Bazy danych

2. Modelowanie danych

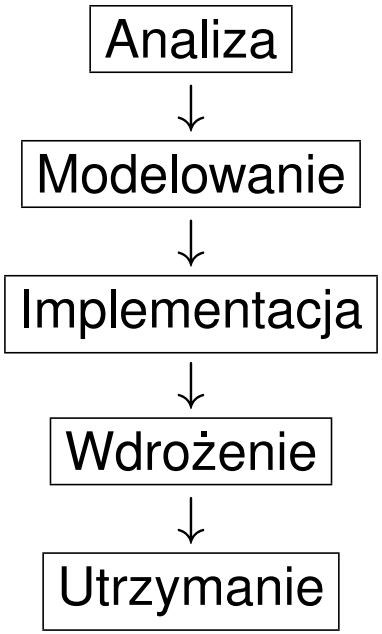
Relacyjny model baz danych

P. F. Góra

http://th-www.if.uj.edu.pl/zfs/gora/

2019/20

Cykl życia projektu informatycznego



Baza danych jest modelem rzeczywistości — ma odpowiadać potrzebom użytkownika, nie wyobrażeniom projektanta

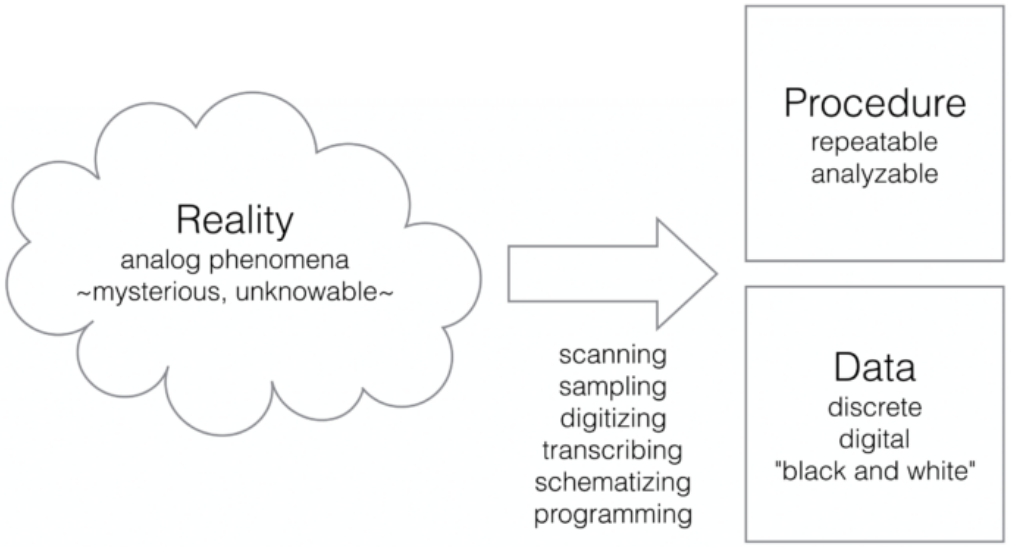
Złamanie tej zasady to typowy (i bolesny) błąd popełniany przy projektowaniu baz danych.

Uzyskanie informacji od klienta czego im *naprawdę* trzeba bywa nie lada wyzwaniem ☹

W Dolinie Krzemowej panuje systemowy brak odpowiedzialności. Zakorzeniony jest w chorobliwej wśród programistów pewności, że rozumieją co jest najlepsze dla świata, podczas gdy w rzeczywistości są naiwni i nie mają nawet pojęcia, jaki jest skomplikowany.

Zeynep Tűfekçi

Programming is Forgetting



Źródło: Allison Parrish, *Programming is Forgetting*

Przykład

```
mysql> CREATE TABLE Studenci
    -> (NrStudenta SMALLINT UNSIGNED NOT NUll AUTO_INCREMENT PRIMARY KEY,
    ->  Imie VARCHAR(20),
    ->  Nazwisko VARCHAR(20) NOT NULL,
    ->  Uwagi VARCHAR(30),
    ->  Grupa CHAR(2) NOT NULL);
Query OK, 0 rows affected (0.06 sec)
```

Projektowanie bazy danych (a w pewnym sensie każdego innego oprogramowania) łączy się z odrzucaniem wielu informacji o świecie zewnętrznym. Część z nich bardzo trudno zdigitalizować. Jakaś część — ta możliwa do digitalizacji lub nie — kiedyś mogłaby się przydać. I co wtedy?

Baza danych jest modelem **jakiegoś fragmentu** rzeczywistości.

Everything should be made as simple as possible, but not simpler.

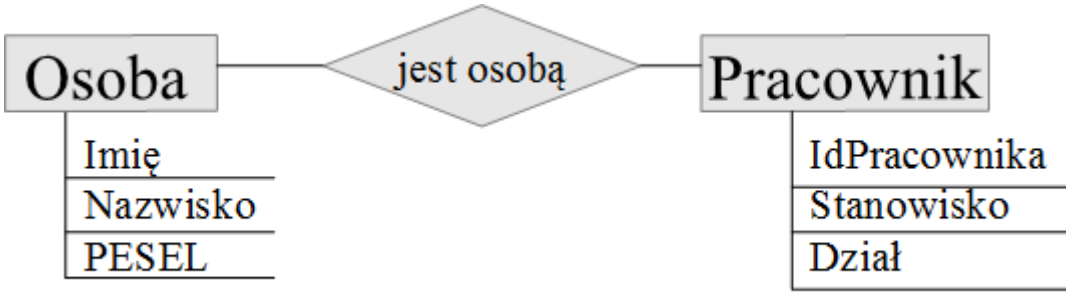
Albert Einstein

Modelowanie danych — diagramy związków encji (ER)

Zbiór encji (entity set) — abstrakcyjna klasa reprezentująca jakiś fragment rzeczywistości: osoba, pracownik, klient, pojazd, książka. Encje mają swoje atrybuty (wszystkie encje z danego zbioru takie same), na przykład imię, nazwisko, PESEL. Zazwyczaj oczekuje się, że istnieje atrybut pozwalający na jednoznaczną identyfikację encji w zbiorze; w podanym przykładzie byłby to PESEL.

Encje mogą wchodzić w **związki** (relationship), łączące elementy jednego zbioru encji z elementami innego (niekiedy tego samego!) zbioru encji. Teoretycznie *związki encji* mogą być wieloargumentowe, a nawet posiadać własne atrybuty. Takie przypadki eliminujemy, wprowadzając “sztuczne” związki dwuargumentowe i/lub dodatkowe zbiory encji (w modelu relacyjnym oznacza to wprowadzanie tabel pomostowych), tak więc ograniczmy się do bezatrybutowych związków binarnych.

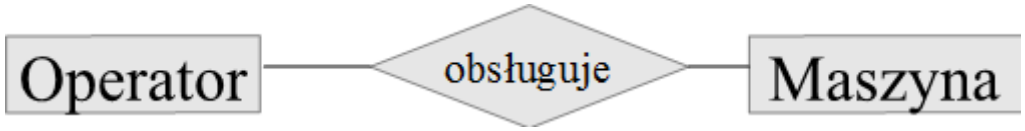
Przykład



Encje, ich atrybuty i związek

Związek: Zbiór par uporządkowanych, podzbiór iloczynu kartezjańskiego zbiorów.

Rodzaje związków między danymi



Związek wiele do wielu — jeden operator może obsługiwać wiele maszyn, jedna maszyna może być obsługiwana przez wielu operatorów.

Niech będą dane zbiory $A = \{x_1, x_2, x_3, x_4\}$, $B = \{y_1, y_2, y_3, y_4, y_5\}$.

Wiele do wielu

A	B
x_1	y_2
x_1	y_3
x_2	y_1
x_2	y_2
x_2	y_5
x_3	y_1
x_4	y_2
x_4	y_5
x_3	y_6

Element y_4 nigdzie się nie pojawia (nie wszystkie elementy zbiorów muszą wchodzić we wskazany związek).

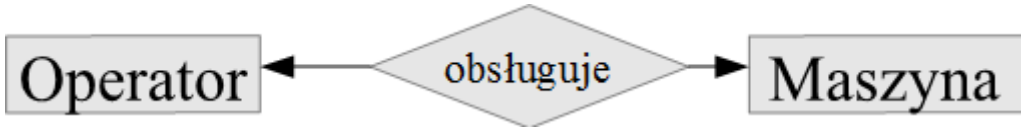
Ostatni wiersz wskazuje na nieporawną, nie należącą do iloczynu kartezjańskiego parę (x_3, y_6). W bazach danych, jeśli *explicite* tego nie zabronimy, tego typu związki można definiować (a co się stanie, gdy taki faktycznie wystąpi, zależy od kontekstu).

Związki wiele do jednego, jeden do wielu

Jedna maszyna może być obsługiwana tylko przez jednego operatora, ale jeden operator może obsługiwać wiele maszyn

Jeden operator może obsługiwać tylko jedną maszynę, ale jedna maszyna może być obsługiwana przez wielu operatorów

Związek jeden do jednego



Jedna maszyna może być obsługiwana tylko przez jednego operatora, a jeden operator może obsługiwać tylko jedną maszynę. Z punktu widzenia informacji można utożsamić operatora z obsługiwaną przez niego maszyną.

Jeden do jednego

A	B
x_1	y_3
x_2	y_1
x_4	y_2

Elementy w żadnej kolumnie nie powtarzają się. Nie wszystkie elementy muszą wystąpić. Ale gdyby element x_3 wystąpił, musiałby się łączyć z y_4 albo y_5.

Integralność referencyjna

Integralność referencyjna (referential integrity) to szczególny, występujący wyłącznie w kontekstach bazodanowych, typ związku wiele-do-jednego, w którym wskazywany element **musi istnieć**. Uniemożliwia to wystąpienie takiej sytuacji, jaka była omawiana na stronie 12. Ze względu na swój ograniczający charakter, ten związek zazwyczaj nazywany jest *więzem integralności referencyjnej*.

Przykład

W bazie danych przychodni medycznej zapisywane są informacje o badaniach zleconych przez zatrudnionych tam lekarzy. Związek integralności referencyjnej

mówi, że w bazie nie da się zapisać informacji o badaniu zleconym przez nie-istniejącego (nie zatrudnionego tam) lekarza. (Oczywiście jeden lekarz może zlecić wiele badań — jest to szczególny rodzaj związku typu wiele do jednego.)

Relacyjny model baz danych

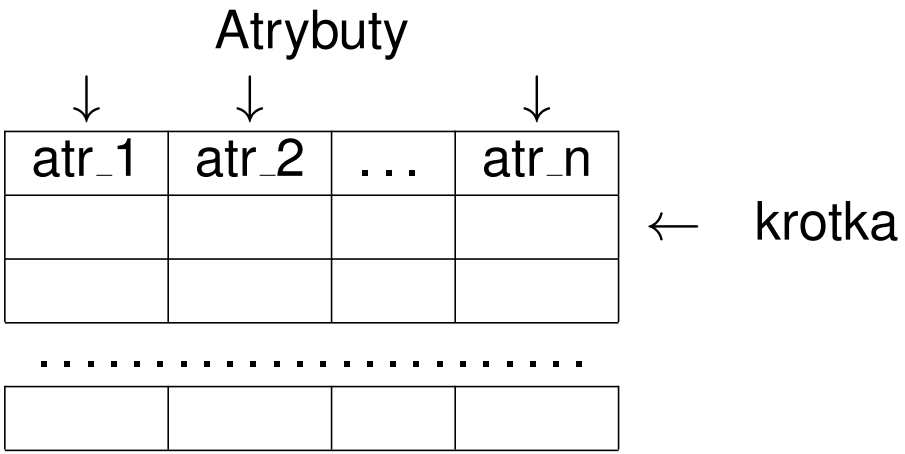
Relacyjny model baz danych jest *stary*, ale wciąż zajmuje $\sim 80\%$ rynku. Nic nie wskazuje na to, aby w najbliższym czasie miało się to zmienić.

Relacyjny model baz danych powstał przy założeniu, że cała baza przechowywana jest na jednym, dostatecznie dużym serwerze.

Obecnie założenie to w praktyce jest bardzo często łamane, ale “w teorii” nadal obowiązuje.

Tabela

W relacyjnym modelu baz danych występuje tylko jedna struktura służąca do przechowywania danych: **tabela** (ang. *relation* — od tego pochodzi nazwa, nie od związków, *relationship*, pomiędzy tabelami).



Każda kolumna zawiera informację o jednym *atrybucie*. Kolumny są uporządkowane. Każdy wiersz, zwany *krotką*, odpowiada jednemu "rekordowi" danych.

Tabela (c.d.)

- Tabela jest **zbiorem** krotek.
  - Zbiór: krotki nie mogą się powtarzać.
  - Zbiór: kolejność krotek jest obojętna.
- Krotki są uporządkowane (to znaczy kolejność atrybutów nie jest obojętna).

Jak zobaczymy, w relacyjnych bazach danych wielką rolę odgrywają **klucze** i **klucze obce**.

Dane a metadane

Tabela (realcja) to obiekt abstrakcyjny. Ma swoje atrybuty i więzy. Zbiór wszystkich takich “projektów” tabel nazywa się *schematem bazy danych*. Schemat wraz z informacjami o użytkownikach i ich uprawnieniach stanowi *metadane* (dane o danych). Schemat tabeli w zasadzie —— w czasie normalnego użytkowania —— nie zmienia się w czasie.

Zbiór wszystkich krotek danej tabeli (“zawartość tabeli”) może się zmieniać w czasie. Zbiór taki istniejący w pewnej chwili czasu nazywa się *instancją* tabeli (relacji). Instancję istniejącą teraz nazywa się instancją bieżącą.

Dwanaście zasad Codda dla RDBMS*

Edgar Codd, IBM, 1970

RDBMS — Relational DataBase Management System

1. Informacje są reprezentowane logicznie w tabelach.
2. Dane są logicznie dostępne przez podanie nazwy tabeli, wartości klucza i nazwy kolumny.
3. Wartości `null` są traktowane w jednolity sposób jako “brakujące informacji”. Nie mogą być traktowane jako puste łańcuchy czy zera.
4. Metadane są umieszczone w bazie danych tak, jak zwykłe dane.
5. Język obsługi danych ma możliwość definiowania danych i widoków, więzów integralności, przeprowadzania autoryzacji, obsługi transakcji i manipulacji danymi.

6. Widoki reagują na zmiany swoich tabel bazowych. Zmiana w widoku powoduje zmianę w tabeli bazowej.
7. Istnieją pojedyncze operacje pozwalające na wyszukanie, wstawienie, uaktualnienie i usunięcie danych.
8. Operacje użytkownika są logicznie oddzielone od fizycznych danych i metod dostępu.
9. Operacje użytkownika pozwalają na zmianę schematu bazy danych bez konieczności tworzenia bazy od nowa.
10. Więzy integralności są umieszczone w metadanych, nie w zewnętrznej aplikacji.
11. Język manipulacji danymi powinien działać bez względu na to jak i gdzie są rozmieszczone fizyczne dane oraz nie powinien wymagać zmian, gdy fizyczne dane są centralizowane lub rozpraszane

12. Operacje na pojedynczych rekordach przeprowadzane w systemie podlegają tym samym zasadom i więzom, co operacje na zbiorach danych.

Algebra relacji

- Działania teoriomnogościowe
  - Suma mnogościowa (unia) $\cup$
  - Przecięcie (iloczyn mnogościowy) $\cap$
  - Różnica mnogościowa $-$
  - Iloczyn kartezjański $\times$
- Selekcja σ
- Rzutowanie (projekcja) π
- Przemianowanie ρ
- Złączenie $\bowtie$

Suma, iloczyn i różnica mnogościowa

Niech będą dane tabele R i S. Tabele te muszą mieć takie same schematy — takie same atrybuty, a także taką samą kolejność atrybutów.

- Suma mnogościowa, $R \cup S$ — zbiór krotek, które należą do R lub S (lub do obu jednocześnie).
- Iloczyn mnogościowy, $R \cap S$ — zbiór krotek, które należą jednocześnie do R i S.
- Różnica mnogościowa, $R - S$ — zbiór krotek, które należą do R i nie należą do S.

Iloczyn kartezjański $R \times S$ — zbiór wszystkich par krotek, w których pierwszy element pary należy do R, drugi do S. Schemat tabeli wyjściowej ma po jednym atrybucie (kolmnie) na każdy atrybut R i po jednym atrybucie na każdy atrybut S. Nazwy atrybutów są, *o ile to możliwe*, dziedziczone.

Uwaga na wielozbiory!

Tabele (relacje) w modelu relacyjnym powinny być zbiorami (krotki nie mogą się powtarzać), ale niekiedy nie są —— jeśli dopuszczamy powtórzenia krotek, czyli zbiory zastępujemy *wielozbiorami*, zmieniają się definicje operacji mnogościowych.

Suma $R \cup S$ —— krotka w wyniku występuje tyle razy, ile występuje w R plus tyle razy, ile występuje w S. Uwaga: jeśli nawet R i S są zbiorami, $R \cup S$ może być wielozbiorem!

Iloczyn $R \cap S$ —— krotka w wyniku występuje tyle razy, ile wynosi minimum jej wystąpień w R i S.

Różnica $R - S$ —— krotka w wyniku występuje tyle razy, ile występuje ona w R minus tyle razy, ile występuje ona w S, ale nie mniej niż 0 razy.

Przykład

$$R = \{A, B, B\}, \quad S = \{A, B, C, C\}$$
$$R \cup S = \{A, A, B, B, B, C, C\}$$
$$R \cap S = \{A, B\}$$
$$R - S = \{B\}$$

Selekcja

$\sigma_C(R)$ — wybierz z tabeli R tylko te krotki, które spełniają warunek wyboru C. W warunku wyboru mogą pojawiać się operatory logiczne. Schemat tabeli wyjściowej jest taki sam, jak tabeli wejściowej.

Rzutowanie

$\pi_{A_1,A_2,\ldots}(R)$ — z tabeli R wybierz tylko kolumny obecne na liście rzutowania $A_1, A_2, \ldots$. Schemat tabeli wyjściowej zawiera tylko kolumny obecne na liście rzutowania.

W formalizmie matematycznym wynik rzutowania jest zbiorem (krotki nie mogą się powtarzać), ale niekiedy — a tak naprawdę dość często — w praktyce może okazać się wielozbiorem. W SQL odpowida to różnicy pomiędzy `SELECT DISTINCT` (zbiór) a `SELECT` (potencjalnie wielozbiór).

Przemianowanie

Niech tabela R ma atrybuty $A_1, A_2, \ldots, A_n$. $\rho_{S(B_1,B_2,\ldots,B_n)}(R)$ (to samo n!) to tabela S, która ma tyle samo atrybutów, co R, ale ich nazwami są $B_1, B_2, \ldots, B_n$. Kolejność atrybutów się nie zmienia. Zawartość krotek się nie zmienia.

Skrótowy zapis $\rho_S(R)$ oznacza tylko zmianę nazwy tabeli z R na S, bez zmiany nazw atrybutów.

Wynikiem działania któregokolwiek z powyższych operatorów na tabelę (lub na parę tabel w przypadku operacji teoriomnogościowych) jest tabela, która sama może stać się obiektem działania któregoś z operatorów algebry relacji. Widać, że można tworzyć operatory złożone, a algebra relacji jest domknięta.

Przykład

Niech tabela R ma atrybuty A, B, C. Napis

$$\rho_S \left(\pi_{A,B} (\sigma_{B>C \wedge C<4}(R)) \right)$$

oznacza, że najpierw z tabeli R wybieramy krotki spełniające warunek "wartość atrybutu B jest większa od wartości atrybutu C" oraz "wartość atrybutu C jest mniejsza, niż 4", a następnie z tak utworzonej tabeli wybieramy tylko kolumny odpowiadające atrybutom A, B. To, co otrzymaliśmy, nazywamy tabelą S.

Złączenie

$$R \bowtie_{C(R,S)} S$$

Złączenie to *podzbiór iloczynu kartezjańskiego* $R \times S$, o tej własności, iż obejmuje on wszystkie i tylko te krotki, które spełniają warunek złączenia $C(R, S)$. Warunek złączenia obejmuje atrybuty obu złączanych tabel. Formalnie

$$R \bowtie_{C(R,S)} S \equiv \sigma_{C(R,S)}(R \times S)$$

Złączenie nie jest operacją elementarną, ale pojawia się tak często, że zazwyczaj jest oddzielnie implementowane i zasługuje na osobne oznaczenie.

Złączenie równościowe (ang. *equijoin*) to takie złączenie, w którym a) złączane tabele mają takie same (powtarzające się) kolumny (atrybuty) — na przykład tabela `Ludzie` i tabela `Pracownicy` mają kolumnę `PESEL` — oraz b) warunek złączenia obejmuje *tylko* równość powtarzających się atrybutów.

Złączenie naturalne to złączenie równościowe po wszystkich powtarzających się kolumnach. Jeżeli dwie tabele mają tylko jedną "wspólną" kolumnę, złączenie równościowe jest złączeniem naturalnym. Jeśli "wspólnych" kolumn jest więcej, do naturalności potrzeba równości wszystkich powtarzających się atrybutów.

Przykład

Dane są dwie tabele:

Tabela R

α	β	γ
A	1	a
A	2	b
B	1	c
C	1	d
C	3	e
D	5	f

Tabela S

α	β	δ	ϵ
A	1	9	p
A	4	8	q
B	2	7	r
C	1	6	s
C	2	5	t
D	5	4	u
D	1	3	v

Dokonamy na nich teraz kilku złączeń.

$$R \bowtie_{R.\beta < S.\beta} S$$

$R.\alpha$	$R.\beta$	γ	$S.\alpha$	$S.\beta$	δ	ϵ
A	1	a	A	4	8	q
A	2	b	A	4	8	q
B	1	c	A	4	8	q
C	1	d	A	4	8	q
C	3	e	A	4	8	q
A	1	a	B	2	7	r
B	1	c	B	2	7	r
C	1	d	B	2	7	r
A	1	a	C	2	5	t
B	1	c	C	2	5	t
C	1	d	C	2	5	t
A	1	a	D	5	4	u
A	2	b	D	5	4	u
B	1	c	D	5	4	u
C	1	d	D	5	4	u
C	3	e	D	5	4	u

Zwróćmy uwagę na notację: Kolumny (atrybuty) o unikalnych nazwach zachowują je, powtarzające się nazwy kolumn są prefiksowane nazwą tabeli.

Przedstawione złączenie **nie** jest złączeniem równościowym.

$$R \bowtie_{R.\beta=S.\beta} S$$

$R.\alpha$	$R.\beta$	γ	$S.\alpha$	$S.\beta$	δ	ϵ
A	1	a	A	1	9	p
B	1	c	A	1	9	p
C	1	d	A	1	9	p
A	2	b	B	2	7	r
A	1	a	C	1	6	s
B	1	c	C	1	6	s
C	1	d	C	1	6	s
A	2	b	C	2	5	t
A	1	a	D	1	3	v
B	1	c	D	1	3	v
C	1	d	D	1	3	v
D	5	f	D	5	4	u

Jest to złączenie równościowe, ale nie jest to złączenie naturalne.

$$R \bowtie_{(R.\alpha = S.\alpha \wedge R.\beta = S.\beta)} S$$

$$R \bowtie S$$

α	β	γ	δ	ϵ
A	1	a	9	p
C	1	d	6	s
D	5	f	4	u

To jest złączenie naturalne: złączenie równościowe po wszystkich powtarzających się atrybutach, w tym wypadku α, β. W wypadku złączenia naturalnego konwencja nakazuje wypisywać powtarzające się atrybuty tylko raz, nie zaś osobno jako kolumny dziedziczone po poszczególnych tabelach. Złączenia naturalne można oznaczać samym symbolem $\bowtie$, bez jawnego podawania warunku złączenia.

Filtry

Rozważmy złączenie

$$R \bowtie_{(\mathcal{P}(R) \wedge \mathcal{Q}(S) \wedge \mathcal{R}(R,S))} S$$

Predykaty $\mathcal{P}$ i $\mathcal{Q}$, działające tylko na kolumnach tabel, odpowiednio, R i S, są *filtrami*, wybierają bowiem pewne podzbiory wierszy tych tabel. Predykat $\mathcal{R}$, działający na kolumnach obu tabel, jest faktycznym warunkiem złączenia (złączenie theta). Im *mniejszy* procent wierszy wybiera z tabeli filtr, tym *większą* ma on selektywność. Dla efektywności złączenia korzystne jest używanie filtru o największej selektywności możliwie najwcześniej.

Przedstawione wyżej złączenie można by także zapisać w postaci

$$(\sigma_{\mathcal{P}}(R)) \bowtie_{\mathcal{R}(R,S)} (\sigma_{\mathcal{Q}}(S))$$