW, DROP PROCEDURE;
  - **operowanie na danych**: dodawanie, udostępnianie, modyfikowanie i usuwanie (pozdbiór DML – Data Manipulation Language), np. INSERT, SELECT, UPDATE, DELETE.
  - **zarządzanie dostępem do danych** (podzbiór DCL – Data Control Language), np. GRANT, DENY, REVOKE;
  - **definiowanie więzów integralności** (podzbiór DIL – Data Integrity Language), np. CREATE RULE.

Plan wykładu

1. Założenia relacyjnego modelu danych.
2. Definicja podstawowych pojęć.
3. Algebra relacyjna i język SQL.
4. **Reguły Codda dla systemów relacyjnych baz danych.**

Reguły Codda dla relacyjnych systemów DBMS

- W 1985 roku twórca relacyjnych baz danych Edgar F. Codd opublikował zestaw 12 reguł, które powinien spełniać relacyjny system zarządzania bazą danych – DBMS (ang. *Database Management System*).
- Reguły Codda zapewniają zgodność systemu DBMS z modelem relacyjnym, na którym jest on oparty, a przez to wpływają korzystnie na integralność danych w bazie i wydajność ich przetwarzania.
- Współczesne produkty DBMS zazwyczaj spełniają jedynie część tych reguł (w najlepszym przypadku około 10 z 12).
- Niektóre reguły są trudne do realizacji w rzeczywistych systemach baz danych (np. reguła 11 – niezależności od rozproszenia).

Reguła podstawowa

Każdy relacyjny system DBMS powinien być zdolny do zarządzania bazami danych wyłącznie poprzez swoje właściwości relacyjne.

Reguła 1 – Reguła informacyjna (ang. *The Information Rule*)

Wszystkie informacje w relacyjnej bazie danych są reprezentowane jawnie na poziomie logicznym w dokładnie jeden sposób – jako wartości pól w tabelach.

Reguła 2 – Reguła gwarantowanego dostępu (ang. *Guaranteed Access Rule*)

Do każdej danej (wartości atomowej) w relacyjnej bazie danych jest zagwarantowany jednoznaczny dostęp logiczny poprzez podanie kombinacji: nazwy tabeli, wartości klucza głównego i nazwy kolumny.

Reguła 3 – Systematyczna obsługa wartości NULL (ang. *Systematic Treatment of Null Values*)

Wartość pola tabeli może przyjmować wartość NULL (różną od ciągu pustego, ciągu spacji, zera i innej liczby), o ile nie jest to kolumna klucza głównego. Relacyjny DBMS powinien zapewniać systematyczną obsługę wartości NULL, które reprezentują brakujące informacje.

Reguła 4 – Dynamiczny katalog on-line oparty na modelu relacyjnym

(ang. *Dynamic On-Line Catalog Based on the Relational Model*)

Relacyjny DBMS musi zapewniać dostęp do struktury bazy danych w taki sam sposób, jak do zwykłych danych.

Komentarz: Jest to zazwyczaj realizowane poprzez przechowywanie definicji struktury w specjalnych tabelach systemowych (zwanych katalogiem).

Reguła 5 – Reguła ogólnego subjęzyka danych (ang. *Comprehensive Data Sublanguage Rule*)

Relacyjny DBMS musi obsługiwać przynajmniej jeden język o dobrze zdefiniowanej składni jako łańcuch znaków, który umożliwia:

- definiowanie danych,
- manipulowanie danymi,
- definiowanie więzów integralności,
- definiowanie uprawnień dostępu do danych,
- kontrolę transakcji.

Komentarz: Wszystkie komercyjne, relacyjne DBMS używają do tego języka SQL (ang. *Structured Query Language*).

Reguła 6 – Reguła modyfikacji perspektyw (ang. *View Updating Rule*)

Dane mogą być logicznie prezentowane użytkownikowi w różny sposób za pomocą widoków (perspektyw). Każdy widok powinien umożliwiać taki sam poziom dostępu do danych (np. INSERT, UPDATE, DELETE), jak tabele, na których jest on oparty.

Komentarz: W praktyce reguła ta jest trudna do spełnienia i nie jest w pełni realizowana przez żaden współczesny DBMS.

Reguła 7 – Wstawianie, aktualizacja i usuwanie na wysokim poziomie

(ang. *High-level Insert, Update, and Delete*)

Dane otrzymywane w wyniku zapytań do bazy relacyjnej mogą być złożone z wielu wierszy i wielu tabel.

Komentarz: Innymi słowy, operacje wstawiania, modyfikacji i usuwania powinny być dostępne dla każdego zbioru danych, który może być wyszukany, a nie tylko dla pojedynczego wiersza w jednej tabeli.

Reguła 8 – Fizyczna niezależność danych (ang. *Physical Data Independence*)

Programy użytkowe i operacje interfejsu powinny być fizycznie niezależne od jakichkolwiek zmian w sposobie przechowywania lub metodach dostępu (np. rozkład plików na dyskach, struktura indeksów).

Reguła 9 – Logiczna niezależność danych (ang. *Logical Data Independence*)

Sposób widzenia danych przez użytkownika powinien być niezależny od jakichkolwiek zmian w strukturze logicznej bazy danych (np. w strukturze tabel).

Komentarz: Jest to reguła szczególnie trudna do spełnienia, ponieważ w większości baz danych istnieje silny związek pomiędzy prezentowanym widokiem danych, a rzeczywistą strukturą tabel, na których ten widok jest oparty.

Reguła 10 – Niezależność integralności (ang. *Integrity Independence*)

Język relacyjnej bazy danych (np. SQL) powinien umożliwiać definiowanie więzów integralności – ograniczeń zachowujących bazę w stanie spójności – bez konieczności ich definiowania w programach użytkowych.

Komentarz: Ta własność nie jest realizowana we wszystkich produktach. Jednakże jako minimum wszystkie bazy zapewniają z poziomu SQL:

- integralność encji – klucz główny tabeli nie może przyjmować wartości NULL,
- integralność referencyjną – dla każdego klucza obcego musi istnieć odpowiedni klucz główny w tabeli powiązanej.

Reguła 11 – Niezależność od rozproszenia (ang. *Distribution Independence*)

Użytkownik nie powinien dostrzegać rozproszenia bazy danych – ich podziału na więcej, niż jedną lokację (serwer).

Komentarz: W rzeczywistych systemach reguła ta jest trudna do spełnienia (np. z powodu opóźnienia w przesyłaniu danych ze zdalnego serwera).

Reguła 12 – Reguła nieomijania reguł
(ang. *Nonsubversion Rule*)

W relacyjnym DBMS nie powinno być innej metody modyfikacji struktury bazy danych oprócz wbudowanego języka operowania na danych (np. SQL).

Komentarz: W wielu współczesnych systemach są dostępne narzędzia administracyjne, które umożliwiają bezpośrednią manipulację na strukturze bazy i przechowywanych danych (a więc niezgodnie z niniejszą regułą), choć w niektórych przypadkach narzędzia są jedynie graficzną nakładką na wbudowany język SQL.

Literatura

1. BEYNON-DAVIES P., *Systemy baz danych – nowe wydanie*, WNT, Warszawa 2003.
2. MICROSOFT, *Books On-Line* – dokumentacja systemu *MS SQL Server*, Microsoft Corp. 1988 – 2000.
3. PANKOWSKI T., *Podstawy baz danych*, Wydawnictwo Naukowe PWN, Warszawa, 1992.
4. RANKINS R., JENSEN P., BERTUCCI P., *Microsoft SQL Server 2000. Księga eksperta.*, HELION, Gliwice 2003 (książka dostępna w bibliotece WSIZ „Copernicus").

Bazy danych

Wykład 2_1

Dziękuję za uwagę !