Pomorska Fundacja
Inicjatyw Gospodarczych

Relacyjne i nierelacyjne bazy danych cz.5

Michał Szymański

Agenda

Dowiem się:

- Jakie mechanizmy autoryzacji oferują bazy danych

Autoryzacja

O czym będziemy mówić

- Jak spowodować, żeby użytkownik miał dostęp do wybranych danych
- Jak spowodować, żeby użytkownik mógł modyfikować tylko wybrane dane

Autoryzacja

Bazujemy na

- Użytkownikach
- Uprawnieniach

Autoryzacja - uprawnienia

Uprawnienia

- SELECT
- INSERT
- UPDATE
- DELETE
- CREATE

Autoryzacja – przykłady

```
DELETE FROM  kontrola.straznik WHERE id IN (
   SELECT straznik_id FROM kontrola.przyznane_nagrody
   WHERE  nazwa=’Nagana’)
```

Musi mieć uprawnienia do:

- Kontrola.straznik - DELETE, SELECT(id)
- Kontrola.przyznane – SELECT(straznik_id, nazwa)

Autoryzacja – GRANT

Twórca danego bytu jest domyślnie właścicielem.
Właściciel ma wszystkie uprawnienia i może nadawać uprawnienia innym.

GRANT <uprawnienia> ON <obiekt> TO <użykownicy/role> [WITH GRANT OPTION]

GRANT INSERT ON pasazer TO 'u1a'@'localhost' ;
GRANT ALL PRIVILEGES ON pasazer TO 'u1a'@'localhost' ;

Autoryzacja – REVOKE

REVOKE <uprawnienia> ON <obiekt> FROM <użytkownicy/role>

Przykład:
REVOKE SELECT ON straznicy FROM 'u1c'@'lobcalhost'

Autoryzacja – rola/konto

Zakładanie użytkownika

CREATE USER 'u1c'@'lobcalhost' IDENTIFIED BY 'abc';

Usuwanie użytkownika:

DROP USER 'u1c'@'lobcalhost' ;

MySQL vs reszta świata

MySQL – zalety i wady

Zalety

- Darmowy
- Dostępny na każdym hostingu
- Różne silniki bazodanowe

Wady

- Optymalizator DB mniej wydajny przy bardziej skomplikowanych bazach danych niż PostgreSQL czy Oracle
- Mniej tranzakcyjny niż MySQL
- Mniej przyjemne procedury bazodanowe

NoSQL

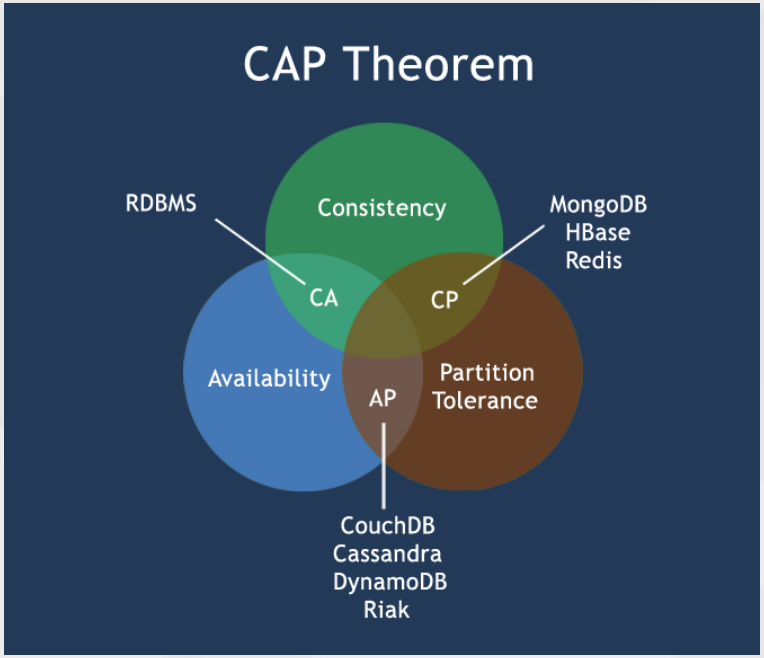
NoSQL – dla kogo..

- Skalowalność
- Elastyczność w przechowywaniu danych
- Łatwiejsza w użytkowaniu
- Większa wydajność

..ale

•Brak standardu języka dla NoSQL

•Zwykle nie są to bazy transakcyjne



- Spójność – otrzymamy najnowszą wersje danych lub błąd.
- Dostępność – każde zapytanie dostanie odpowiedź (bez błędu) nie koniecznie najnowsze dane.
- Part.tolerance – oznacza, że całość systemu powinna działać, mimo, że są przerwy w komunikacji między węzłami.

NoSQL – typy baz danych

- Klucz wartość – Infinispan, Memcached, Terracota, Coherence
- Graph DB – Neo4J, Oracle NoSQL
- Document Store (XML/JSON)– MongoDB, CouchDB, Elasticsearch,CasandraDB
- MapReduce - Hadoop

NoSQL – Key-Value

Klucz	Wartość
GD1234	Jan Kowalski
GD3234	Piotr Nowak
GD2344	Julian Tuwim
…	…

NoSQL – Key-Value

```
// Add a entry cache.put("key", "value");

// Validate the entry is now in the cache
assertEqual(1, cache.size());
assertTrue(cache.containsKey("key"));

// Remove the entry from the cache Object v =
cache.remove("key");

// Validate the entry is no longer in the cache
assertEqual("value", v);
```

NoSQL – Document store MongoDB

Cechy MongoDB
- Zorientowana na dokument np. klient, obiekt sprzedaży
- Szukanie po polu, z użyciem reg-exp
- Możliwość indeksowania dokumentów
- Load balancing (sharding)

MapReduce: Dlaczego prędkość ma znaczenie..

1990

Średnia wielkość dysku	1370 MB
Prędkość odczytu	4.4 MB/s,
Czas odczytu całego dysku	5 min

2010

Średnia wielkość dysku	1 TB (747x większy)
Prędkość odczytu	100 MB/s (25x szybszy)
Czas odczytu całego dysku	**3 godz. (36x dłużej!!!)**

NoSQL – MapReduce, przykład

Znaleźć liczbę INT8 w zbiorze zawierającej 100 mld wartości w postaci cyfr INT8 w zbiorze tekstowym:

343443
454545
5776576
….

Szacowanie wielkości (najgorszy scenariusz)
Jedna liczba to 20 znaków + znak końca lini = 21
Czyli wszystkie liczby to 21 mld znaków = 2 100 000 000 000 bajtów a to 2.1TB (zakładając że kilobajt to 1000B)
Dobry dysk SATA 3 w trybie Burst read – 200MB/s czyli 0.0002 TB/s. (realia około ¼ tej prędkości)
Czyli przeczytanie całość zajmuje 2.1 / 0.002 = 10 500 s = 2.91 godziny (a bardziej realne 12 godzin) !

Nie wygląda to dobrze ☹

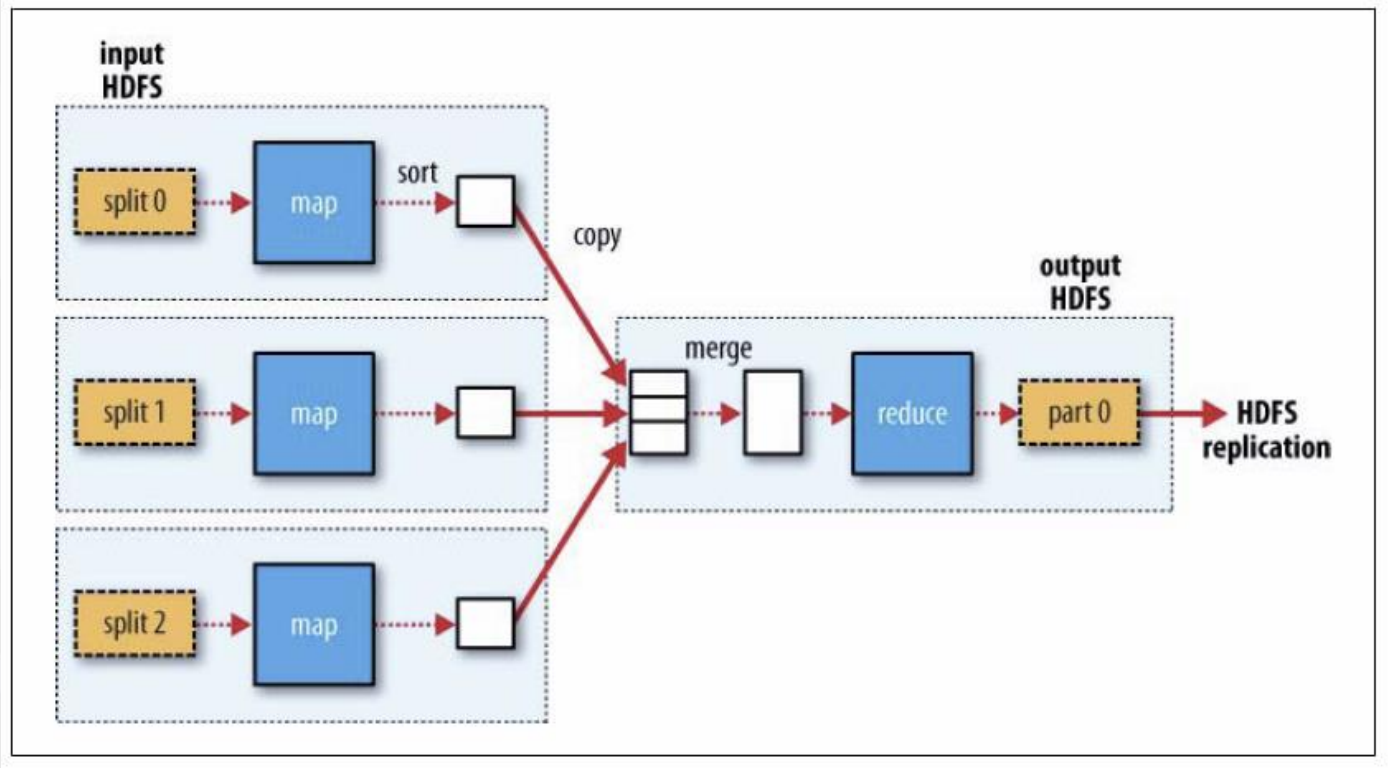
NoSQL MapReduce przykład

Rozwiązanie – przechowywać plik pocięty na kawałki na 1000 serwerach i analizowanie go w częściach !

Algorytm:
•Na każdej części znajdujemy liczbę
•Z wszystkich znalezionych liczb składamy końcową listę

Czas działania programu skracamy do 10 sekund..

NoSQL - MapReduce



NoSQL – MapReduce

Rozwiązanie

Rozłożyć dane na setki dysków i czytać równocześnie.

Problem - więcej elementów mniejsza niezawodność jak zabezpieczyć dane ?

Rozwiązanie

Stosować redundancje w przechowywaniu danych

To realizują systemy plików takie jak Hadoop Distributed Filesystem HDFS

NoSQL MapReduce porównanie do RDBMS

	Traditional RDBMS	MapReduce
Data size	Gigabytes	Petabytes
Access	Interactive and batch	Batch
Updates	Read and write many times	Write once, read many times
Structure	Static schema	Dynamic schema
Integrity	High	Low
Scaling	Nonlinear	Linear

MongoDB

- Open Source
- Baza danych zorientowana na przechowywanie dokumentów
- Dokumenty przechowywane w postaci JSON
- Wyszukiwanie po polach z użyciem wyrażeń regularnych
- Możliwość nakładania indeksów
- Wbudowany mechanizm replikacji
- Balansowanie ruchu (z użyciem sharding)
- Można użyć Map-Reduce do operacji na bazie

MongoDB – przykład dokumentu

```
{
    title: 'MongoDB Overview',
    description: 'MongoDB is no sql database',
    by: 'tutorials point',
    url: 'http://www.tutorialspoint.com',
    tags: ['mongodb', 'database', 'NoSQL'],
    likes: 100
 }
```

MongoDB – przykład wyszukiwania

Operation	Syntax	Example	RDBMS Equivalent
Equality	{<key>: <value>}	db.mycol.find({"by":"tutorials point"}).pretty()	where by = 'tutorials point'
Less Than	{<key>: {$lt: <value>}}	db.mycol.find({"likes": {$lt:50}}).pretty()	where likes < 50
Less Than Equals	{<key>: {$lte: <value>}}	db.mycol.find({"likes": {$lte:50}}).pretty()	where likes <= 50
Greater Than	{<key>: {$gt: <value>}}	db.mycol.find({"likes": {$gt:50}}).pretty()	where likes > 50
Greater Than Equals	{<key>: {$gte: <value>}}	db.mycol.find({"likes": {$gte:50}}).pretty()	where likes >= 50
Not Equals	{<key>: {$ne: <value>}}	db.mycol.find({"likes": {$ne:50}}).pretty()	where likes != 50

Zadanie końcowe

DESER

(czyli to czego nie powiedziałem a warto powiedzieć)

Podział według J.Berkus

Web Application

- DB mniejsze niż RAM
- 90% zapytań jendowierszowych

Online Transaction Processing (OLTP)

- DB nieco większa niż RAM około 1TB
- 20-40% niewielkie zapisy ale pojawiają się większe transakcje

Data Warehouse / Business Intelligence

- Wielkie / olbrzymie transakcje (100GB to 100TB)
- Duże i skomplikowane zapyania
- Duże operacje wczytywania danych

Real-Time

- DB mniejsze niż RAM, małe zapytania
- Ustalony czas odpowiedzi

Transakcje przykład z życia

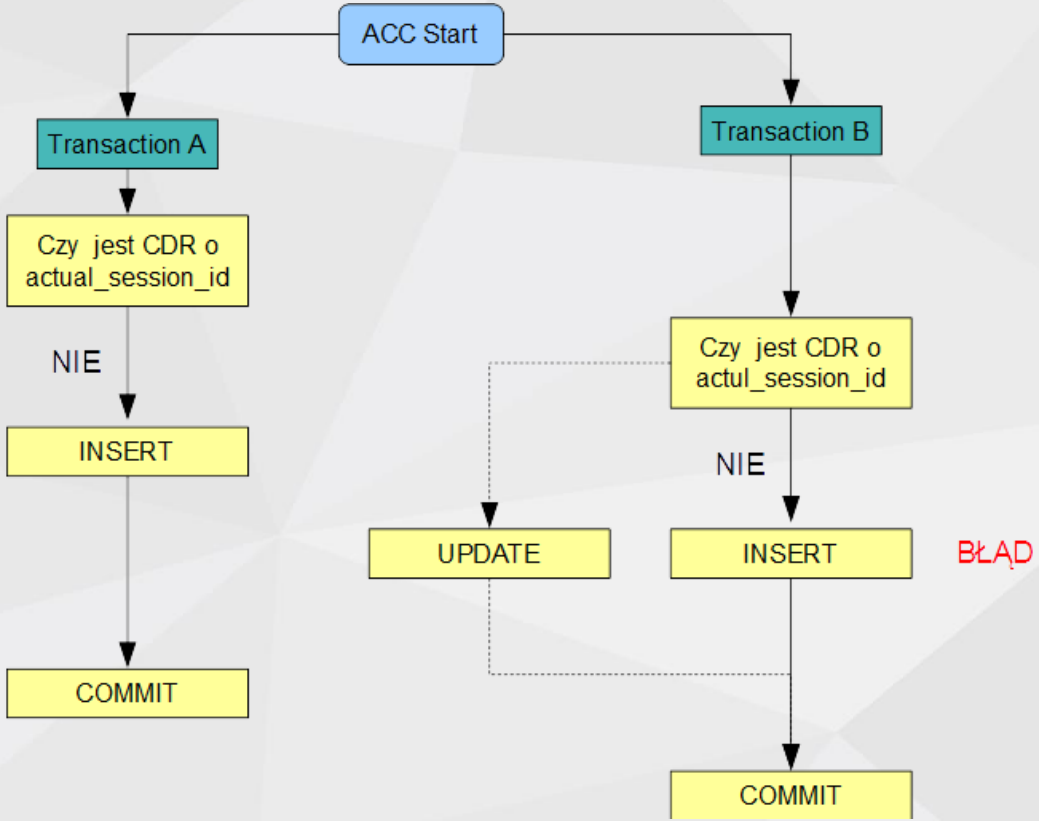
Problem

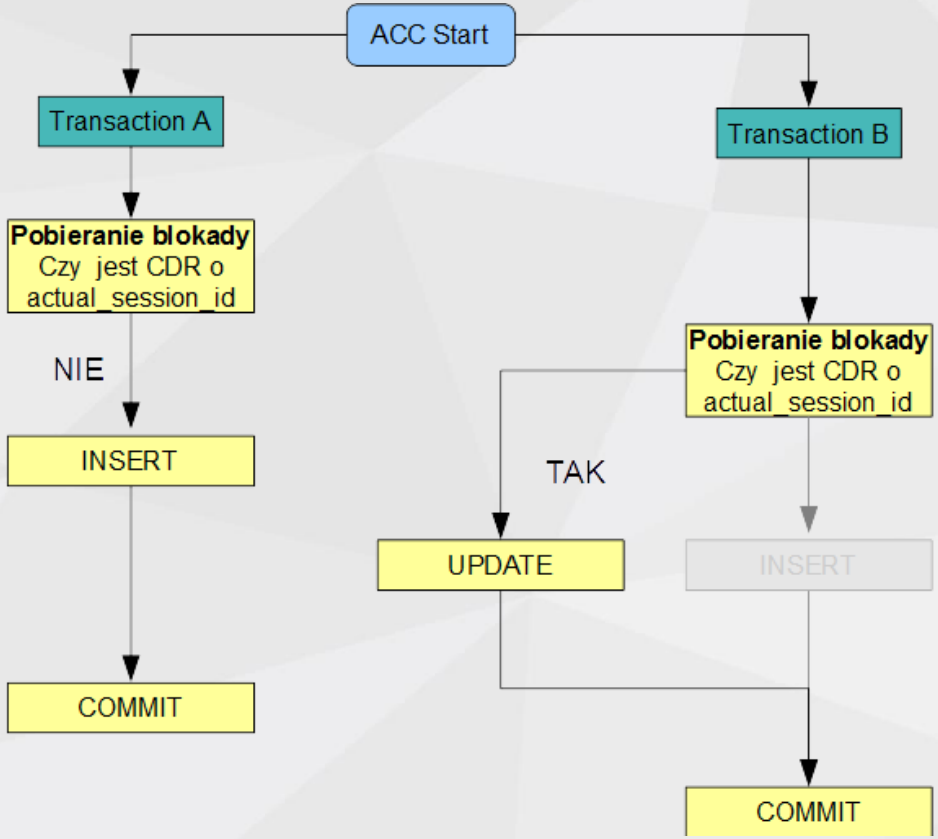
W systemie telco mamy węzły które 'składają' wiersze z połączeniami tzw.CDR. Komunikaty ACCStart / ACCStop które przychodzą do tych wezłów generowane są przez pakiety UDP. Mogą pojawić się retransmisje i dużo 'równoległych' komunikatów. Trzeba zabezpieczyć się przed dupliakatami.

ACCStart – informuje o rozpczęciu rozmowy

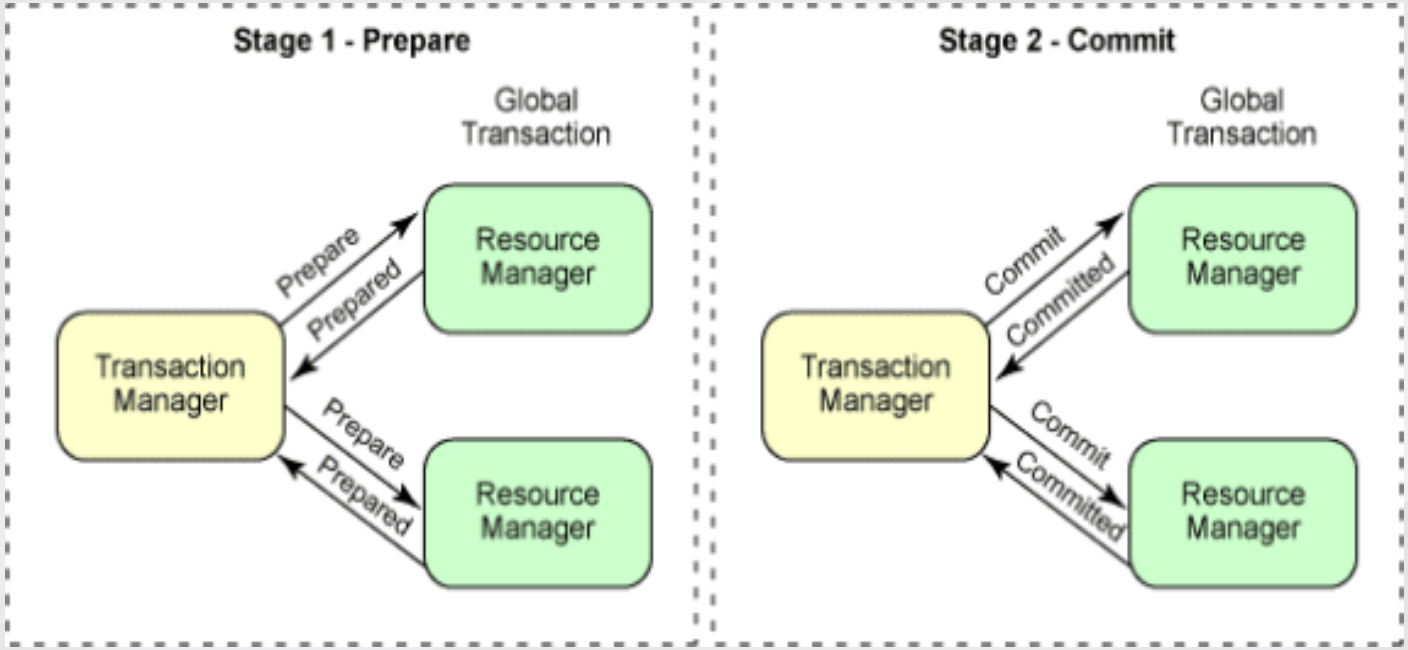
ACCStop – informuje o zakończeniu rozmowy

Co można zarobić żeby tą sytuacje naprawić?




ACC Start
Transaction A
Transaction B
Pobieranie blokady
Czy jest CDR o
actual_session_id
NIE
INSERT
COMMIT
Pobieranie blokady
Czy jest CDR o
actual_session_id
TAK
UPDATE
INSERT
COMMIT

Two phase commit



Transakcje luźne uwagi

- Jeśli nie ma takiej potrzeby to NIE stosować transakcji (zwłaszcza 2PC)
- Jeśli jesteśmy zmuszeni to ograniczać użycie
- Czasami nie mamy wyboru i nie jesteśmy w stanie użyć np. WebService tak więc nie ma problemu
- Nie warto pisać własnych Transaction Manager'ów zwłaszcza dla transakcji rozproszonych
- Zastanowić się nad tym co oznacza procedura ROLLBACK w systemie w którym nie ma transakcji

Optymalizacja

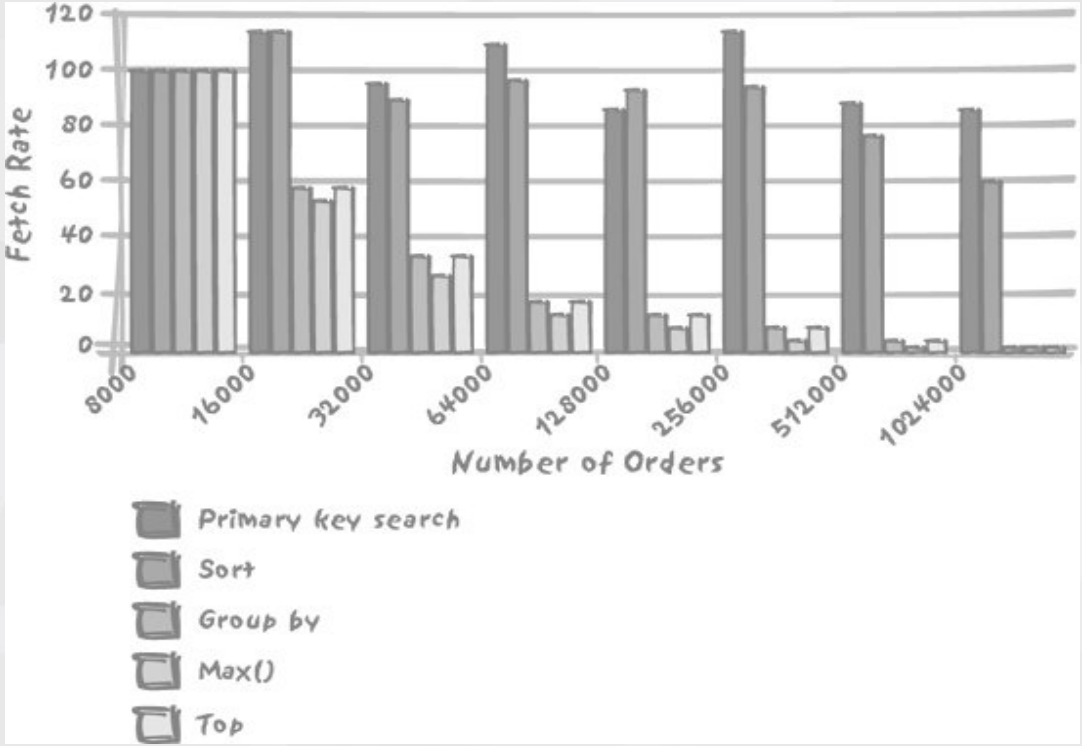
Założenia :

- Będziemy mówić o dużych bazach danych >1GB
- Gdzie mamy >200TPS

Gdzie leży problem??

- Co się zmieniło od momentu gdy wszystko działało wystarczająco dobrze?
- Czy nie jest to problem związany z niewłaściwym użyciem bazy?
- Czy nie jest to problem sprzętowy?
- Optymalizacje rozpocznij jak jesteś pewien, że jest faktycznie potrzebna.
- Opracuj metody pozwalające mierzyć wydajność rozwiązania.
- Zastanów się czy nie jest to 'błąd biznesowy'

Kiedy pojawią się problemy?



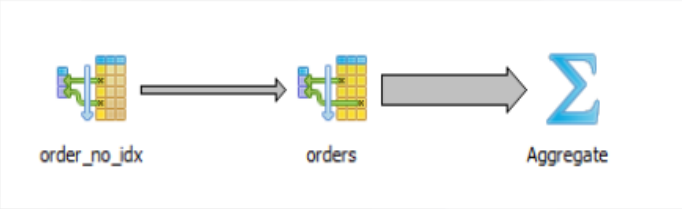
Ilość wierszy	Problemy
0-200tyś (0-100MB)	Nie ma problemów, nawet brak indeksów nie jest problemem. Rozwiązanie „siłowe” rozwiąże każdy problem.
200tyś–20mln (100MB-1GB)	Pojawiają się pierwsze problemy z wyszukiwaniem danych , masowymi updatami i delet'ami. Zadania raportujące mogą trwać długo. Dodanie indeksów poprawia wydajność.
>20mln (>1GB)	Okazuje się że dodanie indeksów nie rozwiązuje problemów, wymagane są zmiany w algorytmach. Masowe operacje typu update / delete trwają bardzo długo i muszę być podzielone na etapy. Eksport i import bazy mierzymy w godzinach.

Użycie indeksów

Suma wartości dla wierszy spełniającej kryterium (range scan)

```
SELECT SUM(item_id) FROM orders WHERE  order_no<200
```

Plan




```
SELECT SUM(item_id) FROM orders WHERE  order_no>200
```

Plan

Czy jesteś sprytniejszy..

niż optymalizatora zapytań?

W większości przypadków...NIE

- algorytmy pisane są przez specjalistów od DB
- optymalizator 'zna' specyfikę danej bazy danych
- optymalizator bierze pod uwagę aktualną ilość i charakterystykę danych w tablicach

Ograniczenia optymalizatora

- optymalizator opiera swoje decyzje na danych znalezionych w bazie – statystyki oraz informacje deklaratywne.
- optymalizator działa najwydajniej gdzie można zastosować przekształcenia matematyczne prowadzące do skonstruowania operacji semantycznie równoważnych
- działanie optymalizatora wpływa na czas wykonania zapytania
- optymalizator usprawnia wykonanie pojedynczego zapytania

Pięć czynników skutecznego SQL

- całkowita ilość danych z których ma być pobrany wynik
- kryterium zastosowane do zdefiniowania zbioru wynikowego
- rozmiar zbioru wynikowego
- liczba tabel przetwarzanych w celu uzyskania oczekiwanego zbioru wynikowego
- liczba innych użytkowników modyfikujących równolegle te same dane

Co wpływa na wydajność - sprzęt

Najważniejsze czynniki wpływające na wydajność DB:

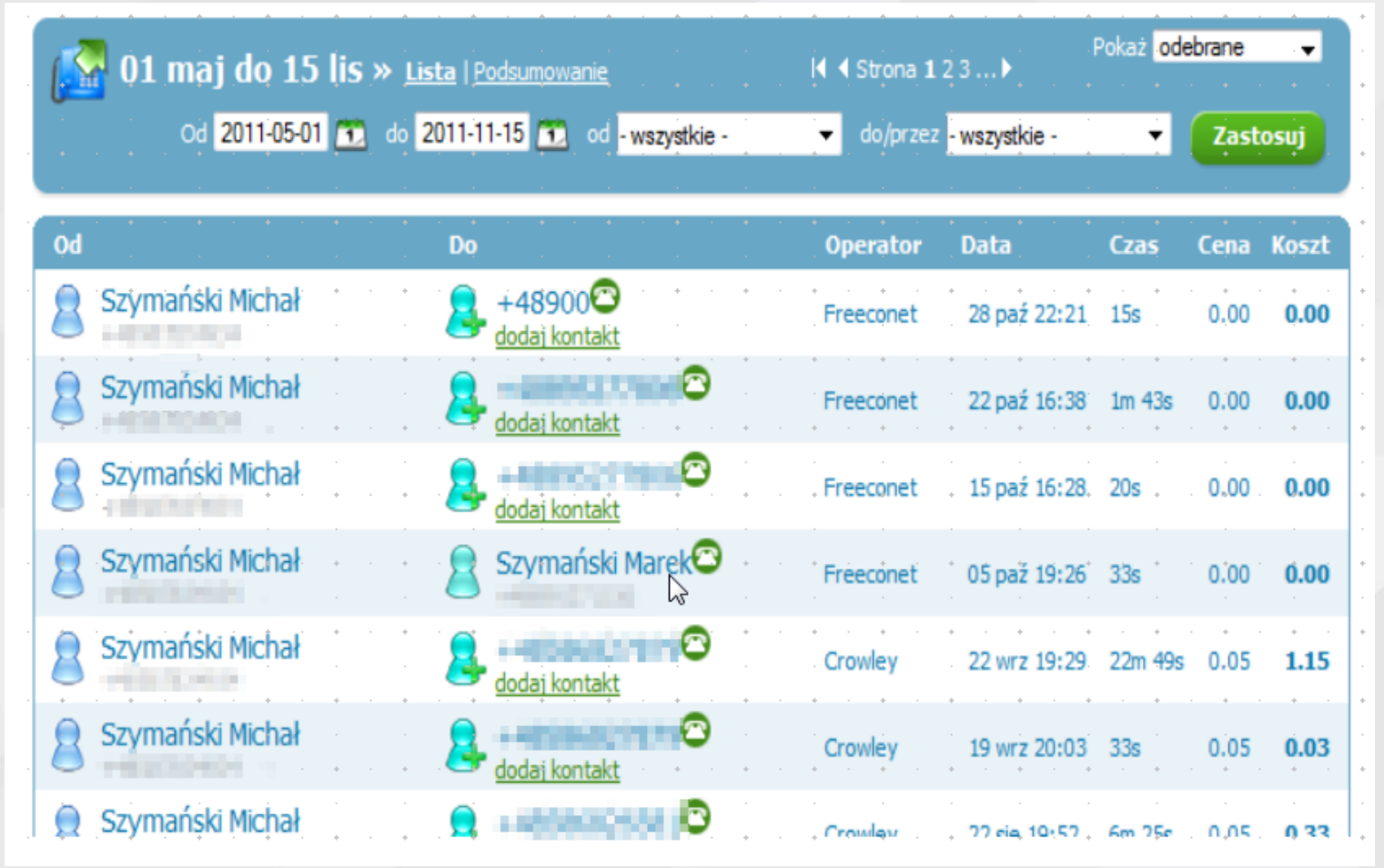
- RAM – jeśli baza uruchomiona jest na systemie 64 bitowym raczej nie napotkamy problemów
- Dyski – rozmiar przestrzeni nie jest problemem, zwykle bazy nie przekraczają 1T. Problemem może być transfer i czas dostępu.
- CPU – zwiększanie ilości rdzeni jest bardzo kosztowne, wydajność nie rośnie liniowo z ilością rdzeni.
- Sieć – zwykle 1Gbit/s albo Infiniband od 2.5Gb/s (chyba że mamy zdalnych klientów)

Historia jednego zapytania

Problem: Pobrać dane o połączeniach dla jednego konta serwisu przy zadanych kryteriach wyszukiwania i zadanym kryterium sortowania. Wyniki są prezentowane w stronach po 10-100 wyników

Zastana sytuacja:

- ok. 70 mln wierszy w tablicy CDR
- ok 200 tyś użytkowników
- wymagany czas odpowiedzi max.6s
- ilość kolumn w tabeli ok. 60
- jedno zapytanie mogło zwrócić nawet 200-300tyś. wierszy


01 maj do 15 lis » Lista | Podsumowanie
Strona 1 2 3 ...
Pokaż odebrane
Od 2011-05-01 do 2011-11-15 od - wszystkie - do/przez - wszystkie - Zastosuj
Od Do Operator Data Czas Cena Koszt
Szymański Michał +48900 dodaj kontakt Freeconet 28 paź 22:21 15s 0.00 0.00
Szymański Michał dodaj kontakt Freeconet 22 paź 16:38 1m 43s 0.00 0.00
Szymański Michał dodaj kontakt Freeconet 15 paź 16:28 20s 0.00 0.00
Szymański Michał Szymański Marek Freeconet 05 paź 19:26 33s 0.00 0.00
Szymański Michał dodaj kontakt Crowley 22 wrz 19:29 22m 49s 0.05 1.15
Szymański Michał dodaj kontakt Crowley 19 wrz 20:03 33s 0.05 0.03
Szymański Michał

Historia jednego zapytania

```
SELECT * FROM accounting.cdr LEFT JOIN crm.crm_account cat ON cat.id_crx_account=id_crx_account_to
 WHERE id_crx_account_from='528918' AND call_status='FINS' AND
       end_time_billing>='1970-01-01 01:00:01+01' AND end_time_billing<'2038-01-19 04:14:07+01'
ORDER BY end_time_billing DESC LIMIT 10 OFFSET 0
```

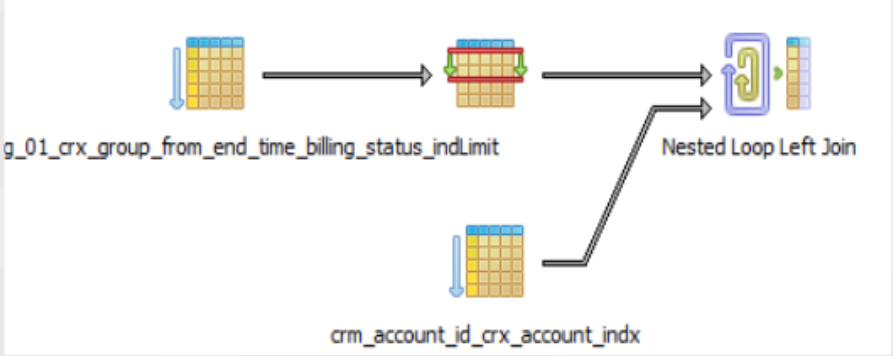
Zapytanie potrafiło trwać nawet parę minut , generować duży load na serwerze.

Podjęte kroki

- Wydzielenie tabeli CDR do bazy non-realtime
- Dodanie niezbędnych indeksów
- Dynamiczne tworzenie zapytania
- Zmiana podejścia do pobierania danych do zakładek
- Modyfikacja zapytania tak żeby LIMIT był stosowany na wcześniejszym etapie
- Partycjonowanie tabeli
- Ograniczonie liczby zwracanych kolumn

Historia jednego zapytania

```
SELECT
        ……
        (CASE WHEN id_crx_group_to IS distinct FROM 520993 THEN
                crm.find_contact_using_number_array (COALESCE(phone_number_to_pre, phone_number_to_org),520993)
        ELSE NULL END) AS contact_info_to,
        EXTRACT(EPOCH FROM start_time_invite)::INT8 AS start_time_call_ts
FROM
        (
                SELECT *
                FROM  accounting.cdr_og_01
                WHERE
                        1=1  AND
                        (id_crx_group_from='520993') AND
                        (id_crx_account_from='528918') AND
                        (call_status='FINS' ) AND
                        (end_time_billing>='1970-01-01 01:00:01+01') AND
                        (end_time_billing<'2038-01-19 04:14:07+01')
                ORDER BY end_time_billing DESC LIMIT  10 OFFSET 0
        )
        AS cdr_subset  LEFT JOIN crm.crm_account cat ON cat.id_crx_account=id_crx_account_to  AND cat.id_crm_group=520637
```



Optymalizator wynagradza tych, którzy pozostają w warstwie relacyjnej

Skalowalność

Scalability is the ability of a system, network, or process, to handle growing amount of work in a graceful manner or its ability to be enlarged to accommodate that growth.

A system whose performance improves after adding hardware, proportionally to the capacity added, is said to be a **scalable system.**

Scalability- the ability for a business or technology to accept increased volume without impacting the contribution margin (**=** revenue - variable costs)

To **scale horizontally** (or scale out) means to add more nodes to a system, such as adding a new computer to a distributed software application. An example might be scaling out from one Web server system to three.

Wikipedia

To **scale vertically** means to add resources to a single node in a system, typically involving the addition of CPUs or memory to a single computer.

Wikipedia

Skalowanie - DB

Skalowanie pionowe:

- Architektura wieloprocesorowa i wielordzeniowa
- Systemy 64bit – pamięć nie jest problemem
- Macierze dyskowe

Skalowanie poziome:

- Partycjonowanie przetwarzania
- Map-Reduce / Sharding

Skalowanie odczytów jest dużo łatwiejsze niż zapisów do DB

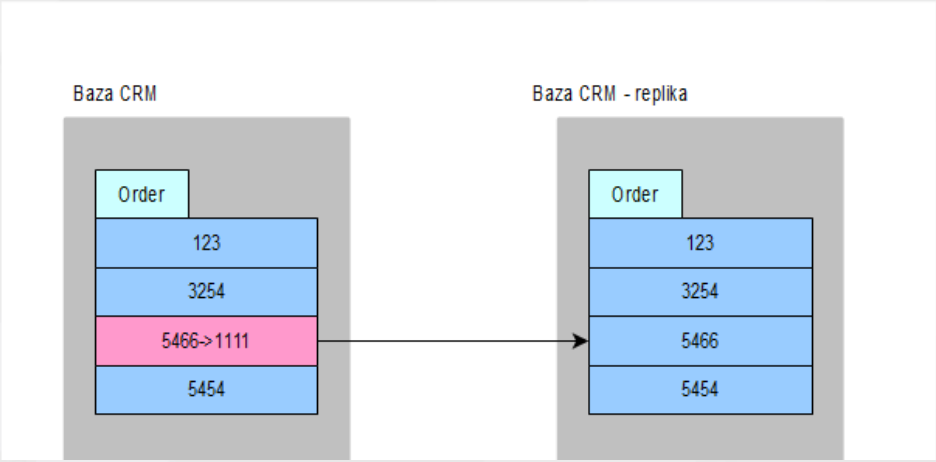
Replikacja

Typy replikacji:

- Row based
- Statement based
- WAL (Write-Ahead-Log) based

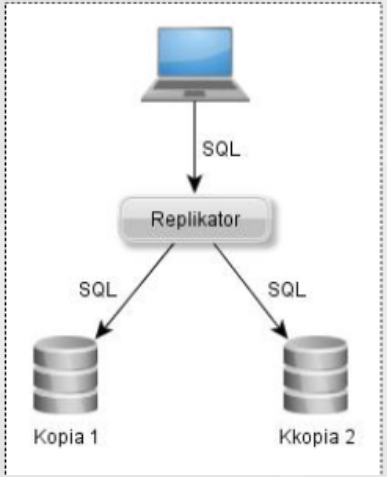
Replikacja – Row based

- Kopiowane są zmiany w wierszach
- Nie przenoszone są zmiany struktur
- Replikacja zwykle asynchroniczna



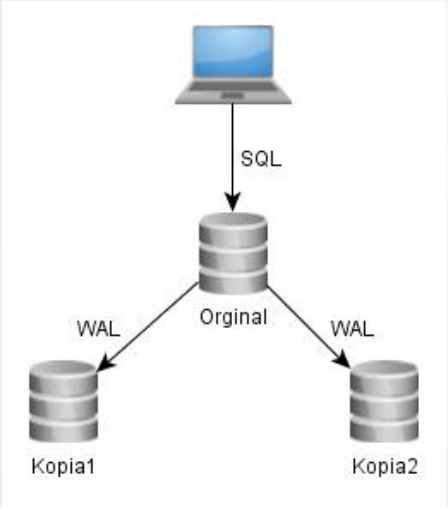
Replikacja – Statement Based

- Kopiowane są **wszystkie** operacje bazodanowe
- Operacje są dokonywane równolegle na 2 bazach



Replikacja – oparta o WAL

- Kopiowane są **wszystkie** operacje bazodanowe
- Kopia jest bazą Read-Only
- Operacja replikacji synchroniczna/asynchroniczna



Replikacja w MySQL

dd

Źródła

Książki

- “SQL. Sztuka programowania” - Stephany Faroult, Peter Robson
- “Refactoring SQL Application” - Stephany Faroult
- “Wysoko wydajny Postgres” – Gregory Smith

Sieć

- Dokumentacja Postgresa / MySQL
- http://www.w3schools.com/sql/default.asp
- Coursera “Introduction to Database” Stanford University