Systemy Baz Danych

Wprowadzenie

Bartosz Zieliński

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Bazy Danych

Baza danych to zbiór danych zapisanych zgodnie z określonymi regułami [Wikipedia]

Do zarządzania bazami danych służą
systemy zarządzania bazami danych
(ang. *Database Management Systems, DBMS*)

Przykład Zbioru Danych

Dane często prezentujemy w formie zbioru rekordów (tablicy), np:

Author's Name	Birth	Title	Year	Price	Genre
C.S. Lewis	1898	*Prince Caspian*	1951	10$	Fantasy
C.S. Lewis	1898	*The Last Battle*	1956	10$	Fantasy
C.S. Lewis	1898	*Perelandra*	1943	12$	Science Fiction
C.S. Lewis	1898	*The Screwtape Letters*	1942	8$	Satire, Fantasy
W.H. Hodgson	1877	*The House on the Borderland*	1908	5$	Horror
W.H. Hodgson	1877	*The Night Land*	1912	8$	Horror, Fantasy
H.P. Lovecraft	1890	*At the Mountains of Madness*	1931	5$	Horror
H.P. Lovecraft	1890	*The Shadow over Innsmouth*	1931	5$	Horror
H.P. Lovecraft	1890	*The Call of Cthulhu*	1928	2$	Horror
H.P. Lovecraft	1890	*Herbert West–Reanimator*	1922	5$	Horror, Comedy
...	...	...	...	...	...

Problemy Baz Danych

- Jakich typów są dane?
- Jak reprezentować na dysku dane różnych typów?
- Jaką nadać strukturę danym?
- Jak zapewnić wielu użytkownikom dostęp do tych samych danych tak by nie przeszkadzali sobie nawzajem?
- Jak kontrolować dostęp do danych?
- Jak zapewnić możliwość wykonywania złożonych operacji na danych (w tym zapytań)?
- Jak zapewnić trwałość i spójność danych?
- ...

Rolą systemów zarządzania bazami danych jest rozwiązywanie (lub ułatwianie rozwiązywania) powyższych problemów.

Jakich Typów są Dane

DBMS dostarcza standardowego zestawu typów. Rolą użytkownika jest przypisanie typów do poszczególnych elementów danych.

- Imię autora to niewątpliwie napis (łańcuch znaków).
- A rok urodzenia? Liczba całkowita, napis czy coś innego?
- Numer telefonu to liczba czy napis? A PESEL?
- A cena? Liczba stało- czy zmienno-przecinkowa?

Problem Reprezentacji Danych Różnych Typów

Dane numeryczne

- Reprezentacja maszynowa — zwarta i ułatwia wykonywanie operacji, ale jest nieprzenośna.
- Reprezentacja tekstowa — zajmuje więcej miejsca, wymaga konwersji w celu wykonania operacji ale jest przenośna.

Dane tekstowe

- Reprezentacja o stałej długości, np. `C.S._Lewis________` (marnuje miejsce, ułatwia skok do następnego pola rekordu).
- Reprezentacja o zmiennej długości, np. `C.S._Lewis\0`.

Problem Możliwej Redundancji Danych

Te same informacje o autorach powtarzają się w wielu rekordach:

Author's Name	Birth	Title	Year	Price	Genre
C.S. Lewis	1898	*Prince Caspian*	1951	10$	Fantasy
C.S. Lewis	1898	*The Last Battle*	1956	10$	Fantasy
C.S. Lewis	1898	*Perelandra*	1943	12$	Science Fiction
C.S. Lewis	1898	*The Screwtape Letters*	1942	8$	Satire, Fantasy
W.H. Hodgson	1877	*The House on the Borderland*	1908	5$	Horror
W.H. Hodgson	1877	*The Night Land*	1912	8$	Horror, Fantasy
H.P. Lovecraft	1890	*At the Mountains of Madness*	1931	5$	Horror
H.P. Lovecraft	1890	*The Shadow over Innsmouth*	1931	5$	Horror
H.P. Lovecraft	1890	*The Call of Cthulhu*	1928	2$	Horror
H.P. Lovecraft	1890	*Herbert West–Reanimator*	1922	5$	Horror, Comedy
...	...	...	...	...	...

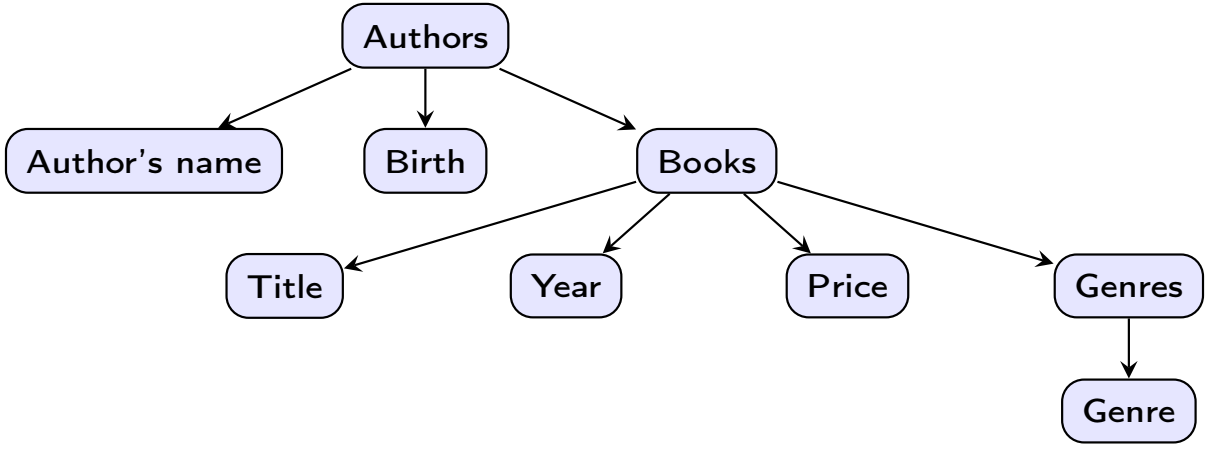
Marnowane jest miejsce na dysku. A co jeśli chcemy zmodyfikować datę urodzenia jednego z autorów? Jak utrzymać spójność danych?

Przechowywanie Danych w Postaci Hierarchicznej

W tym wypadku można uniknąć redundancji przechowując dane w postaci hierarchicznej:

		Books			
					Genres
Author's Name	**Birth**	**Title**	**Year**	**Price**	**Genre**
C.S. Lewis	1898	*Prince Caspian*	1951	10$	Fantasy
		The Last Battle	1956	10$	Fantasy
		Perelandra	1943	12$	Science Fiction
		The Screwtape Letters	1942	8$	Satire Fantasy
W.H. Hodgson	1877	*The House on the Borderland*	1908	5$	Horror
		The Night Land	1912	8$	Horror Fantasy
H.P. Lovecraft	1890	*At the Mountains of Madness*	1931	5$	Horror
		The Shadow over Innsmouth	1931	5$	Horror
		The Call of Cthulhu	1928	2$	Horror
		Herbert West Reanimator	1922	5$	Horror Comedy
...	...	...	...	...	...

Struktura Hierarchiczna Danych

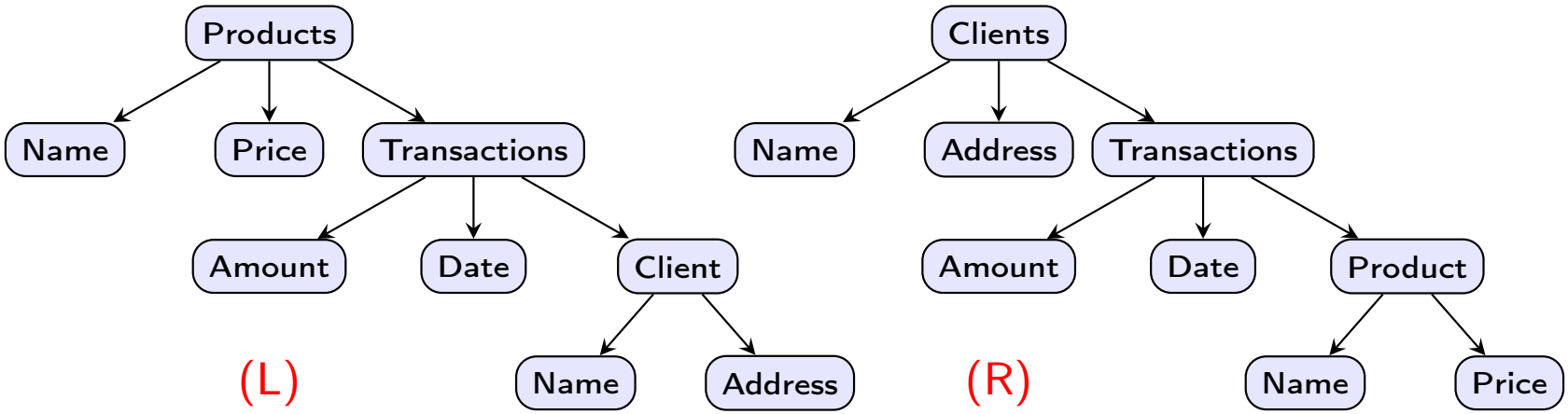


Hierarchiczny Model Danych

- Ciągle używany (rejestr Windows, usługi katalogowe)
- Działa dobrze gdy dla danych istnieje pojedyńcza, naturalna hierarchia. Co jednak zrobić gdy możliwych naturalnych hierarchii jest kilka (niekompatybilnych) albo żadna?
  - Czy fotografie układamy najpierw latami a potem tematami czy na odwrót?

Przykład: Baza Hierarchiczna „Produkty-Transakcje-Klienci”

Możemy albo uznać klientów za część składową produktów które kupili (L) albo uznać produkty za część składową klientów (R):



	(L)	(R)
Zapisz dane o produktach których jeszcze nikt nie kupił	Można	Nie można
Zapisz dane klientów którzy jeszcze nic nie kupili	Nie można	Można
Wyszukaj produkty kupione przez danego klienta	Trudno	Łatwo
Wyszukaj klientów którzy kupili dany produkt	Łatwo	Trudno
Redundantne kopie danych	Klientów	Produktów

Zobacz model sieciowy dla tego przykładu.

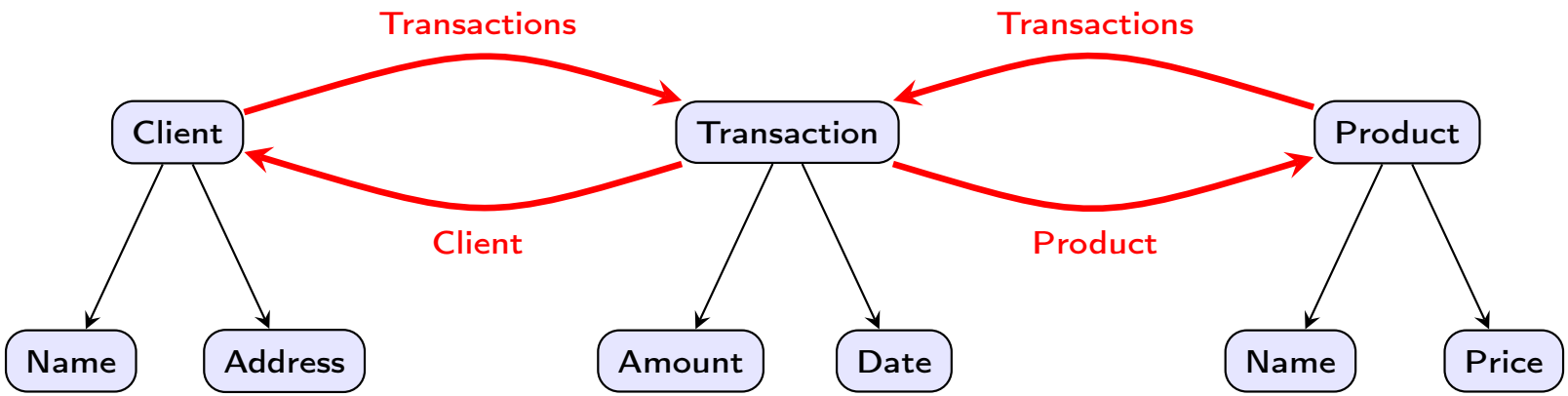
Model Sieciowy Danych

Sieciowy model danych pozwala na nawigację pomiędzy elementami danych po zdefiniowanych przez projektanta bazy ścieżkach które, inaczej niż w modelu hierarchicznym, nie muszą tworzyć drzewa.

- Model sieciowy danych (patrz: CODASYL) ma wiele wspólnego ze współczesnymi modelami grafowym i obiektowym.
- Co zrobić jeśli chcemy skojarzyć elementy danych po ścieżkach których nie przewidział projektant?

Przykład: Baza Sieciowa „Produkty-Transakcje-Klienci”

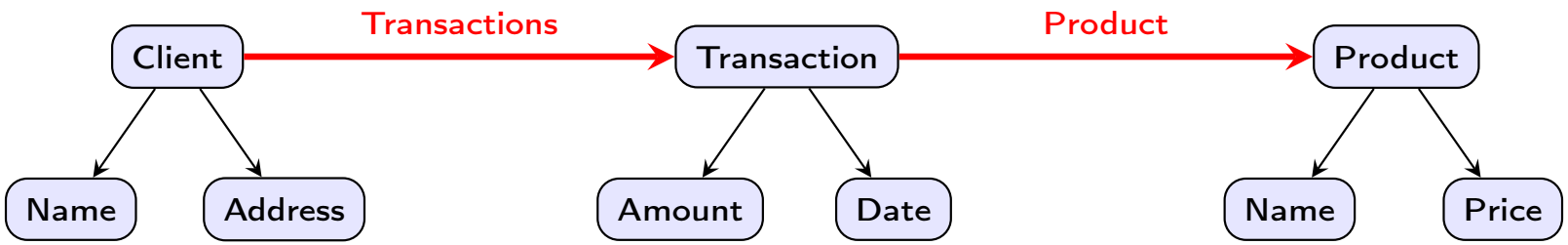
Mamy trzy „klasy obiektów” połączonych ze sobą „referencjami”:



- Każda ze strzałek z etykietą “Transactions” oznacza nie pojedyńczą referencję ale zbiór referencji (od, odpowiednio, klienta i produktu do każdej transakcji w której ten klient i produkt brali udział).
- Powyższy schemat nie ma wad schematów hierarchicznych. Można jedynie zauważyć redundancję referencji z uwagi na ich jednokierunkowość.

Za Mało Referencji

W modelu sieciowym danych już na etapie projektowania bazy należy znać wszystkie możliwe zapytania jakie będą z tą bazą używane ponieważ możliwość zadania pewnych zapytań zależy od obecności w bazie odpowiednich referencji.



Do bazy danych o strukturze przedstawionej powyżej możemy zadać zapytanie o produkty kupione przez danego klienta ale nie o klientów którzy kupili dany produkt.

Model Relacyjny

- **Model relacyjny** to najbardziej rozpowszechniony model danych używany współcześnie w DBMS-ach.
- Całkowicie wyparł w zastosowaniach ogólnobiznesowych modele hierarchiczny i sieciowy, dominujące w latach 70-tych i na początku 80-tych (ponieważ nie miał ich wad).
- Omówienie modelu relacyjnego i relacyjnych baz danych zajmie większą część wykładu i prawie całość ćwiczeń.

Inne Modele

Istnieje wiele innych modeli danych. Są one jednak używane głównie w zastosowaniach specjalnych i żaden z nich nie dorównuje w popularności ani powszechności zastosowań modelowi relacyjnemu:

- Bazy obiektowe
- Bazy grafowe (np. w analizie sieci społecznościowych)
- Bazy XML
- Bazy hierarchiczne (Windows register, DNS, usługi katalogowe)
- Bazy par klucz wartość (Riak, Redis, Memcached)
- Bazy wielowymiarowe (w analizie danych).

Kontrola Współbieżności

Przypuśćmy że dwóch użytkowników modyfikuje jednocześnie pewien rekord. Jeden z nich zapisuje **“Horror, Fantasy”**, drugi **“Fantasy, Horror”**. Bez kontroli współbieżności ostatecznie w rekordzie mogło zostać zapisane **“Fontory Forrasy”**

- Wynik jednoczesnego wykonania złożonych operacji może zależeć od ich przeplecenia.
- Zwykle akceptowalne są wyniki wykonania złożonych operacji jedna po drugiej (bez nietrywialnego przeplotu).
- **Serializowalność** złożonych operacji jest wymuszana przez DBMS-y przez zakładanie odpowiednich blokad na czytane i modyfikowane elementy danych.
- Użytkownik wymusza traktowanie grupy operacji jako pojedyńczej złożonej operacji wykonując te operacje w ramach jednej **transakcji**.

Inny Przykład Kłopotów ze Współbieżnością

Następujące instrukcje wykonywane są jednocześnie przez użytkowników A i B (n jest zmienną prywatną dla każdego z użytkowników):

```
n:=read(f);
n++;
write(n,f);
```

Dwa różne przeplecenia poleceń A i B:

```
//f zawiera 0
n:=read(f);  //A
n++;         //A
write(n,f);  //A
n:=read(f);  //B
n++;         //B
write(n,f);  //B
//f zawiera 2
```

```
//f zawiera 0
n:=read(f);  //A
n:=read(f);  //B
n++;         //A
n++;         //B
write(n,f);  //A
write(n,f);  //B
//f zawiera 1
```

Transakcje

Transakcje spełniają także inną rolę poza kontrolą współbieżności.

Rozważmy przelew bankowy 100$ z konta A do konta B. Składa się on z dwóch operacji: **dodania 100 do konta** B i **odjęcia 100 od konta** A. Przypuśćmy że po **dodaniu 100 do** B doszło do awarii.

- Grupa poleceń umieszczonych wewnątrz transakcji wykonuje się jako jedna całość albo wcale.
- Nieukończoną transakcję można wycofać (cofnąć skutki wykonanych w ramach tej transakcji poleceń).
- Skutki nieukończonej transakcji przerwanej przez awarię zostają automatycznie wycofane przez DBMS podczas jego ponownego uruchomienia.

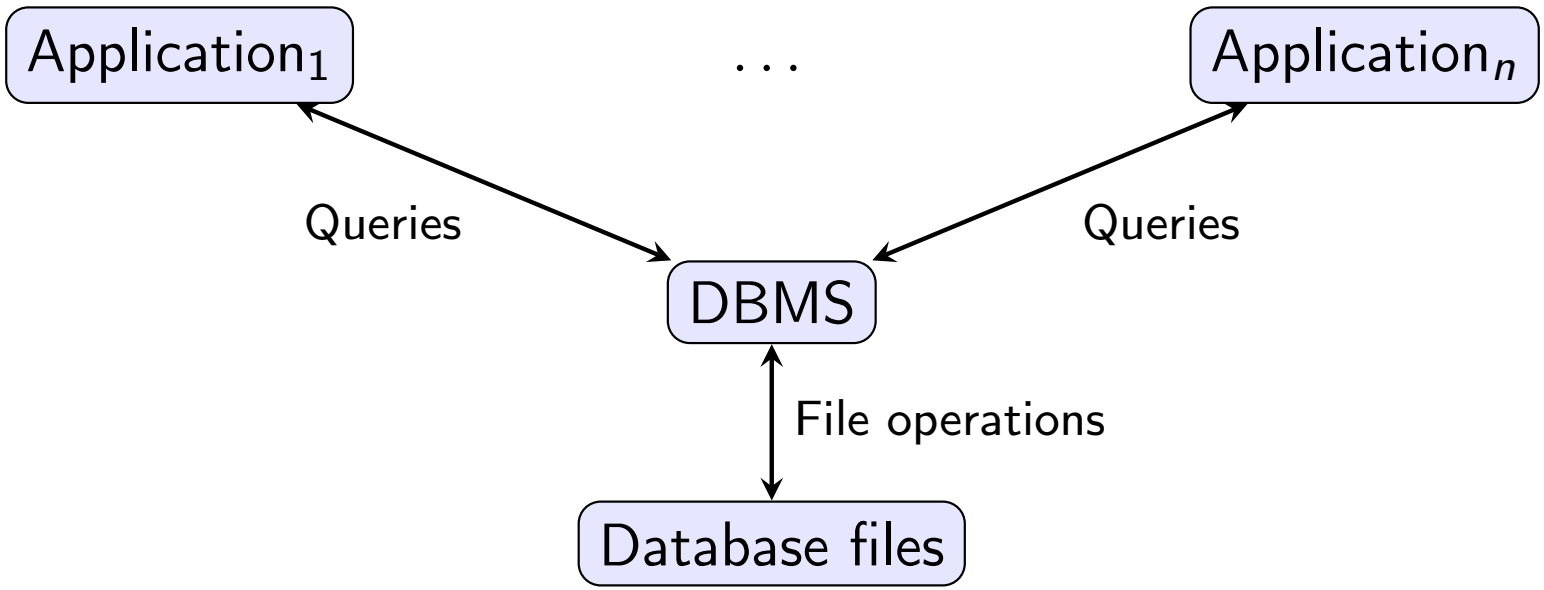
Kontrola Dostępu

Gdy z danych korzysta wielu użytkowników potrzebne jest często ograniczenie uprawnień użytkowników do oglądania lub modyfikowania danych:

- Pracownik może oglądać dane o swoich zarobkach ale nie o zarobkach innych pracowników.
- Pracownik nie może modyfikować swojej pensji.

- Uprawnienia można kontrolować na poziomie aplikacji
- Lepszym rozwiązaniem jest kontrola dostępu na poziomie DBMS. Pozwala to np. na wspólną politykę dla aplikacji dzielących dane.

Architektura Aplikacji Bazodanowych



Aplikacje wykonują operacje na danych wysyłając odpowiednie polecenia do DBMS, nigdy nie wykonują bezpośrednio operacji na plikach danych (do których i tak zwykle nie mają dostępu).

Języki Zapytań

DBMS korzystają ze specjalnych języków zapytań. W językach tych można opisywać wymagane modyfikacje danych, dane, oraz jakie informacje chcemy wyciągnąć z bazy.

Najbardziej rozpowszechnionym językiem zapytań w relacyjnych bazach danych jest **SQL** (*Structured Query Language*).

Praktycznie wszystkie relacyjne bazy danych rozpoznają (jakiś dialekt) SQL

Składowe SQL

- **DML** (*Data Manipulation Language*) — instrukcje służące do wybierania, modyfikowania, usuwania i wstawiania danych do bazy.
- **DDL** (*Data Definition Language*) — instrukcje służące do tworzenia struktur danych takich jak tabele.
- **DCL** (*Data Control Language*) — instrukcje służące do opisu kontroli dostępu do danych.

Przykłady Poleceń SQL

```
SELECT avg(salary) AS "Average Salary", department_id
FROM employees GROUP BY department_id ORDER BY "Average Salary";
```

```
UPDATE employees SET salary = salary + 500
WHERE department_id = 10;
```

```
CREATE TABLE employees(
   employee_id INTEGER PRIMARY KEY,
   name VARCHAR(60) NOT NULL,
   salary NUMBER(6,2) NOT NULL,
   department_id INTEGER);
```

Przykład Programu Java Korzystającego z JDBC

```
Connection conn
     = DriverManager.getConnection(URL,USR,PASS);
Statement stmt = conn.createStatement();
String sql = "SELECT emp_id, name FROM Employees";
ResultSet rs = stmt.executeQuery(sql);
while(rs.next()){
   int id  = rs.getInt("emp_id");
   String name = rs.getString("name");
   System.out.println("emp_id: " + id);
   System.out.println("name: " + name);
}
rs.close();
stmt.close();
conn.close();
```