Bazy danych

1. Pojęcia podstawowe

P. F. Góra

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

2020/21

Baza danych:

Duża kolekcja danych, odpowiednio zorganizowana w celu szybkiego przeszukiwania i dostępu do informacji, uzyskiwanego przy pomocy komputera.

Używaj właściwych narzędzi!

Małe bazy danych można przetwarzać w dowolnym narzędziu komputerowym. Ale co to znaczy *mała baza*? Coś, co wydaje się małe, niepostrzeżenie może stać się duże. Portal `marketwatch.com` donosił 5 października 2020:

Microsoft Excel coding issue drives big spike in U.K. COVID tally

Published: Oct. 5, 2020 at 6:52 a.m. ET

By Steve Goldstein

13

The Twitter feed of Public Health England, an executive agency of the Department of Health and Social Care in the U.K.
JUSTIN TALLIS/AGENCE FRANCE-PRESSE/GETTY IMAGES

The official U.K. COVID-19 tally spiked this weekend — because of a backlog caused by a Microsoft Excel error.

According to Public Health England, 15,841 cases between Sept. 25 and Oct. 2 weren't uploaded to the government dashboard, because the column limits on an Excel spreadsheet reached its maximum size.

Excel is not a database. Excel is not a database. Excel is not a database. . . ☹

Patron?

Św. Izydor z Sewilli (VI wiek), patron Internetu, stworzył pierwszy katalog

Bazy danych są wszechobecne!

- użytkownicy*
- projektanci/programiści aplikacji bazodanowych†
- projektanci/programiści baz danych, administratorzy baz danych
- projektanci/programiści systemów bazodanowych (DataBase Management System, DBMS)
- projektanci programiści innych systemów i usług‡

*Widzą tylko front–end

†Błędnie nazywanych “bazami danych”

‡Niekiedy nie wiedzą, że projektują/programują systemy bazodanowe ☺

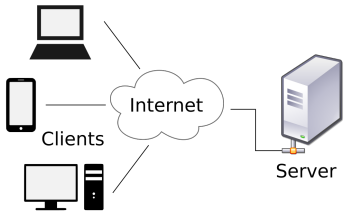
Cel wykładu:

Projektowanie i programowanie baz danych

Zrozumienie najważniejszych algorytmów bazodanowych

Poznanie problematyki systemów rozproszonych

Architektura klient-serwer



lasagna code:

aplikacja bazodanowa

system operacyjny $\rightarrow$ $\updownarrow$ $\nwarrow$

klient ? business logic

sieć $\rightarrow$ $\updownarrow$ $\swarrow$

serwer

Czy *logikę biznesową* umieszczać “po stronie serwera” czy “po stronie aplikacji”?

Architektura klient-serwer (cd)

- sytemy wielodostępne, heterogeniczne
- najbardziej kosztowna operacja: przesyłanie danych
- wiele operacji wykonywanych “po stronie serwera”
- Logika biznesowa *raczej* “po stronie serwera”
- struktura bazy (schemat) i więzy umieszczone bezpośrednio w bazie, nie w aplikacji

Schemat i instancja

Baza danych to, w pewnym sensie, abstrakcyjny model systemu o określonym **schemacie**.

Schemat bazy danych zawiera informację o tableach (lub innych obiektach) w bazie, kolumnach, typach danych, indeksach, wzajemnych powiązaniach pomiędzy tabelami, a także o użytkownikach, ich uprawnieniach itd itd.

Konkretne wystąpienie bazy danych, wraz konkretną (chwilową!) zawartością danych, nazywa się **instancją**.

Każda instancja musi być fizycznie zlokalizowana na jakimś sprzęcie. Co zrobić, gdy zaczyna brakować miejsca, mocy obliczeniowej i innych zasobów?

Skalowanie i elastyczność

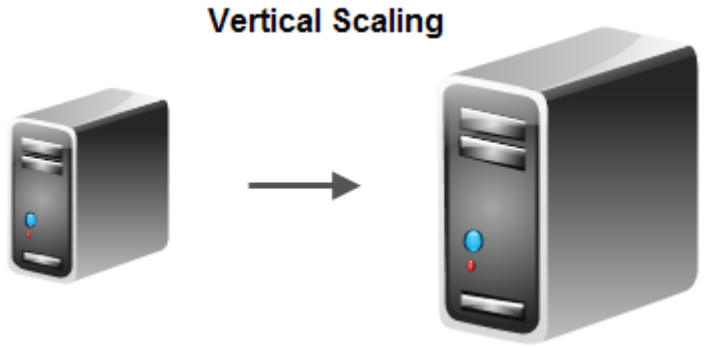
Gdy brakuje zasobów, *skalujemy* nasz system bazodanowy: rozbudowujemy serwer lub dodajemy nowe serwery.

Skalowalność: Zdolność systemu, lub sieci, lub procesu do wykonania narastającej liczby zadań lub też potencjalna możliwość rozbudowania systemu tak, aby poradził sobie z narastającą liczbą zadań.

Elastyczność: Stopień, w jakim system potrafi dostosować się do zmiennego w czasie obciążenia poprzez uruchamianie i zwalnianie zasobów, tak, aby w każdej chwili zaangażowane zasoby jak najlepiej odpowiadały bieżącemu zapotrzebowaniu.

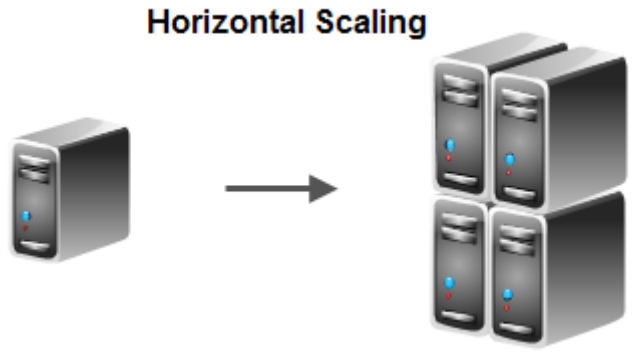
Skalowanie pionowe

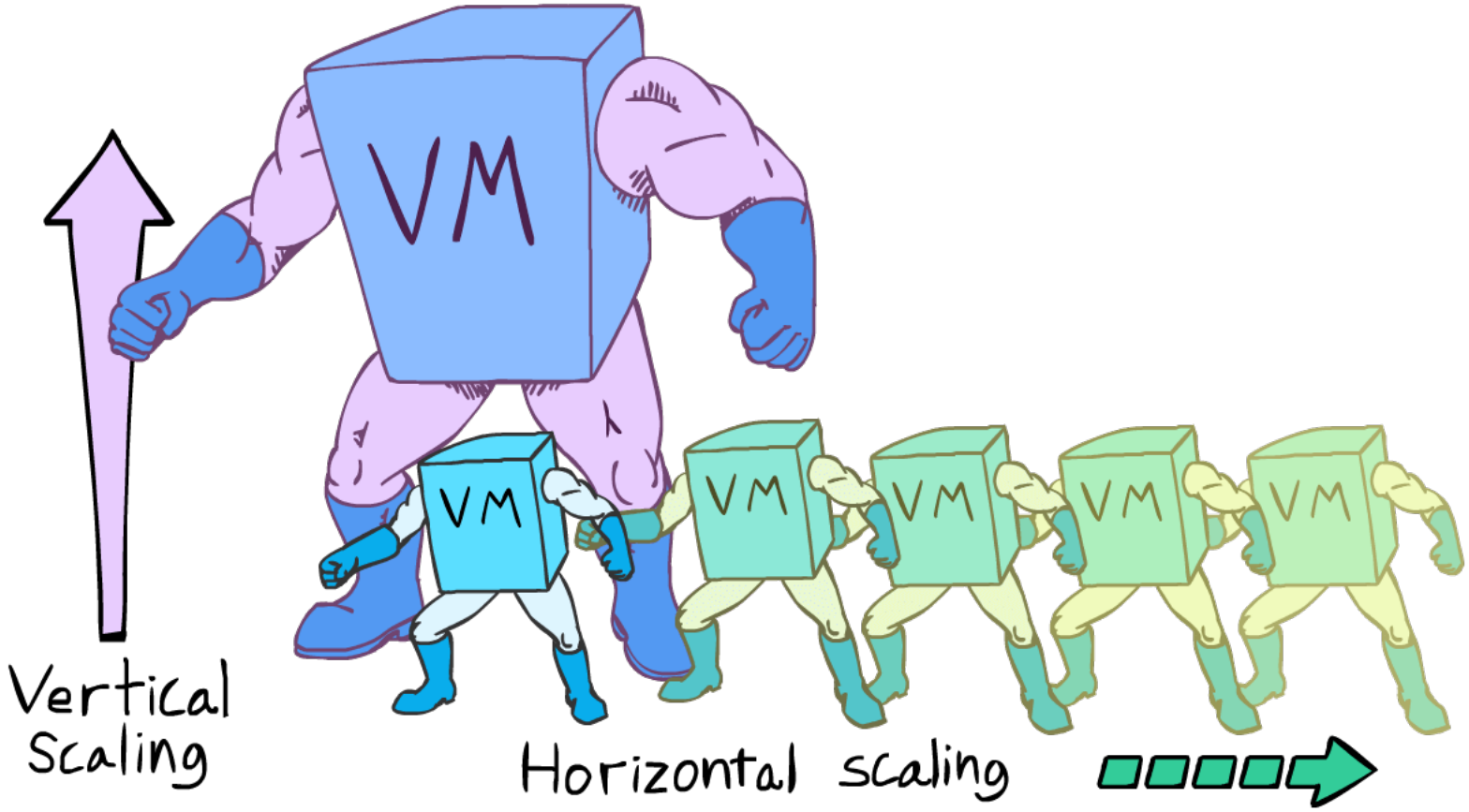
Skalowanie pionowe (vertical scaling, scaling up) polega na dokładaniu zasobów (pamięci, przestrzeni dyskowej, procesorów, portów komunikacyjnych etc) do istniejącego serwera.



Skalowanie poziome

Skalowanie poziome (horizontal scaling, scaling out) polega na dokładaniu serwerów, które przechowują fragmenty lub kopie bazy danych i obsługują część ruchu.





Skalowanie pionowe	
Zalety	Wady
nie ma spowolnień w celu zapewnienia spójności danych	bardzo niska elastyczność trzeba przenosić całą bazę na mniejszy/większy serwer
mniejsze koszta eksploatacyjne mniejsze zużycie mocy mniejszy koszt chłodzenia mniejsza zajętość serwerowni	bardzo wysokie koszta jednostkowe
nie wymaga dodatkowego oprogramowania	awaria unieruchamia *całą* bazę
nie wymaga dodatkowego sprzętu sieciowego	ograniczenia fizyczne
	prawo malejących przychodów

Prawo Amdahla

Jeśli mamy jeden procesor, dołożenie drugiego da nam większy zysk, niż dołożenie setnego procesora do 99 istniejących.

Niech α oznacza część operacji, które muszą być wykonywane sekwencyjnie. $1-\alpha$ jest częścią operacji, którą można zrównoleglić. Wówczas jeśli mamy P procesorów, prawo Amdahla głosi, że osiągniemy przyspieszenie równe

$$S = \frac{1}{\alpha + \frac{1-\alpha}{P}}$$

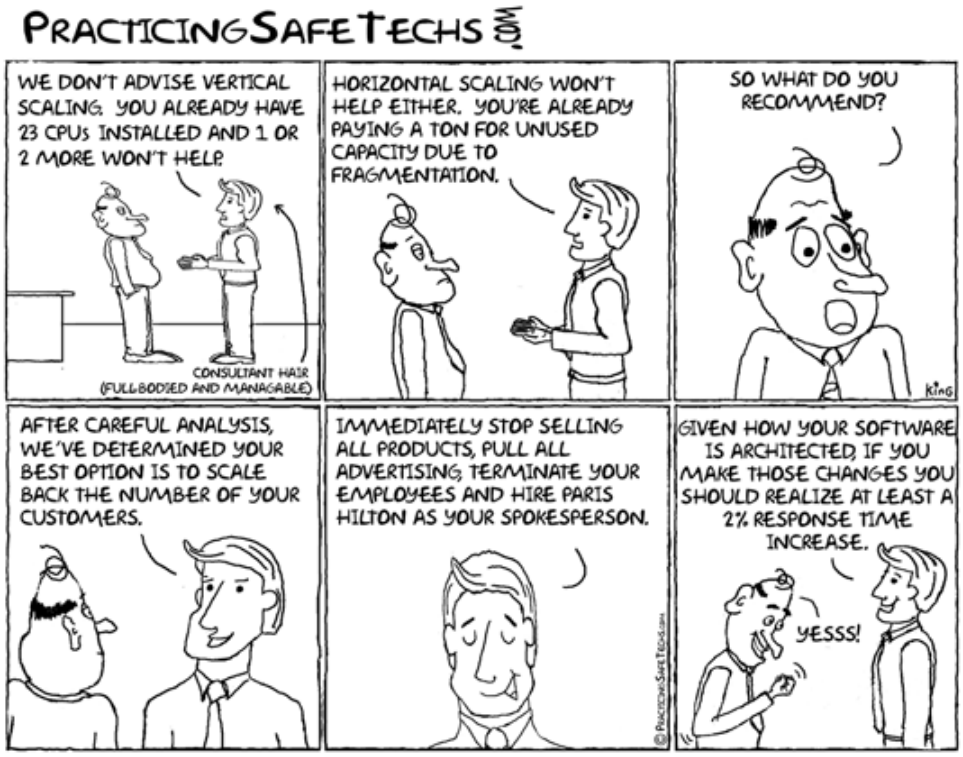
Przypadki graniczne: $\alpha = 0$ (wszystko da się zrównoleglić): $S = P$.
Dla $\alpha > 0$

$$\lim_{P \to \infty} S = \frac{1}{\alpha}$$

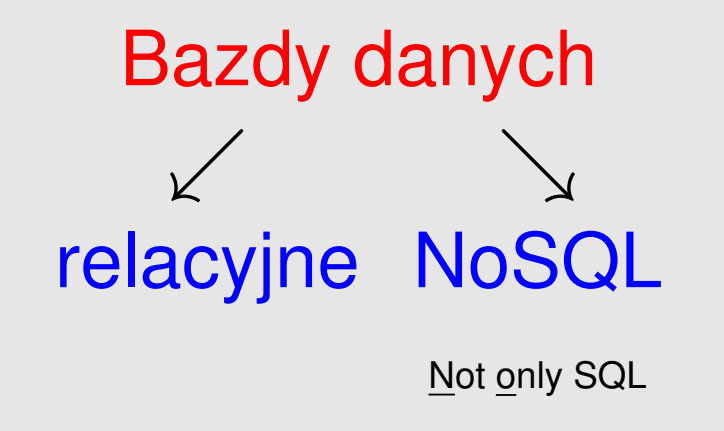
Prawo Amdahla jest szczególnym przypadkiem prawa malejących przychodów (*the law of diminishing returns*). W pewnym momencie zwiększanie zasobów jedynego serwera przestaje się opłacać.

Skalowanie poziome	
Zalety	Wady
duża elastyczność	możliwe znacznie opóźnienia ze względu na zapewnienie spójności danych
niższy koszt jednostkowy	większe koszta eksploatacyjne większe zużycie mocy, większy koszt chłodzenia większa zajętość serwerowni
awaria *jednego* serwera nie unieruchamia całej bazy	potencjalnie niejednorodne oprogramowanie wymagane więcej sprzętu sieciowego nierównomierne obciążenie serwerów (fragmentacja)

W praktyce decyzja o wyborze skalowania bywa trudna. Zależy od charakteru bazy, liczby klientów jednocześnie komunikujących się z bazą, a nawet od (typowej) zawartości bazy (instancji bazy), posiadanego budżetu itp.



Zasadniczy podział



- Bazy relacyjne — $\sim$ 80% rynku
- Bazy NoSQL — największe firmy technologiczne (Amazon, Facebook) ☺ plus $\sim$ 95% aplikacji wytwarzanych przez niczego nieświadomych programistów ☹

Relacyjne bazy danych

- Oparte o mocne matematyczne podstawy
- Wymagają dobrego (kosztownego, czasochłonnego, przemyślanego) zaprojektowania
- Instytucje rządowe, banki, duże firmy (tradycyjne)
- Chronią przed wieloma błędami
- Zapewniają bezpieczeństwo danych
- Zapewniają stałą spójność danych pomiędzy serwerami
- . . . i generują wynikające z tego opóźnienia
- Możliwy **bardzo** długi czas życia
- Kłopot, gdy dane nie pasują do schematu
- Mogą obsługiwać setki, a nawet tysiące jednoczesnych użytkowników

Bazy relacyjne i skalowanie poziome

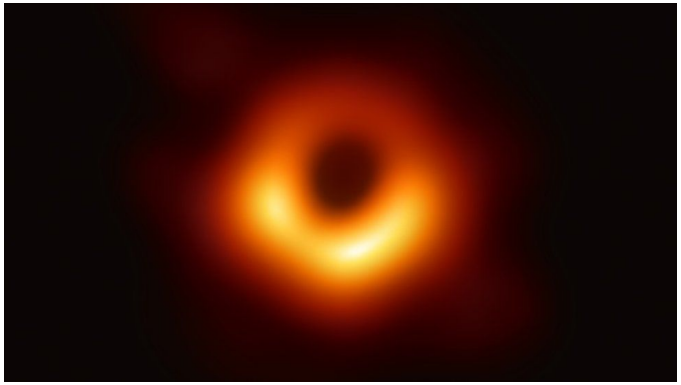
- replikacja
  - dla zapewnienia kopii bezpieczeństwa
  - jeśli dane są głównie do odczytu, modyfikacja stanu bazy jest rzadka, replikacja nie powoduje dużych opóźnień dla zapewnienia spójności
  - w przeciwnym wypadku możliwe znaczne opóźnienia
- podział dużych tabel ze względu na wartość klucza

Bazy NoSQL

- Luźniejsze, gorzej zdefiniowane podstawy
- Nie wymagają *aż tak* dokładnego projektowania
- Największe firmy nowych technologii plus większość małych firm nowych technologii
- Większa odpowiedzialność spoczywa na programiście
- Mnóstwo oprogramowania “samo” generuje takie bazy
- Nie muszą zapewniać stałej spójności danych pomiędzy serwerami
- . . . zapewniając lepszą dostępność, kosztem niejednoznacznych odczytów
- Typowo raczej krótki czas życia
- Znacznie bardziej elastyczne przy zróżnicowanych danych
- Pomyślane do obsługiwania dziesiątek i setek tysięcy jednoczesnych użytkowników

Bazy NoSQL i skalowanie poziome

- bazy NoSQL nie wymagają spójności danych
- skalowanie poziome jest dla baz NoSQL naturalne
- Big Data praktycznie wyłącznie w skalowaniu poziomym



pierwsze zdjęcie czarnej dziury

Katie Bouman i część dysków z surowymi danymi

Do zrobienia pierwszego zdjęcia czarnej dziury potrzebne było 5 petabajtów surowych danych. Przy technologii z 2018 wymagało to pół tony twardych dysków.