1. Tidy Data Frame

- **Definicja**: Dane w formacie "Tidy" są zorganizowane w sposób zgodny z zasadami analizy statystycznej i wizualizacji danych.
- **Cechy charakterystyczne (Hadley Wickham, 2014)**:
    1. **Każda zmienna to kolumna.**
    2. **Każda obserwacja to wiersz.**
    3. **Każdy typ jednostki obserwacyjnej to osobna tabela.**
- **Korzyści**:
    - Łatwość manipulacji i analizy danych.
    - Kompatybilność z narzędziami do analizy danych, np. Python (`pandas`) czy R (`tidyverse`).
- **Przykład danych Tidy**:

ID	Name	Age	Country
1	John	28	USA
2	Maria	35	Poland
3	Hiroshi	42	Japan

Czym jest zestaw danych?

- Większość statystycznych zestawów danych to **prostokątne tabele** złożone z wierszy i kolumn:
    - **Kolumny** są prawie zawsze oznaczone etykietami.
    - **Wiersze** czasami mają etykiety.
- **Zestaw danych** to zbiór wartości:
    - **Liczbowych** (jeśli dane są ilościowe).
    - **Łańcuchów znaków** (jeśli dane są jakościowe).

Organizacja wartości w zestawie danych

- Każda wartość należy do:
    - **Zmiennej** – reprezentuje jedną cechę, np. wysokość, temperatura, czas trwania.
    - **Obserwacji** – zawiera wartości dla jednego obiektu, np. osoby, dnia, wyścigu.

Tidy Data – Definicja

- **Tidy Data** to standard organizacji danych, który mapuje znaczenie danych na ich strukturę.
- Dane są **czyste (tidy)**, jeśli spełniają poniższe zasady:
    1. Każda zmienna tworzy **kolumnę**.
    2. Każda obserwacja tworzy **wiersz**.
    3. Każdy typ jednostki obserwacyjnej tworzy **tabelę**.
- Dane nieczyste (**messy**) to każde inne ułożenie danych.

Inspiracja normalizacją danych

- Zasady Tidy Data są zbliżone do **3. postaci normalnej (3NF)** w modelu relacyjnym, opisanej przez E.F. Codda (1990).
  Różnice:
  - Ramy są opisane w języku statystyki.
  - Skupiają się na pojedynczym zbiorze danych, a nie wielu połączonych zbiorach.

Problemy z nieczystymi danymi i ich rozwiązania

1. **Nagłówki kolumn są wartościami, a nie nazwami zmiennych**.
   Rozwiązanie: Przekształć dane tak, aby każda zmienna miała własną kolumnę.
2. **Wiele zmiennych zapisanych w jednej kolumnie**.
   Rozwiązanie: Rozdziel dane na oddzielne kolumny.
3. **Zmienne są przechowywane zarówno w wierszach, jak i kolumnach**.
   Rozwiązanie: Przekształć dane na spójny układ kolumnowy.
4. **Wiele typów jednostek obserwacyjnych przechowywanych w jednej tabeli**.
   Rozwiązanie: Rozdziel dane na osobne tabele.
5. **Jednostka obserwacyjna rozdzielona na wiele tabel**.
   Rozwiązanie: Połącz dane w jedną tabelę.

Dlaczego Tidy Data jest ważne?

- Tidy Data ułatwia:
  - **Manipulację danymi**.
  - **Modelowanie danych**.
  - **Wizualizację**.
- Dzięki spójnej strukturze:
  - Niewielki zestaw narzędzi wystarczy do czyszczenia danych.
  - Narzędzia wejściowe i wyjściowe mogą operować na spójnej strukturze.

Kluczowe korzyści

- **Łatwiejsze czyszczenie danych**:
  - Mniej czasu spędzonego na uciążliwej manipulacji.
  - Większa efektywność analizy.
- **Prostota narzędzi**:
  - Jednolite struktury danych pozwalają tworzyć specjalistyczne narzędzia, które upraszczają proces analizy.

Przykłady zastosowań

- Case study z publikacji:
  - Demonstracja korzyści wynikających z **spójnej struktury danych**.
  - Unikanie powtarzalnych i skomplikowanych operacji manipulacji.

2. Bazy klucz-wartość (Key-Value Stores)

- **Definicja**: Bazy danych, w których dane są przechowywane w postaci par klucz-wartość.
- **Przykłady**:

    - Redis
    - Amazon DynamoDB
    - Riak
- **Zastosowania**:
    - Przechowywanie sesji użytkowników.
    - Cache'owanie danych.
    - Szybki dostęp do danych o prostym schemacie.
- **Zalety**:
    - Wysoka wydajność.
    - Skalowalność pozioma.
- **Przykład**:

```
{
    "user_123": {"name": "Alice", "age": 30},
    "user_456": {"name": "Bob", "age": 25}
}
```

Klucz	
• anna • Opole:15 • Wtorek:13:30 • 2022-02-22 2:22 • 1515 • 45-046 • adam@poczta.pl • HHDE:23423:DKIIIJD949434234:A	• Dowolny binarny ciąg • Unikalność • Może być generowany przez algorytm • Może być wyliczany • Niezbyt długi (wydajność)

Wartość	
• Jest powiązana z kluczem, • Ustawianie, odczytywanie, kasowanie wartości z użyciem klucza, • Liczby, teksty, JSON, obrazy, strony, … cokolwiek • Ograniczenia rozmiaru	

Konwencja	
KLUCZ:	WARTOŚĆ:
Budownictwo:5:2021 -> Budownictwo:zima:lab:15 -> Adam:Kwiatkowski -> Ewa:Kwiatkowska ->	• Matematyka Dyskretna • 124 320 500 • (55, 'Prószkowska', '2022-12-12') • {'Opole': 33, 'Kraków': 45, 'Łódź': 11}

Implementacje:
- Redis
- DynamoDB
- Riak
- Project Voldemort

Zalety:
• Prostota obsługi
• Elastyczność

• Wydajność
• Skalowalność (horyzontalna, sharding)
Wady?:
• Prostota …?
• Wyszukiwanie…

Zastosowania:
• Cache
  • Querry cache
• Session Cache
  • maintaining a cache isn't typically mission critical with regards to consistency, most users wouldn't exactly enjoy if all their cart sessions went away, now would they?
• Full Page Cache (FPC)
  • from something like PHP native FPC, a full page cache backend. (i.e. for WordPress)
• Queues
  • a message queue, using push/pop operations with lists in programming languages such as Python, i.e. Celery, has a backend using Redis as a broker.
• Leaderboards/Counting
  • amazing at increments and decrements since it's in-memory. Sets and sorted sets
  • i.e. pull the top 10 users from a sorted set • ZRANGE user_scores 0 10

3. Bazy danych plikowe

- **Definicja**: Bazy danych, w których dane są przechowywane jako pliki, bez potrzeby użycia serwera bazodanowego.
- **Przykłady**:
  - SQLite
  - HDF5
  - Parquet
- **Zalety**:
  - Łatwość instalacji (brak serwera).
  - Dobre dla aplikacji desktopowych lub lokalnych projektów.
- **Ograniczenia**:
  - Słaba obsługa dużych zbiorów danych w porównaniu z systemami serwerowymi.
- **Zastosowania**:
  - Prototypowanie aplikacji.
  - Przechowywanie danych dla aplikacji mobilnych.

4. BI (Business Intelligence)

- **Definicja**: Proces analizy danych biznesowych w celu wsparcia decyzji strategicznych i operacyjnych.
- **Elementy BI**:
  - Zbieranie danych (ETL – Extract, Transform, Load).
  - Wizualizacja danych (dashboardy, raporty).
  - Analiza danych (np. statystyki, predykcje).
- **Narzędzia BI**:
  - Microsoft Power BI
  - Tableau

    - QlikView
    - SAP BusinessObjects
- **Korzyści**:
    - Lepsze podejmowanie decyzji dzięki analizie danych.
    - Monitorowanie kluczowych wskaźników (KPI).
    - Automatyzacja raportowania.

5. OLAP i transakcje (OLTP)

- **OLAP (Online Analytical Processing)**:
    - Służy do analizy danych historycznych.
    - Optymalizowane pod kątem zapytań analitycznych.
    - Przykłady zastosowań:
        - Raportowanie sprzedaży.
        - Analiza trendów.
    - **Przykład narzędzia**: OLAP Cube.
- **OLTP (Online Transaction Processing)**:
    - Obsługuje operacje transakcyjne w czasie rzeczywistym.
    - Optymalizowane pod kątem dużej liczby krótkich transakcji.
    - Przykłady zastosowań:
        - Systemy bankowe.
        - Rezerwacje lotów.
    - **Przykład bazy danych**: MySQL, PostgreSQL.

Właściwość	OLAP	OLTP
Typ danych	Historyczne, analityczne	Bieżące, transakcyjne
Liczba operacji	Mniej, ale bardziej złożone	Więcej, ale prostsze
Przykłady narzędzi	Power BI, Tableau	PostgreSQL, MySQL

6. Power BI

- **Definicja**: Narzędzie do wizualizacji danych i analizy BI od Microsoft.
- **Funkcjonalności**:
    - Tworzenie interaktywnych raportów i dashboardów.
    - Integracja z różnymi źródłami danych (SQL, Excel, API itp.).
    - Wbudowane funkcje analityczne i obsługa języka DAX.
- **Zastosowania**:
    - Monitorowanie wyników firmy.
    - Prognozowanie sprzedaży.
    - Analiza danych w czasie rzeczywistym.
- **Przykładowe wizualizacje**:
    - Wykresy słupkowe, liniowe.
    - Mapy ciepła.
    - Analiza KPI (Key Performance Indicators).

Model relacyjny – model organizacji danych bazujący na matematycznej teorii mnogości, w szczególności na pojęciu relacji. Na modelu relacyjnym oparta jest **relacyjna baza danych** (ang. *Relational Database*) – baza danych, w której dane są przedstawione w postaci relacyjnej.W najprostszym ujęciu w modelu relacyjnym dane grupowane są w relacje, które reprezentowane są przez tabele. Relacje są pewnym zbiorem rekordów o identycznej strukturze wewnętrznie powiązanych za pomocą związków zachodzących pomiędzy danymi. Relacje zgrupowane są w tzw. schematy bazy danych. Relacją może być tabela zawierająca dane teleadresowe pracowników, zaś schemat może zawierać wszystkie dane dotyczące firmy. Takie podejście w porównaniu do innych modeli danych ułatwia wprowadzanie zmian, zmniejsza możliwość pomyłek, ale dzieje się to kosztem wydajności.

Podejście intuicyjne:

W modelu relacyjnym każda relacja (prezentowana w postaci np. tabeli) posiada unikatową nazwę, *nagłówek* i *zawartość*. Nagłówek relacji to zbiór atrybutów, gdzie atrybut jest parą *nazwa_atrybutu:nazwa_typu*, zawartość natomiast jest zbiorem krotek (reprezentowanych najczęściej w postaci wiersza w tabeli). W związku z tym, że nagłówek jest *zbiorem atrybutów* nie jest ważna ich kolejność. Atrybuty zazwyczaj utożsamiane są z kolumnami tabeli. Każda krotka (wiersz) wyznacza zależność pomiędzy danymi w poszczególnych komórkach (np. osoba o danym numerze PESEL posiada podane nazwisko i imię oraz adres)

Model relacyjny a SQL:

Większość współczesnych relacyjnych baz danych korzysta z jakiejś wersji języka SQL pozwalającego wprowadzać zmiany w strukturze bazy danych, jak również zmiany danych w bazie i wybieranie informacji z bazy danych. Język ten opiera się na silniku bazy danych, który pozwala zadawać w języku SQL pewnego rodzaju pytania (kwerendy) i wyświetlać dane, które spełniają warunki zapytania. Zapytania SQL mogą także wykonywać operacje wstawiania danych, usuwania danych i ich aktualizacji. Język SQL zapewnia również zarządzanie bazą danych. Informacja o samej bazie przechowywana jest w postaci relacji (tabel) wewnątrz bazy danych.

Przykład relacyjnej bazy danych

Oto prosty przykład dwóch tabel, których mała firma może używać do przetwarzania zamówień na swoje produkty. Pierwsza tabela służy do przechowywania informacji na temat klientów. Jej każdy rekord zawiera nazwę, adres, informacje wysyłkowe i rozliczeniowe, numer telefonu oraz inne dane kontaktowe klienta. Każdy fragment informacji (każdy atrybut) znajduje się w oddzielnej kolumnie, a baza danych przypisuje każdemu wierszowi unikatowy identyfikator (klucz). W drugiej tabeli — służącej do przechowywania zamówień klientów — każdy rekord zawiera identyfikator klienta, który złożył zamówienie, zamówiony produkt, ilość, wybrany rozmiar i kolor itd., ale nie zawiera imienia i nazwiska ani danych kontaktowych klienta.

Te dwie tabele mają tylko jedną wspólną cechę: kolumnę identyfikatora (klucz). Dzięki tej wspólnej kolumnie relacyjna baza danych może utworzyć relację między dwiema omawianymi tabelami. Wówczas, kiedy aplikacja przetwarzająca zamówienia firmy przesyła zamówienie do bazy danych, baza danych może przejść do tabeli zamówień klienta, pobrać poprawne informacje o zamówieniu produktu i użyć identyfikatora klienta z tej tabeli, aby

wyszukać informacje rozliczeniowe i wysyłkowe klienta w tabeli informacji o kliencie. Magazyn może następnie pobrać właściwy produkt, klient może otrzymać zamówienie w terminie, a firma może otrzymać zapłatę.

Jak są zorganizowane relacyjne bazy danych

Model relacyjny oznacza, że logiczne struktury danych — tabele danych, widoki i indeksy — są oddzielone od fizycznych struktur pamięci. Dzięki temu administratorzy baz danych mogą zarządzać fizycznym przechowywaniem danych bez wpływu na dostęp do tych danych jako struktury logicznej. Na przykład zmiana nazwy pliku bazy danych nie powoduje zmiany nazw przechowywanych w nim tabel.

Rozróżnienie między strukturą logiczną a fizyczną dotyczy również operacji na bazach danych, które są wyraźnie zdefiniowanymi działaniami umożliwiającymi aplikacjom manipulowanie danymi i strukturami bazy danych. Operacje logiczne umożliwiają aplikacji określenie potrzebnej zawartości, a operacje fizyczne ustalają, w jaki sposób należy uzyskać dostęp do danych, a następnie wykonują to zadanie.
Aby zapewnić zawsze maksymalną dokładność i dostępność danych, relacyjne bazy danych przestrzegają określonych reguł integralności. Reguła integralności może na przykład określać, że w tabeli nie są dozwolone duplikaty wierszy, eliminując w ten sposób możliwość wprowadzenia do bazy danych błędnych informacji.

Model relacyjny

We wczesnych latach rozwoju baz danych każda aplikacja zapisywała dane w swojej własnej unikatowej strukturze. Chcąc tworzyć aplikacje korzystające z tych danych, programiści musieli dobrze znać konkretną strukturę danych, aby znaleźć potrzebne dane. Te struktury danych były nieefektywne i trudne do utrzymania. Trudno je też było zoptymalizować, aby zapewnić wysoką wydajność aplikacji. Relacyjny model bazy danych zaprojektowano w celu rozwiązania problemu istnienia wielu arbitralnych struktur danych.

Relacyjny model danych zapewnił standardowy sposób reprezentowania i wysyłania zapytań dotyczących danych, z którego można było korzystać w każdej aplikacji. Od samego początku programiści uznali, że główną siłą relacyjnego modelu bazy danych było wykorzystywanie tabel, które oferowały intuicyjny, wydajny i elastyczny sposób przechowywania ustrukturyzowanych informacji oraz uzyskiwania do nich dostępu.

Z czasem ujawniła się jeszcze jedna zaleta modelu relacyjnego. Programiści mogli bowiem korzystać z języka SQL (structured query language, strukturalny język zapytań) przeznaczonego do zadawania pytań i wyszukiwania potrzebnych informacji w bazie danych. Przez wiele lat SQL był powszechnie używany jako język do formułowania zapytań do baz danych. Oparty na algebrze relacyjnej język SQL to wewnętrznie spójny język matematyczny, zapewniający wyższą wydajność wszystkich zapytań do baz danych. Dla porównania, inne podejścia wymagają definiowania pojedynczych zapytań.

Korzyści płynące z systemu zarządzania relacyjną bazą danych

Prosty, ale silny model relacyjny jest używany przez różnego rodzaju i różnej wielkości przedsiębiorstwa do zaspokajania szerokiej gamy potrzeb informacyjnych. Relacyjne bazy danych służą do śledzenia zapasów, przetwarzania transakcji w handlu elektronicznym,

zarządzania ogromnymi ilościami kluczowych informacji o klientach itd. Relacyjna baza danych może służyć do zaspokajania dowolnych potrzeb informacyjnych w sytuacjach, w których elementy danych są ze sobą powiązane i muszą być zarządzane w sposób bezpieczny, oparty na regułach i spójny.

Relacyjne bazy danych istnieją od lat 70-tych XX wieku. Zalety modelu relacyjnego sprawiają, że jest to nadal najszerzej akceptowany model baz danych.

Model relacyjny i spójność danych

Model relacyjny najskuteczniej zachowuje spójność danych w aplikacjach i kopiach baz danych (zwanych instancjami). Na przykład, gdy klient wpłaca pieniądze w bankomacie, a następnie sprawdza saldo konta w telefonie komórkowym, oczekuje, że natychmiast zobaczy ten depozyt w zaktualizowanym saldzie konta. Relacyjne bazy danych pozwalają zapewnić taką spójność danych, sprawiając, że wiele instancji bazy danych zawiera przez cały czas te same dane.

W przypadku innych typów baz danych utrzymanie przez cały czas takiego poziomu spójności przy dużej ilości danych jest trudne. Niektóre najnowsze bazy danych, takie jak NoSQL, mogą zapewniać tylko „spójność końcową”. Zgodnie z tą zasadą, gdy baza danych jest skalowana lub gdy wielu użytkowników korzysta z tych samych danych w tym samym czasie, dane potrzebują trochę czasu, aby „dogonić” stan aktualny. Spójność końcowa jest akceptowalna w przypadku niektórych zastosowań, takich jak prowadzenie list w katalogu produktów, ale w przypadku newralgicznych operacji biznesowych, takich jak transakcje związane z obsługą koszyka zakupów, „złotym standardem” jest nadal relacyjna baza danych.

Zatwierdzenie i niepodzielność

Relacyjne bazy danych obsługują reguły i zasady biznesowe na bardzo szczegółowym poziomie — obowiązują na przykład bardzo rygorystyczne zasady zatwierdzania (tj. wprowadzania zmian w bazie danych na stałe). Rozważmy na przykład bazę danych dotyczącą stanu zapasów, która monitoruje trzy zawsze używane razem części. Kiedy jedna z tych części jest pobierana z zapasów, muszą zostać pobrane również pozostałe dwie. Jeśli jedna z tych trzech części jest niedostępna, nie może zostać pobrana żadna z nich — aby baza danych dokonała zatwierdzenia, muszą być dostępne wszystkie trzy części. Relacyjna baza danych nie zatwierdzi do wydania jednej części, o ile nie może zatwierdzić wszystkich trzech. Ta wielopłaszczyznowość zatwierdzania jest nazywana niepodzielnością. Niepodzielność jest kluczem do zachowania dokładności danych w bazie danych i zapewnienia ich zgodności z zasadami, przepisami i polityką firmy.

Właściwości ACID a RDBMS

Cztery kluczowe właściwości definiują transakcje w relacyjnych bazach danych: niepodzielność, spójność, izolacja i trwałość — zwykle są one określane jako ACID (ang. atomicity, consistency, isolation, and durability).

- **Niepodzielność** określa wszystkie elementy składające się na kompletną transakcję bazy danych.
- **Spójność** określa zasady utrzymywania elementów danych we właściwym stanie po transakcji.

- **Izolacja** oznacza, że efekt transakcji będzie niewidoczny dla innych, dopóki nie zostanie ona zatwierdzona, aby uniknąć nieporozumień.
- **Trwałość** oznacza, że zmiany w danych stają się trwałe po zatwierdzeniu transakcji.

Procedury składowane i relacyjne bazy danych

Dostęp do danych obejmuje wiele powtarzających się czynności. Na przykład proste zapytanie, które ma zapewnić uzyskanie informacji z tabeli danych, może wymagać powtórzenia setek lub tysięcy razy, aby uzyskać pożądany wynik. Te funkcje dostępu do danych wymagają jakiegoś rodzaju kodu, aby uzyskać dostęp do bazy danych. Twórcy aplikacji nie chcą pisać nowego kodu dla tych funkcji w każdej nowej aplikacji. Na szczęście relacyjne bazy danych obsługują procedury składowane będące blokami kodu, do których można uzyskać dostęp poprzez proste wywołanie aplikacji. Na przykład pojedyncza procedura składowana może zapewnić spójne znakowanie rekordów dla użytkowników wielu aplikacji. Procedury składowane mogą również pomagać programistom w zadbaniu o to, aby określone funkcje danych w aplikacji zostały zaimplementowane w konkretny sposób.

Blokowanie bazy danych i współbieżność

Kiedy wielu użytkowników lub wiele aplikacji próbuje jednocześnie zmienić te same dane, w bazie danych mogą występować konflikty. Techniki blokowania i współbieżności zmniejszają ryzyko konfliktów przy jednoczesnym zachowaniu integralności danych.

Blokowanie uniemożliwia innym użytkownikom i aplikacjom dostęp do danych podczas ich aktualizacji. W niektórych bazach danych blokowanie dotyczy całej tabeli, co ma negatywny wpływ na wydajność aplikacji. Inne bazy danych, takie jak relacyjne bazy danych Oracle, stosują blokady na poziomie rekordu, umożliwiając dostęp do pozostałych rekordów w tabeli, co pomaga zapewnić lepszą wydajność aplikacji.

Współbieżność występuje wówczas, gdy wielu użytkowników lub wiele aplikacji jednocześnie wywołuje zapytania dotyczące tej samej bazy danych. Funkcja ta zapewnia odpowiedni dostęp użytkownikom i aplikacjom, zgodnie ze zdefiniowanymi zasadami kontroli danych.

Na co zwracać uwagę przy wyborze relacyjnej bazy danych

Oprogramowanie używane do przechowywania, wyszukiwania i pobierania danych przechowywanych w relacyjnej bazie danych oraz do zarządzania nimi to tzw. system zarządzania relacyjnymi bazami danych (RDBMS). RDBMS zapewnia interfejs między użytkownikami i aplikacjami a bazą danych, a także funkcje administracyjne do zarządzania przechowywaniem danych, dostępem i wydajnością.

Na wybór określonego typu bazy danych i produktów związanych z relacyjną bazą danych wypływa kilka czynników. Wybór RDBMS zależy od potrzeb biznesowych firmy. Należy sobie wtedy zadać następujące pytania:

- Jakie są nasze wymagania w zakresie dokładności danych? Czy przechowywanie i dokładność danych zależą od logiki biznesowej? Czy nasze dane są objęte rygorystycznymi wymaganiami dotyczącymi dokładności (na przykład dane finansowe i raporty dla organów administracji publicznej)?

- Czy potrzebujemy skalowalności? Jaka jest skala zarządzanych danych i jaki jest oczekiwany wzrost ich ilości? Czy model bazy danych będzie musiał obsługiwać lustrzane kopie bazy danych (jako osobne instancje) w celu skalowania? Jeśli tak, czy może utrzymać spójność danych w tych instancjach?
- Jak ważna jest współbieżność? Czy wielu użytkowników i wiele aplikacji będzie potrzebować jednoczesnego dostępu do danych? Czy oprogramowanie bazy danych obsługuje współbieżność, jednocześnie chroniąc dane?
- Jakie są nasze potrzeby w zakresie wydajności i niezawodności? Czy potrzebujemy produktu o wysokiej wydajności i niezawodności? Jakie są wymagania dotyczące wydajności odpowiedzi na zapytania? Jakie są zobowiązania dostawcy dotyczące umów o poziom usług (SLA) lub nieplanowanych przestojów?

Relacyjna baza danych przyszłości: samoczynna baza danych

Z biegiem lat relacyjne bazy danych stały się lepsze, szybsze, wydajniejsze i łatwiejsze w obsłudze. Ale są także bardziej skomplikowane, a administrowanie bazą danych od dawna jest zajęciem na pełen etat. Zamiast wykorzystywać swoją wiedzę i koncentrować się na opracowywaniu innowacyjnych aplikacji przynoszących firmie korzyści, programiści muszą spędzać większość czasu na działaniach związanych z zarządzaniem, niezbędnych do zapewnienia optymalnego działania bazy danych.

Oferowana obecnie technologia autonomiczna opiera się na mocnych stronach modelu relacyjnego, chmurowej bazy danych oraz uczenia maszynowego, udostępniając nowy typ relacyjnej bazy danych. Samoczynna baza danych (zwana również autonomiczną bazą danych) zachowuje moc obliczeniową i zalety modelu relacyjnego, ale wykorzystuje sztuczną inteligencję (AI), samouczenie się maszyn i automatyzację do monitorowania obsługi zapytań i zadań zarządzania oraz zwiększania jej wydajności. Na przykład aby poprawić wydajność obsługi zapytań, samoczynna baza danych może postawić hipotezę i przetestować indeksy w celu przyspieszenia tego procesu, a następnie przekazać najlepsze wyniki do produkcji — wszystko robi samodzielnie. Samoczynna baza danych wprowadza tego rodzaju usprawnienia nieustannie, nie angażując przy tym człowieka.
Technologia autonomiczna zwalnia programistów z wykonywania żmudnych zadań związanych z zarządzaniem bazą danych. Nie muszą oni na przykład z wyprzedzeniem definiować wymagań dotyczących infrastruktury. Zamiast tego, dzięki samoczynnej bazie danych, mogą swobodnie dodawać zasoby pamięci masowej i moc obliczeniową w miarę rozrastania się bazy danych. Programiści mogą łatwo, w zaledwie kilku krokach, utworzyć autonomiczną relacyjną bazę danych, przyspieszając czas tworzenia aplikacji.

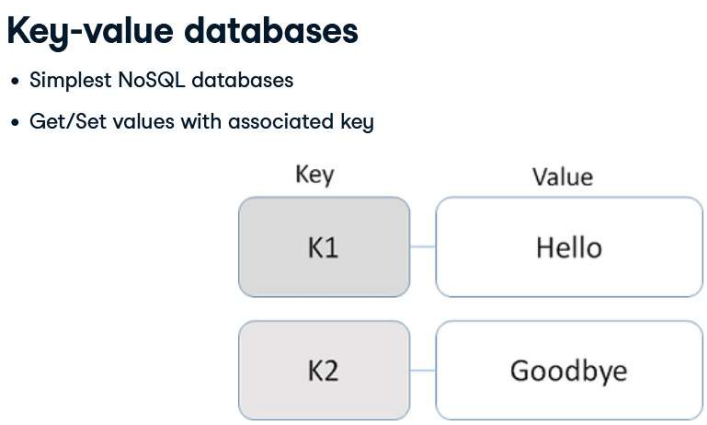
Bazy klucz-wartość

1

Kiedy prędkość ma znaczenie…

2

Bazy klucz - wartość



3

Klucz

- anna
- Opole:15
- Wtorek:13:30
- 2022-02-22 2:22
- 1515
- 45-046
- adam@poczta.pl
- HHDE:23423:DKIIIJD949434234:A

- Dowolny binarny ciąg
- Unikalność
- Może być generowany przez algorytm
- Może być wyliczany
- Niezbyt długi (wydajność)

4

Wartość

- Jest powiązana z kluczem,
- Ustawianie, odczytywanie, kasowanie wartości z użyciem klucza,
- Liczby, teksty, JSON, obrazy, strony, … cokolwiek
- Ograniczenia rozmiaru

konwencja :

KLUCZ:	WARTOŚĆ:
Budownictwo:5:2021 ->	• Matematyka Dyskretna
Budownictwo:zima:lab:15 ->	• 124 320 500
Adam:Kwiatkowski ->	• (55, ‘Prószkowska’, ‘2022-12-12’)
Ewa:Kwiatkowska ->	• {‘Opole’: 33, ‘Kraków’: 45, ‘Łódź’: 11}

5

Implementacje

6

Zalety

- Prostota obsługi
- Elastyczność
- Wydajność
- Skalowalność (horyzontalna, sharding)

Wady?

- Prostota …?
- Wyszukiwanie…

7

Zastosowania

- Cache
  - Querry cache
- Session Cache
  - maintaining a cache isn't typically mission critical with regards to consistency, most users wouldn't exactly enjoy if all their cart sessions went away, now would they?
- Full Page Cache (FPC)
  - from something like PHP native FPC, a full page cache backend. (i.e. for WordPress)
- Queues
  - a message queue, using push/pop operations with lists in programming languages such as Python, i.e. Celery, has a backend using Redis as a broker.
- Leaderboards/Counting
  - amazing at increments and decrements since it's in-memory. Sets and sorted sets
  - i.e. pull the top 10 users from a sorted set
    - ZRANGE user_scores 0 10

Więcej: https://www.objectrocket.com/blog/how-to/top-5-redis-use-cases/

8

Redis

Redis - overview

- *Remote Dictionary Server*
- Popular **key-value database**
- **Fast in-memory data** structure store
  - In-memory dataset
  - Also allows to persist data to disk
- **Used as:**
  - Database
  - Cache
  - Message broker

- Open source
- **Redis Labs:** 400+ employees

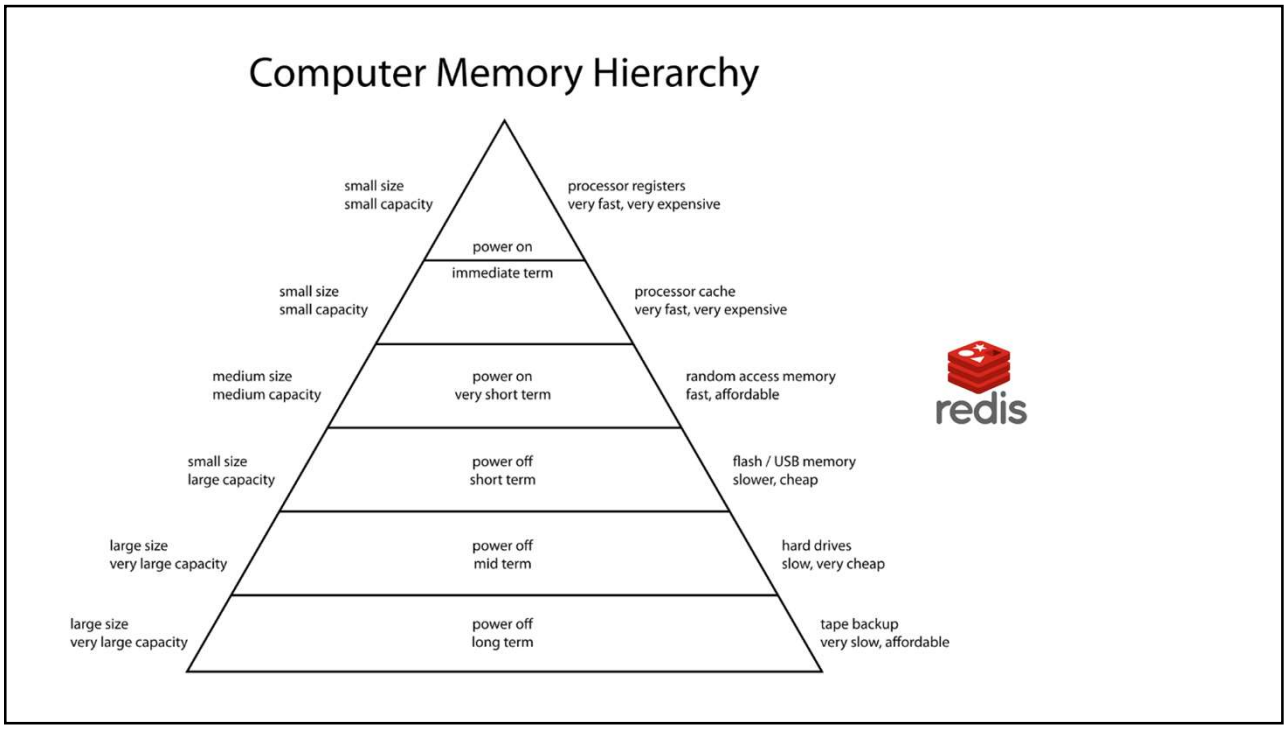
9

```
Redis - a glorified dictionary.

r.get(key)

r.set(key, value)
```

10



11

Redis

Redis - data structures

- Strings

```
SET name Ann
```

- Lists

```
RPUSH my_numbers 1 2 3
```

- Sets

```
SADD my_set 1 2 3
```

- Hashes

```
HMSET user:123 name Ann surname Smith
```

- ...

12

Redis

Redis - features

- Atomic **operations**
- **Transactions**
- **Lua scripting** for complex operations
- **Programming languages:** Python, R, C#, Java, JavaScript, PHP...
- **Asynchronous replication**

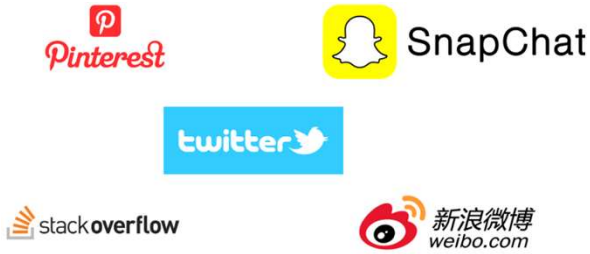
13

Redis - popular uses

- **Caching** (query results, images, files...)
- **Session storage** (user profiles, credentials...)
- **Chatting, messaging, and queues** (chat rooms, real-time comments, social media feeds...)
- **Real-time analytics** (social media analytics, advertisement)
- **Gaming leaderboards** (ranked lists in real-time)
- etc.

14

Ktoś tego używa?



Redis - on the cloud

- Amazon Web Services **Elasticache** for Redis
- **Microsoft Azure Cache** for Redis in Azure
- Alibaba **ApsaraDB** for Redis in Alibaba Cloud

15

Redis on Docker

- Pull the latest Redis Docker image

```
docker pull redis
```

- Run the Redis server as a daemon (background process)
  we map port 6379 on our computer to the exposed port 6379 from the Redis container.
  Port 6379 is the default port for Redis servers.

```
docker run -d -p 6379:6379 --name redis-server redis
```

- Exploring Basic Redis Concepts Using a CLI Client
  Connect to the Redis server using built in Redis CLI client.

```
docker exec -it redis-server redis-cli
    127.0.0.1:6379>
```

16

Przykłady i zadanie

17

Źródła

- https://redis.io/docs/about/
- https://pypi.org/project/redis/

- https://app.datacamp.com/learn/courses/nosql-concepts
- https://www.tutorialspoint.com/redis/redis_overview.htm
- https://realpython.com/python-redis/
- https://betterprogramming.pub/getting-started-with-redis-a-python-tutorial-3a18531a73a6

- https://developer.redis.com/explore/what-is-redis/
- https://blog.anynines.com/redis-what-is-it-and-when-should-i-use-it/

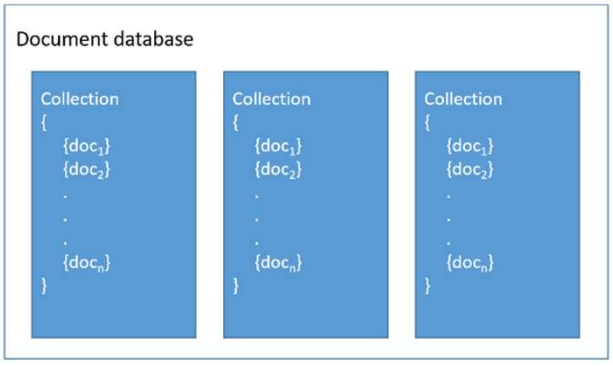
18

Bazy plikowe

1

Bazy plikowe

- Document databases store data in documents.
- Documents are grouped into collections.



- Documents -> rows
- Collections -> tables

2

Bazy plikowe

- Set of key-value pairs
- **Keys**: strings
- **Values**: numbers, strings, booleans, arrays or objects
- **Schemaless**: no need to specify the structure
- **Formats**: JSON, BSON, YAML, or XML

3

Dokumenty – format JSON

```
{
  "user_id": 512,
  "name": "Carol",
  "last_name": "Harper",
  "email": "carolharper@datazy.com",
  "address": {
    "street": "123 Sesame Street",
    "city": "New York City",
    "state": "New York",
    "country": "USA"
  },
  "hobbies": [
    "hiking",
    "painting"
  ]
}
```

- All the users who live in New York and like hiking
- All the users older than 40
- User's data by user_id
- ...

4

Polimorfizm dokumentów

```
{
  "user_id": 512,
  "name": "Carol",
  "last_name": "Harper",
  "email": "carolharper@datazy.com",
  "address": {
    "street": "123 Sesame Street",
    "city": "New York City",
    "state": "New York",
    "country": "USA"
  },
  "hobbies": [
    "hiking",
    "painting"
  ]
}
```

```
{
  "user_id": 513,
  "name": "Benjamin",
  "last_name": "Lieberman",
  "email": "benjaminlieberman@datazy.com",
  "date_of_birth": "07/04/1984",
  "hobbies": [
    "reading"
  ]
}
```

5

Kolekcje

Collections

- Sets of documents
- Store the **same type of entities**
- **Organize** documents and collections by thinking about the **queries**

6

Implementacje

7

Zalety

- Elastyczność
  - **Don't need** to predefine the **schema**
  - **Documents can vary** over time
    - Avoids schema migrations
  - Embedded documents **avoid joins**
    - Better times
  - One of the **first reasons** to choose document databases

8

Zalety

- Intuicyjność
  - **Natural** way to work
  - JSON is **human-readable**
  - **Documents map objects** in code
    - Less coding
    - Simpler and faster development
    - Start coding and storing objects as documents are created
  - **Easier** for new developers

9

Zalety

- horizontal scalability
- document databases can scale horizontally by using the sharding technique.

10

Ograniczenia

- more responsibility
  - Care about data in the **application code**
    - e.g. check required email
  - Care about **redundant data**
    - e.g. modify duplicated name

11

Zastosowania

- **Event logging**
  - Types of events:
    - User logging
    - Product purchase
    - Errors
    - ...
  - Sharding by:
    - Time
    - Type of event
    - ...


```
{
  "type": "info",
  "message": "user_logged",
  "user_id": 551,
  ...
}
```

12

Zastosowania

- **User profiles**

```
{
  "user_id": 512,
  "name": "Carol",
  "last_name": "Harper",
  "email": "carolharper@datazy.com",
  "address": {
    "street": "123 Sesame Street",
    "city": "New York City",
    "state": "New York"
  },
  ...
}
```

- Information may vary
- Document flexibility

13

Zastosowania

- **Content management systems**
  - Blogs, video platforms, etc.
  - Users' content
    - Comments
    - Images
    - Videos
    - ...


```
{
  "id": 458,
  "url": "myblog/datazy.com",
  "title": "How to write a blog entry at Datazy",
  "tags": [
    "Datazy",
    "Blog"
  ],
  "last10comments": [
    { "name": "Eliza", "comment": "Great!" },
    { "name": "Eric", "comment": "Thank you!"}
  ]...
}
```

14

Zastosowania

- **Real-time analytics**
  - Page views, unique visitors...
  - Easy to store the information

```
{
  "_id": "1000241",
  "hour": "Sat Jun 12 2021 16:40:00 GMT+0200 (EST)"
  "site": "datazy",
  "uniques": 5,
  "pageviews": 15,
  ...
}
```

15

Przeciwwskazania

- Mocno ustrukturyzowane dane
- Konieczność zapewnienia ciągłej spójności danych

16

Typy danych

Objects `{}`

- String keys & values `{'key1':value1, 'key2':value2,...}`
- Order of values is not important

```
{
'id': 12345,
'name': 'Donny Winston',
'instructor': true
},
```

Arrays `[]`

- Series of values `[value1, value2,...]`
- Order of values is important

```
[
 "instructor_1",
 "instructor_2",
 ...
]
```

17

Typy danych

```
{
  'people': [
    { 'id': 12345,
      'name': 'Donny Winston',
      'instructor': true,
      'tags': ['Python', 'MongoDB']
    },

    { 'id': 54321
      'name': 'Guido van Rossum'
      'instructor':false
      'tags': null
    },
  ]
}
```

Values

- Strings `'name':'Donny Winston'`
- Numbers `'id': 12345`
- `true` / `false`
- `null`
- Another array `'tags': ['Python', 'MongoDB']`
- Another object `[{ 'id': 12345, ...},...]`

18

MongoDB

- Popular document database
- BSON (Binary JSON) format

- MongoDB Query Language (**MQL**)

```
db.users.find({ "address.zipcode" : "10245" })
```

- Native drivers for **programming languages**: C#, Java, Python, Scala, etc.
- **Indexes** on any field
- **ACID** transactions (Atomicity, Consistency, Isolation, and Durability)
- **Joins** in queries

19

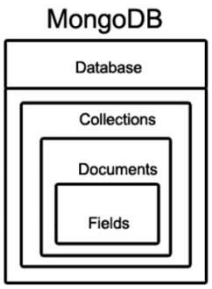
MongoDB

- **MongoDB Compass:**
  - Free GUI
  - Explore schema, create queries visually...
- **MongoDB Atlas:**
  - Cloud service
  - AWS, Azure, Google Cloud
- **MongoDB Enterprise Advanced:**
  - Run MongoDB in our infrastructure
- **MongoDB Atlas Lake:**
  - Query and analyze data
  - AWS S3 and MongoDB Atlas
  - MQL
- **MongoDB Charts:**
  - Visualizations of the data
- **Realm Mobile Database:**
  - Store data locally on iOS or Android

20

MongoDB

MongoDB	JSON	Python
Databases	Objects	Dictionaries
↳Collections	Arrays	Lists
↳↳Documents	Objects	Dictionaries
↳↳↳Subdocuments	Objects	Dictionaries
↳↳↳ Values	Value types	Value types + `datetime`, regex...



21

Typy danych w MongoDB

JSON	Python
Objects	Dictionaries `dict`
Arrays	Lists `list`
Values:	
· *strings*	`str`
· _numbers _	`int`, `float`
· `true` / `false`	`True` / `False`
· `null`	`None`
· *other objects/arrays*	other `dict` / `list`

22

Zastosowania mongoDB

- **Single view applications:** financial services, government, high tech, retail...
- **Gaming:** player profiles, leaderboards...
- **Catalogs:** financial services, government, high tech, retail...
- **Real-time analytics**
- **Content management**
- **Internet of Things**

23

Mongo on Docker

- pobranie obrazu

```
docker pull mongo
```

- uruchomienie obrazu w tle:

```
docker run -d -p 50001:27017 --name mongo-server mongo
```

- zainstalowanie aplikacji klienckiej:

```
sudo apt install mongodb-clients
```

- uruchomienie aplikacji klienckiej:

```
mongo --port 50001
```

24

Przykłady i zadanie

25

Źródła

- https://docs.mongodb.com/
- https://pymongo.readthedocs.io/en/stable/tutorial.html
- https://docs.docker.com/engine/reference/commandline/run/
- https://www.bmc.com/blogs/mongodb-docker-container/
- https://campus.datacamp.com/courses/nosql-concepts/document-databases?ex=1
- https://campus.datacamp.com/courses/introduction-to-using-mongodb-for-data-science-with-python/
- https://docs.mongodb.com/mongocli/master/install/ - instalacja klienta *mongocli*
- https://www.mongodb.com/developer/article/atlas-sample-datasets/ - przykładowe zbiory danych do wykorzystania
- https://docs.atlas.mongodb.com/sample-data/#std-label-load-sample-data - instalacja przykładowych baz danych, np. z użyciem *mongocli*
- https://docs.mongodb.com/manual/tutorial/manage-mongodb-processes/ - uruchamianie bazy MongoDB jako proces, w systemie Linux lub Windows (mongod)
- https://docs.mongodb.com/bi-connector/master/connect/powerbi/ - łczenie z BI do MongoDB

26

WPROWADZENIE

ZAAWANOWANE SYSTEMY BAZ DANYCH

1

ZAAWANOWANE SYSTEMY BAZ DANYCH

Autorstwo wykładu:

- Dr inż. Ewelina Piotrowska
- Dr inż. Bogdan Ruszczak
  - b.ruszczak@po.edu.pl

Katedra Informatyki - Wydział Elektrotechniki, Automatyki i Informatyki

Politechnika Opolska

2

Businness Inteligence

- W języku polskim nie mamy żadnego odpowiednika dla Business Intelligence (czasem stosowane „inteligencja biznesowa", „wywiad gospodarczy", czy „analityka biznesowa" są nieodpowiednie, lub zupełnie mylące).
- Jedna z definicji mówi, że BI to: zbiór praktyk, metodyk, narzędzi I technologii informatycznych, służących zbieraniu i integrowaniu danych w celu dostarczania informacji i wiedzy właściwym osobom, we właściwym miejscu oraz we właściwym czasie

3

BI - historia

- Systemy BI to ewaluacja z rozwiązań klasy DSS, EIS, czy MIS. Pojęcia te są ze sobą mocno związane i trudno zaznaczyć jasne granice pomiędzy nimi.
- Najstarszym z pojęć jest DSS, czyli **Decision Support Systems** (Systemy Wspomagania Decyzji, SWD).
  - *Pojęcie to ma swoje początki w badaniach z przełomu lat 50-tych i 60-tych ubiegłego stulecia, zostało zdefiniowane na początku lat 70-tych, a największą popularność zdobyło w latach 80-tych.*
  - *Odnosi się do systemów informatycznych wspierających podejmowanie decyzji.*
    - Na przestrzeni czasu DSS przyjęło nazywać się systemy, które pozwalały wykorzystywać dane, informacje i wiedzę do rozwiązywania słabo ustrukturalizowanych problemów decyzyjnych.

4

EIS i MIS

- Systemy informowania kierownictwa EIS (Executive Information Systems),
  - *Były efektem wykorzystania DSS poprzez wysoko wyspecjalizowanych analityków którzy opracowywali parametryzowane analizy i raporty, udostępniane wyższej kadrze kierowniczej, czy członkom zarządów poprzez*
  - *największą popularność osiągnęły zaczynając od połowy do końca lat 80-tych.*
- Mianem MIS, czyli Management Information Systems, przyjęło się nazywać szeroką grupę systemów, z DSS i EIS włącznie, które służą do wsparcia zarządzania poprzez analizę danych.

5

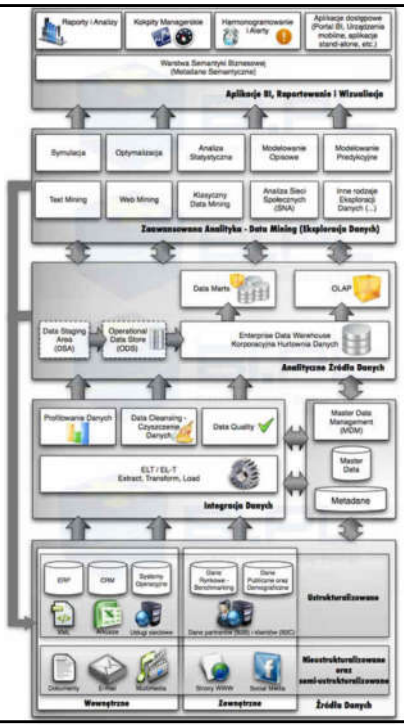
Businness inteligence

- Dość dobrze sens BI oddaje jeden z popularnych sloganów marketingowych, mówiących że BI to „**dostarczanie właściwej informacji, właściwym osobom we właściwym czasie**”
- Najczęściej kojarzone jest Hurtowniami Danych (Data Warehouse) oraz systemami analityczno-raportującymi, jednak zakres BI jest dużo szerszy.

6

BI - dzisiaj

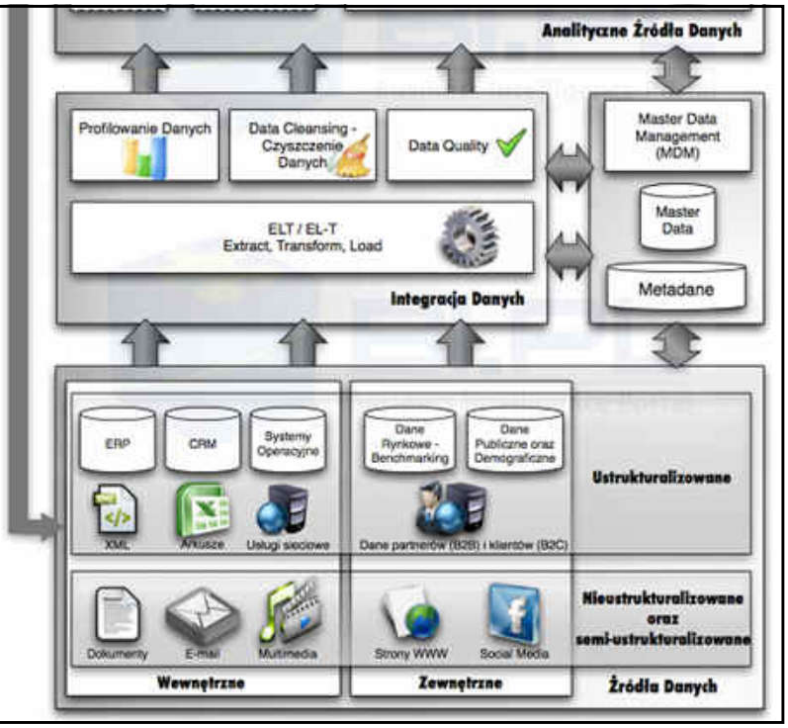
- Lata 90-te to początki kompleksowych rozwiązań analitycznych, opartych o hurtownie danych i zintegrowane modele korporacyjne, przetwarzanie analityczne w czasie rzeczywistym (OLAP) wraz z aplikacjami służącymi do dostępu do tych danych i ich analizy, zwane Business Intelligence.



7

BI - dzisiaj

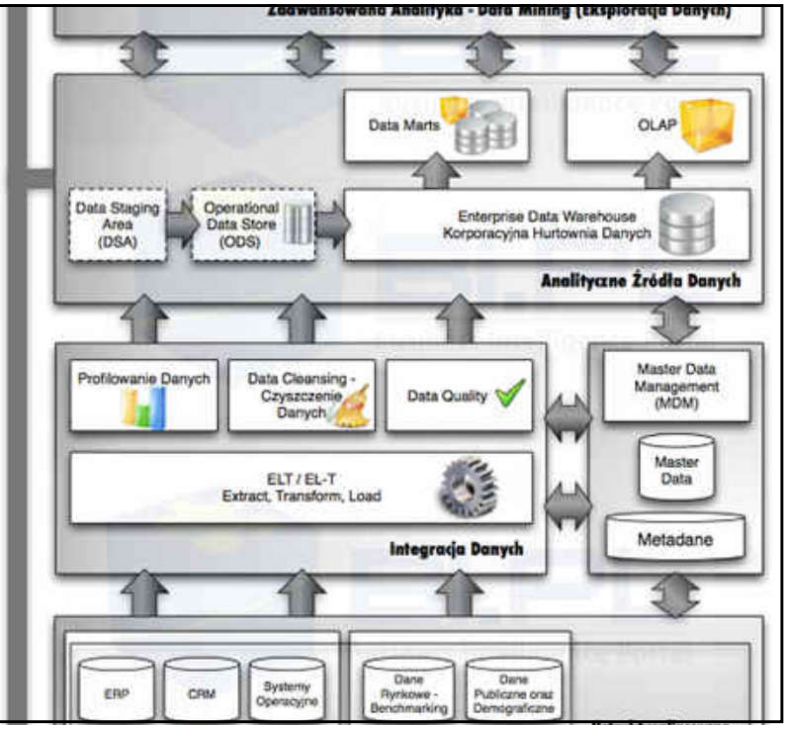
- Lata 90-te to początki kompleksowych rozwiązań analitycznych, opartych o hurtownie danych i zintegrowane modele korporacyjne, przetwarzanie analityczne w czasie rzeczywistym (OLAP) wraz z aplikacjami służącymi do dostępu do tych danych i ich analizy, zwane Business Intelligence.



8

BI - dzisiaj

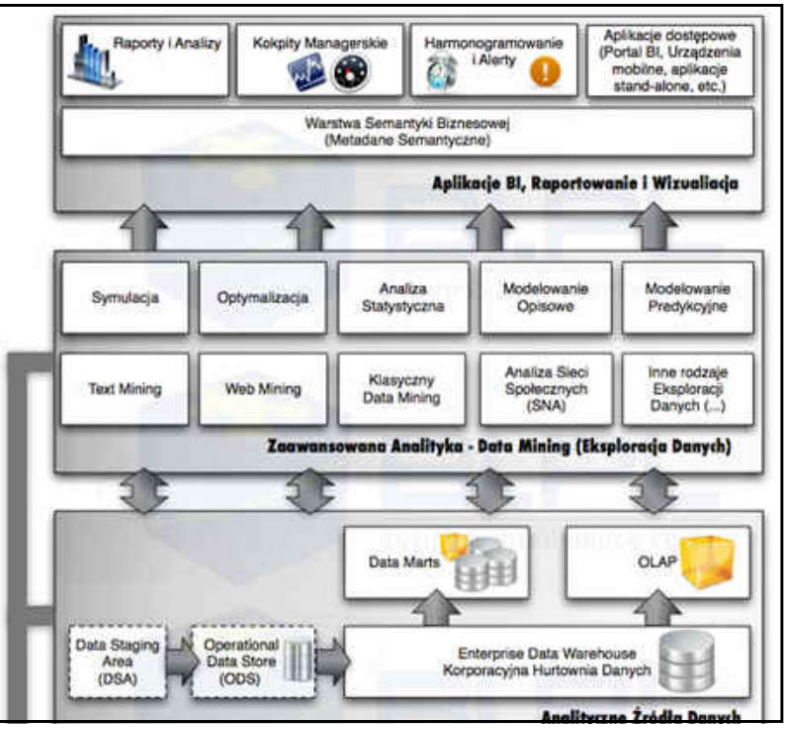
- Lata 90-te to początki kompleksowych rozwiązań analitycznych, opartych o hurtownie danych i zintegrowane modele korporacyjne, przetwarzanie analityczne w czasie rzeczywistym (OLAP) wraz z aplikacjami służącymi do dostępu do tych danych i ich analizy, zwane Business Intelligence.



9

BI - dzisiaj

- Lata 90-te to początki kompleksowych rozwiązań analitycznych, opartych o hurtownie danych i zintegrowane modele korporacyjne, przetwarzanie analityczne w czasie rzeczywistym (OLAP) wraz z aplikacjami służącymi do dostępu do tych danych i ich analizy, zwane Business Intelligence.



10

OLAP a OLTP

- OLTP - On Line Transaction Processing *transakcyjne/operacyjne*
- OLAP – OnLine Analytical Processing *analityczne*

11

Systemy OLTP

- OLTP - On Line Transaction Processing – każdy biznesowy system informatyczny pracujący w sposób transakcyjny
- OLTP zwane są często operacyjnymi bazami danych
- przede wszystkim mają w bezpieczny sposób gromadzić bieżące dane (jeżeli też historyczne, to niezbyt „odległe")
- kompleksowe zabezpieczenia przed: utratą danych, utratą spójności danych, nieautoryzowanym dostępem
- zwykle przetwarzają wielką liczbę stosunkowo krótkich transakcji
  - *INSERT, UPDATE, DELETE*
- wielodostęp (często w reżimie 24 –7–365)
  - *wymagany szybki dostęp do bieżących danych*
  - *dla wszystkich pracujących w danej chwili użytkowników*
- też praca w trybie wsadowym (np. nocne aktualizacje danych)
- najczęściej bazują na klasycznych relacyjnych bazy danych
- zwykle wysoce znormalizowane (przynajmniej 3 PN)

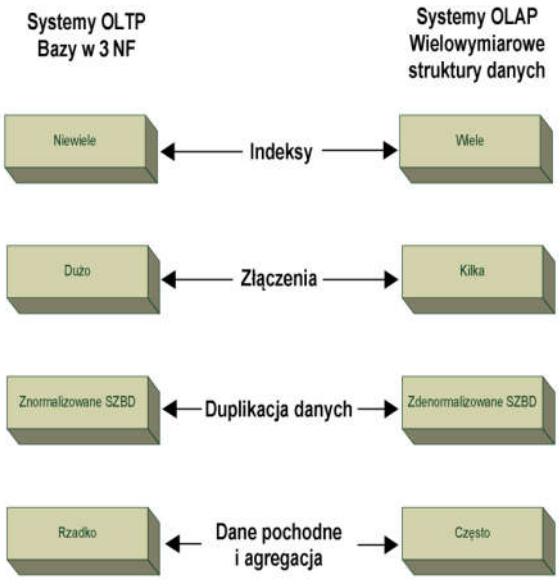
12

Systemy OLAP

- OLAP – OnLine Analytical Processing
  - *OLAP zwane są często analitycznymi bazami danych*
  - *tworzone przede wszystkim w celu efektywnego analizowania wielkich ilości danych*
    - podsumowania, raportowanie, prognozowanie, ocena, itp.
  - *gromadzą w zasadzie dane tylko historyczne*
    - zmiany w danych wyłącznie sporadyczne (np. usuwanie zauważonych błędów)
    - dane są często uzupełniane ale bardzo rzadko kasowane
    - głównie odczyt danych (SELECT)
    - niewielka liczba ale za to długich transakcji
  - *najczęściej bazują na **hurtowniach danych***
    - zwykle słabo znormalizowane, dane silnie zagregowane
  - *umiarkowany wielodostęp*

13

Struktura OLAP vs. OLTP



14

Cechy OLAP vs. OLTP

Element / cecha	OLAP	OLTP
Źródło danych	Dane skonsolidowane, zwykle na bazie innych systemów OLTP	Dane operacyjne na bazie typowych baz relacyjnych
Zakres czasowy	Dane wieloletnie	Dane z ostatnich tygodni, miesięcy
Cel posiadania	Analizy, planowanie, wspieranie w podejmowaniu decyzji biznesowych	Bieżące przetwarzanie danych zapewniające właściwe działanie przedsiębiorstwa
Dla kogo	Kadra zarządzająca	Pracownicy niższego szczebla
Wykonywane operacje	Cykliczne odświeżanie danych (zwykle wstawianie, rzadko kasowanie)	Dużo operacji wstawiania, modyfikacji i kasowania danych
Rodzaj zapytań	Złożone zapytania, odwołujące się do danych zagregowanych	Dużo prostych zapytań odwołujących się do wielu tabel i widoków relacyjnych
Czas odpowiedzi	Może być bardzo długi (nawet godziny)	Bardzo szybki (maksymalnie kilka sekund, lepiej mniej)
Wymagane zasoby	Wielkie (dużo danych wstępnie zagregowanych, dużo danych historycznych, dużo indeksów)	Relatywnie małe (gdy nie przechowujemy zbyt dużo danych historycznych)
Model danych	Dane zdenormalizowane, typowe dla HD układy (gwiazda, płatek śniegu)	Relacyjny (mocno znormalizowany), wiele tabel, widoków

15

Business intelligence

- Business intelligence
  - *proces przekształcania danych w informacje, a informacji w wiedzę, która może być wykorzystana do zwiększenia konkurencyjności przedsiębiorstwa*
  - *kombinację architektury systemu, aplikacji oraz baz danych, które razem umożliwiają prowadzone w czasie rzeczywistym analizy i przekształcenia, dostarczające potrzebną informację i wiedzę biznesowi*

16

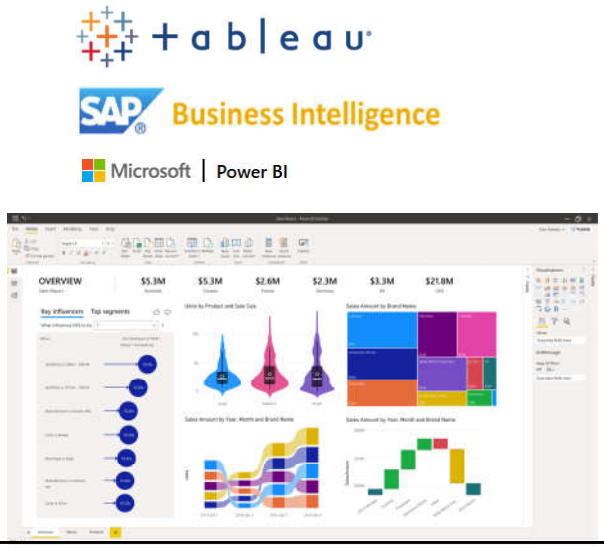
Business intelligence

- Od strony technicznej:
  - *Efektywne eksploatowanie narzędzi BI jest uzależnione od utworzenia hurtowni danych,*
  - *pozwala to na ujednolicenie i powiązanie danych zgromadzonych z różnorodnych systemów informatycznych przedsiębiorstwa.*
  - *Utworzenie hurtowni danych zwalnia systemy transakcyjne od tworzenia raportów i umożliwia równoczesne korzystanie z różnych systemów BI.*

17

Narzędzia Business intelligence

- Przykładowe popularne narzędzia BI
  - *Tableau: Business Intelligence*
  - *Microsoft Power BI*
  - *SAP: Business Intelligence*



18

Bibliografia

- Chris Todman: Projektowanie hurtowni danych, ISBN 83-204-2790-8 WNT, 2011
- Rumiński Jacek: Wprowadzenie do hurtowni i eksploracji danych, Wyd. Politechniki Gdańskiej, 2015,
- Współczesne problemy systemów baz danych Cz. 1." red. Włodzimierz Stanisławski, Skrypty Politechniki Opolskiej ; nr 27. , 2005. - 210 s..
- Vidette Poe, Patricia Klauer, Stephen Brobst: Tworzenie hurtowni danych, ISBN 83-204-2435-6, WNT 2000
- Czapla Krystyna: Podstawy projektowania i języka SQL baz danych, ISBN: 978-83-283-1296-8, Helion 2015 3.
- Adam Pelikant: Hurtownie danych. Od przetwarzania analitycznego do raportowania. ISBN: 978-83-246-5422-2, Helion 2011
- Sam Alapati, Darl Kuhn, Bill Padfield: Oracle Database 12c. Problemy i rozwiązania, ISBN: 978-83-246-9801-1, Helion 2014

19

Dziękuję za uwagę

- b.ruszczak@po.edu.pl

20