

A framed poster with a black border. Inside, a light gray rectangle contains the text 'COSMOSDB'. Above and below this rectangle are horizontal red lines. Below the bottom red line, the names 'PATRYK CEBRAT' and 'DAWID SZCZERBA' are printed in black.

COSMOSDB

PATRYK CEBRAT

DAWID SZCZERBA

WYZWANIA GLOBALNIE ROZPROSZONYCH BAZY DANYCH:

- Dla firmy bardzo trudnym zadaniem jest stworzyć własną, rozproszoną bazę danych.
Główną wadą rozproszenia danych jest znacznie większa złożoność systemu związana przede wszystkim z koordynacją działań pomiędzy stanowiskami:
- Większy koszt oprogramowania (system rozproszony jest znacznie trudniejszy w implementacji, więc mamy dodatkowe koszty) .
- Większe ryzyko błędów (stanowiska pracują równolegle – trudniej jest zagwarantować poprawność działania algorytmów w takim środowisku).



Ewolucja Cosmos DB

W roku 2010
Microsoft zaczął mieć coraz
bardziej poważne problemy wynikają
ce z wad SQL.

Ich własny serwer miał problemy
w obsłudze takich oprogramowań
jak Office czy Xbox.

Rozpoczęcie prac nad nowym
oprogramowaniem.

Rok 2015: Wydanie przez
Microsoft nowej bazy danych
nazwanej documentDB.

Wspieranie zapytań SQL na
dokumentach JSON.

2017:

Przemianowanie
DocumentDb na
CosmosDB.

Rozwiązanie
problemów globalnej
dystrybucji i
skalowalności.



Dlaczego cosmosDB?

Zarządzanie:

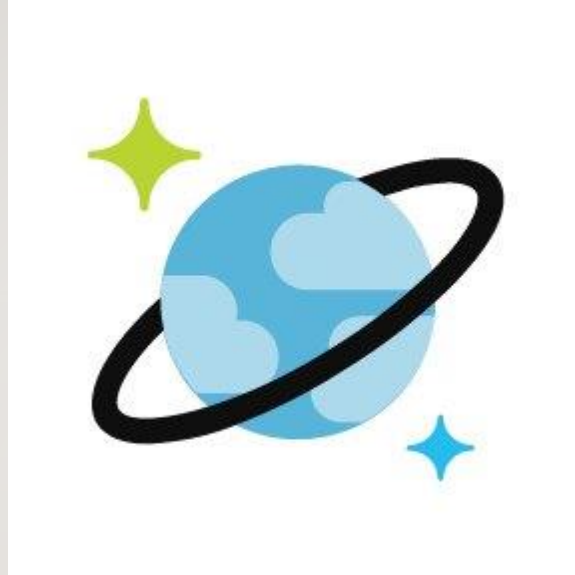
- Database as a service (DaaS).
- Bezserwerowa aplikacja.
- Brak kosztów operacyjnych.
- Zarządzanie schematem i indexem.

Dystrybucja globalna:

- Turnkey global distribution

Wspieranie wielu formatów i języków programowania:

- Wspieranie JSON, tabel, grafów, kolumnowy model danych
- Dostępny dla wielu technologii: Java, .NET, Node.js, JavaScript, itd.



Konfigurowanie spójności:

- Wspieranie 5 poziomów spójności.

Skalowalność:

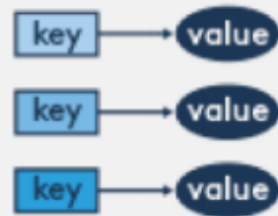
- Nielimitowana skalowalność dla magazynowania i przepustowości.

Wysoka dostępność, niezawodność i bezpieczeństwo:

- 99,999% SLA.
- ms czas oczekiwania.

NoSQL

Key-Value



Column-Family



Graph



Document



Table



Gremlin
 $G = (V, E)$



cassandra



mongoDB



SQL

SQL API

Ten rodzaj interfejsu bazy danych Azure Cosmos DB zapewnia możliwość manipulowania danymi przez użytkowników, którzy znają standardy zapytań języka SQL. Zasadniczo dane są przechowywane jako dokumenty JSON, ale możemy je przeszukiwać w prosty sposób za pomocą zapytań podobnych w składni do SQL. Komunikacja jest obsługiwana przez protokoły HTTP/HTTPS, za których pośrednictwem przesyłamy żądania do instancji bazy.

COLLECTION

```
Document 1:
{
  "id": "Nordcloudian1",
  "lastName": "Kowalski",
  "firstName": "Jan",
  "skills": [
    { "Azure": true },
    { "AWS": false }
  ],
  "employment": { "role": "CA", "country": "Poland", "city": "Poznan" },
  "isActive": true
}
Document 2:
{
  "id": "Nordcloudian2",
  "lastName": "Janiszewska",
  "firstName": "Mariola",
  "skills": [
    { "Azure": true },
    { "AWS": true }
  ],
  "employment": { "role": "Engineer", "country": "Germany", "city": "Berlin" },
  "isActive": true
}
Document 3:
{
  "id": "Nordcloudian3",
  "lastName": "Nowak",
  "firstName": "Janusz",
  "skills": [
    { "Azure": false },
    { "AWS": true }
  ],
  "employment": { "role": "DevOps", "country": "Finland", "city": "Helsinki" },
  "isActive": true
}
```

QUERY

```
SELECT *
FROM Nordcloudians n
WHERE n.id = "Nordcloudian1"
```

RESULTS

```
{
  "id": "Nordcloudian1",
  "lastName": "Kowalski",
  "firstName": "Jan",
  "skills": [
    { "Azure": true },
    { "AWS": false }
  ],
  "employment": { "role": "CA", "country": "Poland", "city": "Poznan" },
  "isActive": true
}
```

MongoDB, Cassandra, Gremlin, Table API

Istniejące instancje bazy można przenieść do platformy Azure Cosmos DB bez dużego nakładu pracy. Oba standardy są ze sobą kompatybilne. Po zakończeniu migracji danych do środowiska bazodanowego w Azure wystarczy w aplikacji zaktualizować łańcuch połączeniowy, aby przywrócić jej działanie z nowym źródłem danych.

Poniżej znajdują się dokładne informacje pochodzące z oficjalnej witryny Azure Cosmos DB, dotyczące dostępności poszczególnych interfejsów API dla najbardziej popularnych języków programowania.

	 Java	 .NET	 Node.js	 Python	 Gremlin	 Go	 Xamarin
SQL API	↗	↗	↗	↗			↗
Azure Cosmos DB's API for MongoDB	↗	↗	↗	↗		↗	↗
Gremlin API	↗	↗	↗	↗	↗		
Table API	↗	↗	↗	↗			
Cassandra API	↗	↗	↗	↗			

KRYTERIUM DECYZYJNE

	Core (SQL)	MongoDB	Cassandra	Azure Table	Gremlin
New projects being created from scratch	✓				
Existing MongoDB, Cassandra, Azure Table, or Gremlin data		✓	✓	✓	✓
Analysis of the relationships between data					✓
All other scenarios	✓				

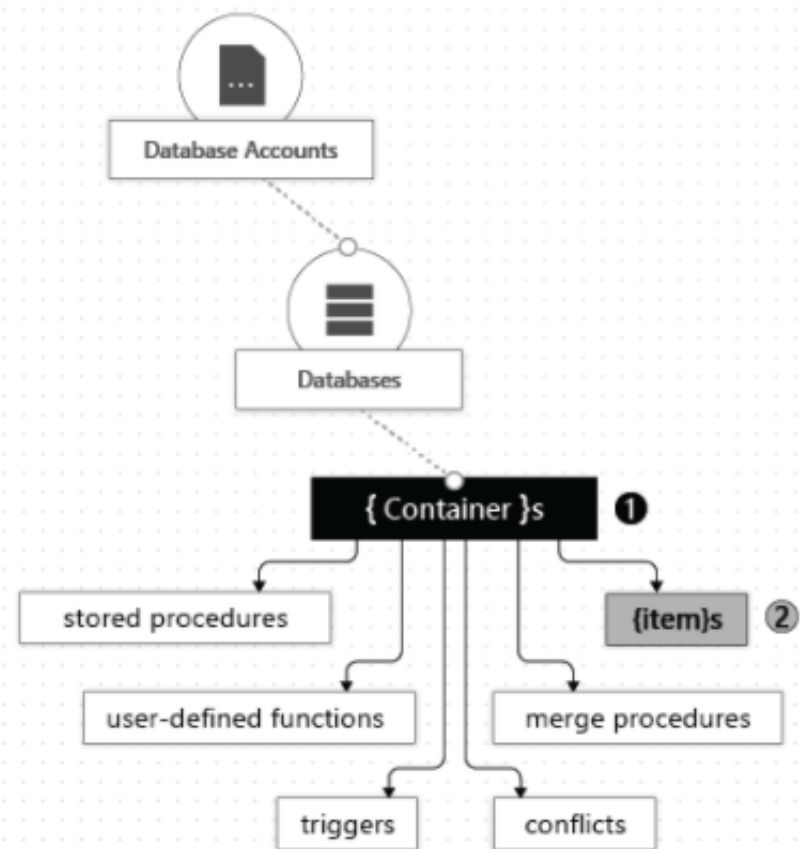
AZURE TABLE STORAGE VS COSMOS DB TABLE API

AZURE TABLE STORAGE

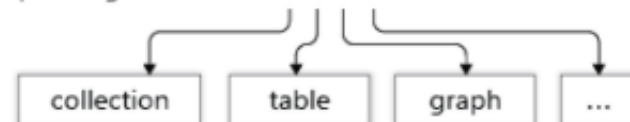
- Ograniczona replikacja geograficzna
- Wsparcie tylko dla primary key
- Mniejsza wydajność (ograniczona przepustowość, mniejszy czas oczekiwania)
- Brak poziomów spójności

COSMOSDB TABLE API

- Replikacja geograficzna w wybranych przez użytkownika regionach
- Wsparcie dla secondary index
- Lepsza wydajność (nielimitowana przepustowość, mniejszy czas oczekiwania)
- 5 poziomów spójności



1 { Container }'s
Depending on the Cosmos API, a container is realized as:



2 { item }'s
Depending on the Cosmos API, an item is realized as:



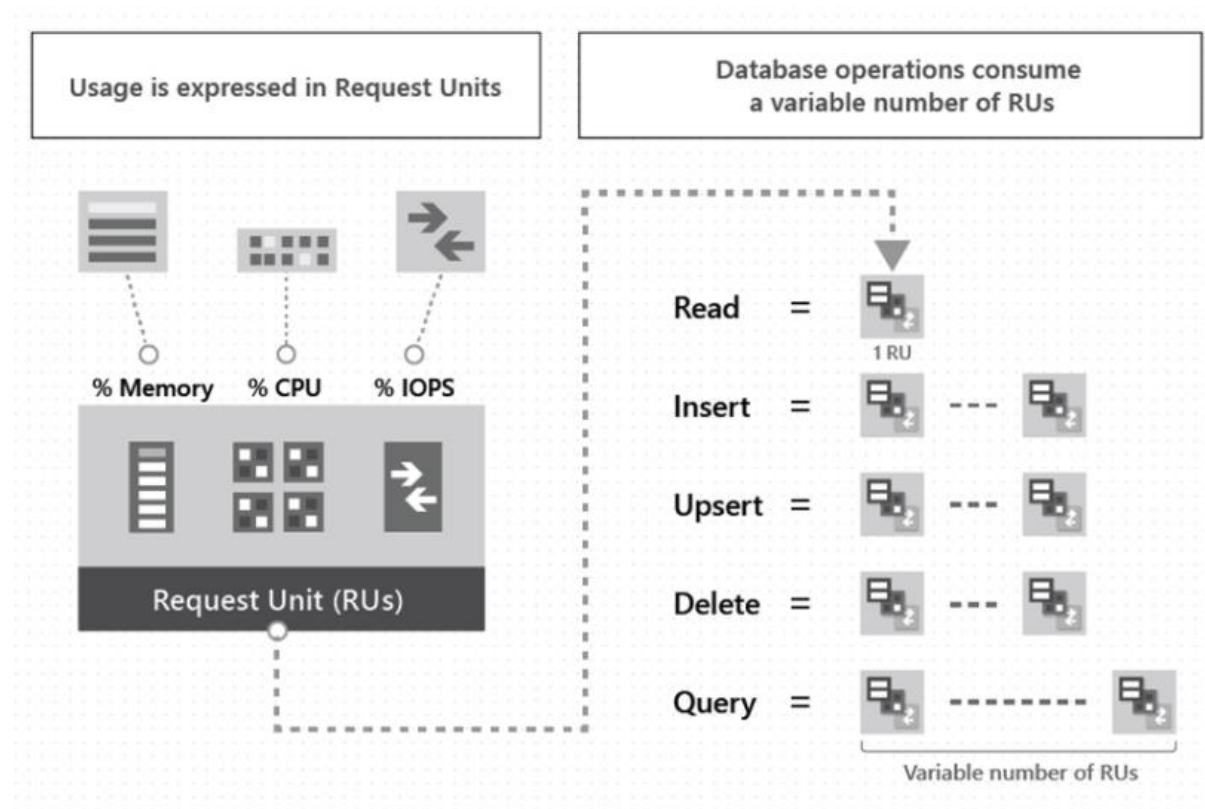
Azure Cosmos entity	SQL API	Cassandra API	MongoDB API	Gremlin API	Table API
Azure Cosmos database	Database	Keyspace	Database	Database	NA
Azure Cosmos container	Container	Table	Collection	Graph	Table
Azure Cosmos item	Document	Row	Document	Node or edge	Item

WYDAJNOŚĆ

Opóźnienie (Latency)
- Jak szybko
otrzymujemy
odpowiedź na dany
request?

Przepustowość
(Throughput) - Jak
wiele requestów
może być obsłużonych
w określonym czasie?

REQUESTS UNITS



REQUESTS UNITS

New Database New Collection Open Full Screen **New SQL Query** Feedback

SQL API

flights

departuredelays

Documents

Scale & Settings

Stored Procedures

User Defined Functions

Triggers

Documents Query 1

Execute Query Open Query From Disk

```
1 SELECT * FROM d WHERE d.origin = 'ABE'
```

Results **Query Stats**

METRIC	VALUE
Request Charge	46.180000000000001 RUs
Showing Results	1 - 100
Round Trips	1

* Container id ⓘ

B

* Partition key ⓘ

/id

☐ My partition key is larger than 100 bytes

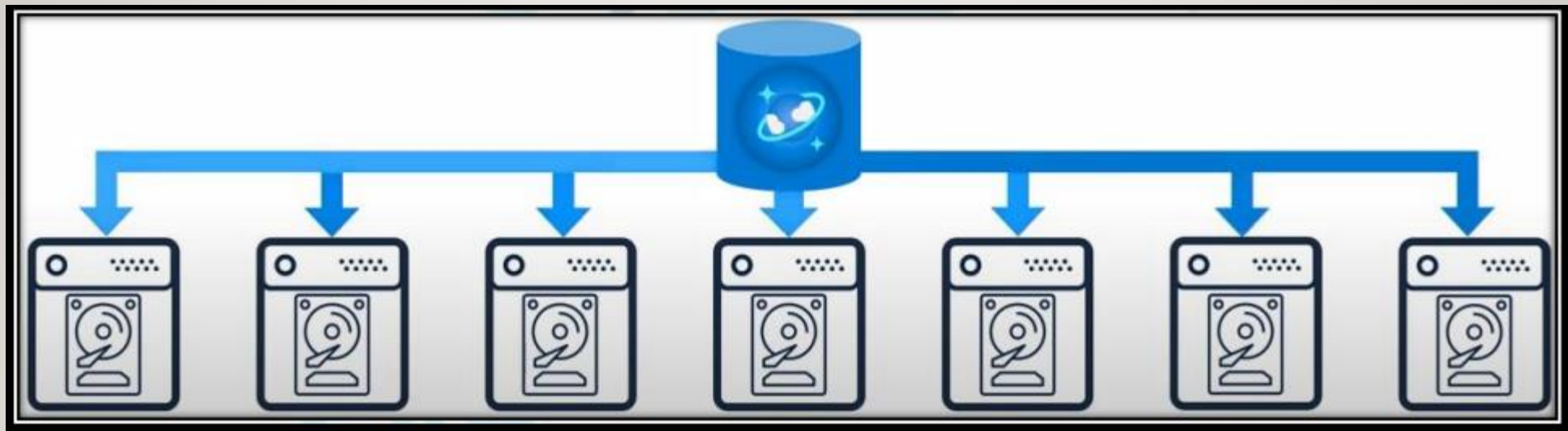
☒ Provision dedicated throughput for this container ⓘ

* Throughput (400 - 100,000 RU/s) ⓘ

400

Estimated spend (USD): **\$0.58 hourly / \$13.82 daily** (8 regions, 400RU/s, \$0.00016/RU)

SKALOWANIE POZIOME

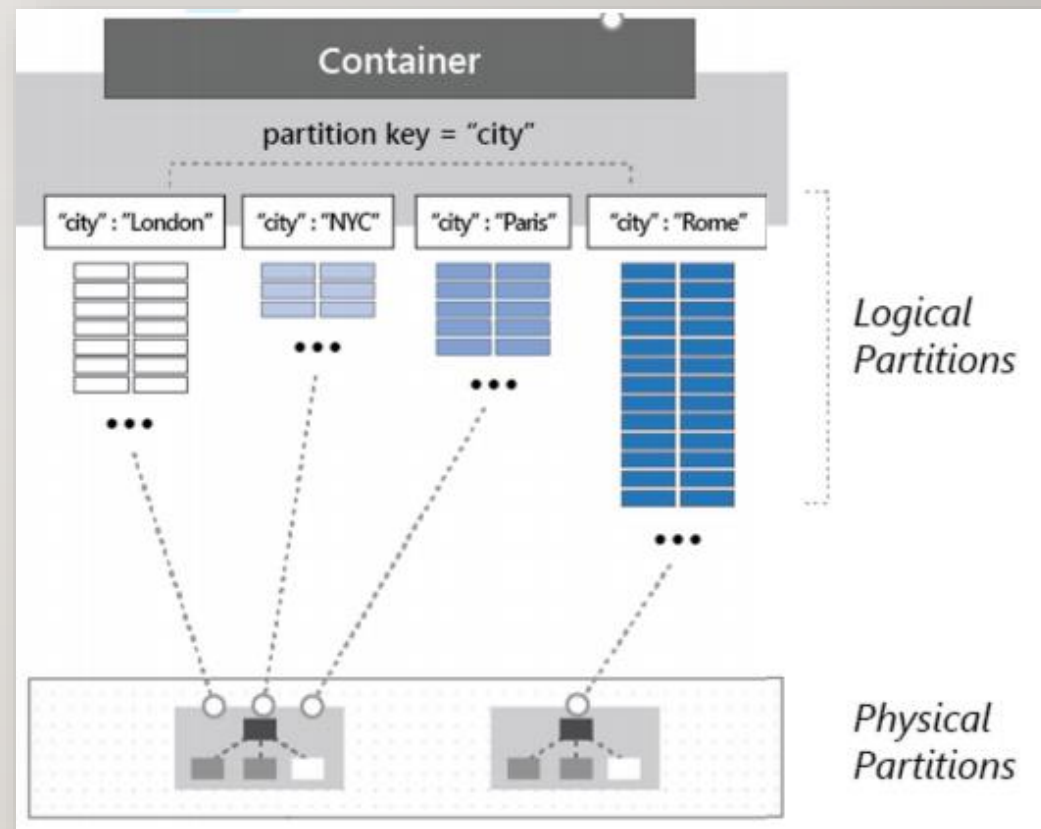


UNLIMITED STORAGE

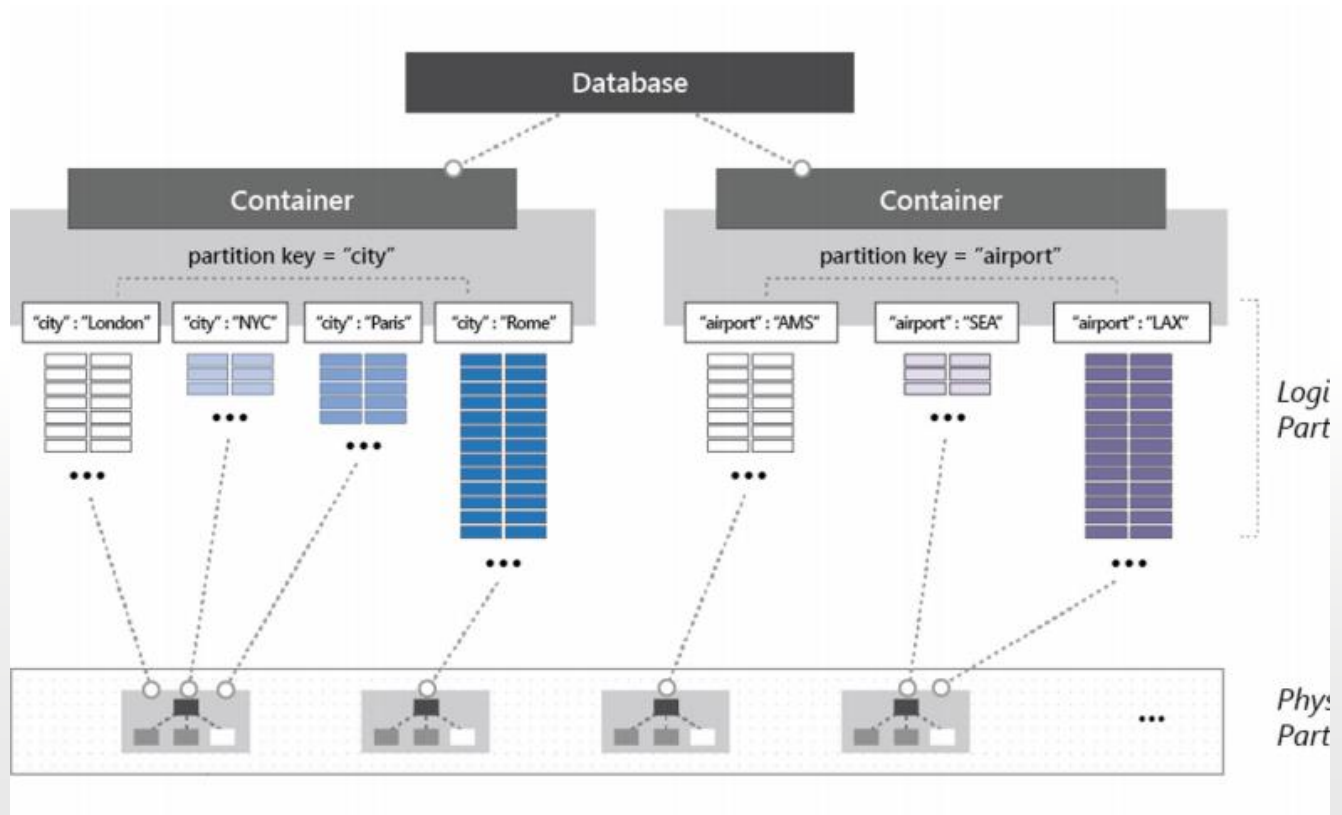
UNLIMITED
THROUGHPUT

PATRYCJONOWANIE

- Partycjonowanie: elementy w kontenerze są podzielone na odrębne podzbiory zwane partycjami logicznymi.
- Klucz partycji to wartość, według której Azure organizuje Twoje dane w logiczne podziały.
- Partycje logiczne tworzone są w oparciu o wartość klucza partycji, który jest skojarzony z każdym elementem w kontenerze
- Fizyczne partycje: Wewnętrznie, jedna lub więcej partycji logicznych jest mapowanych do jednej partycji fizycznej



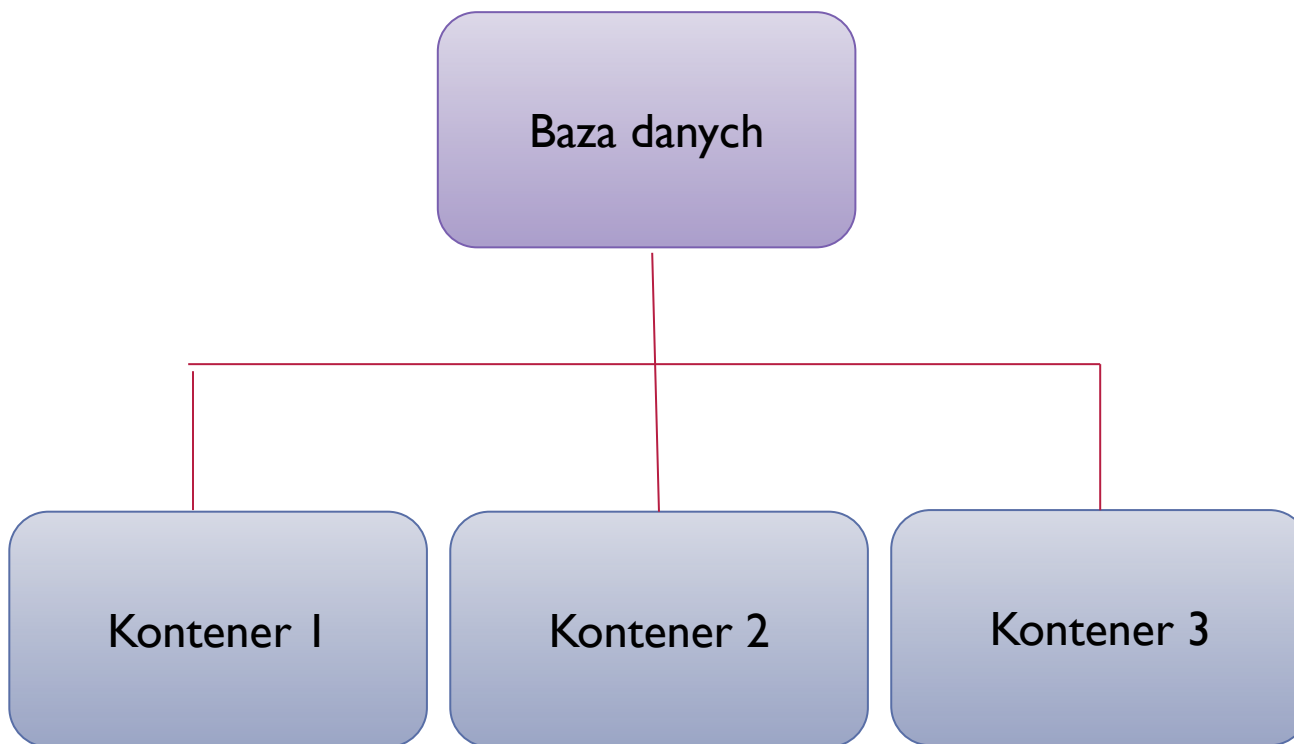
PARTYCJONOWANIE



PRZEPUSTOWOŚĆ DEDYKOWANA VS PRZEPUSTOWOŚĆ WSPÓŁDZIELONA

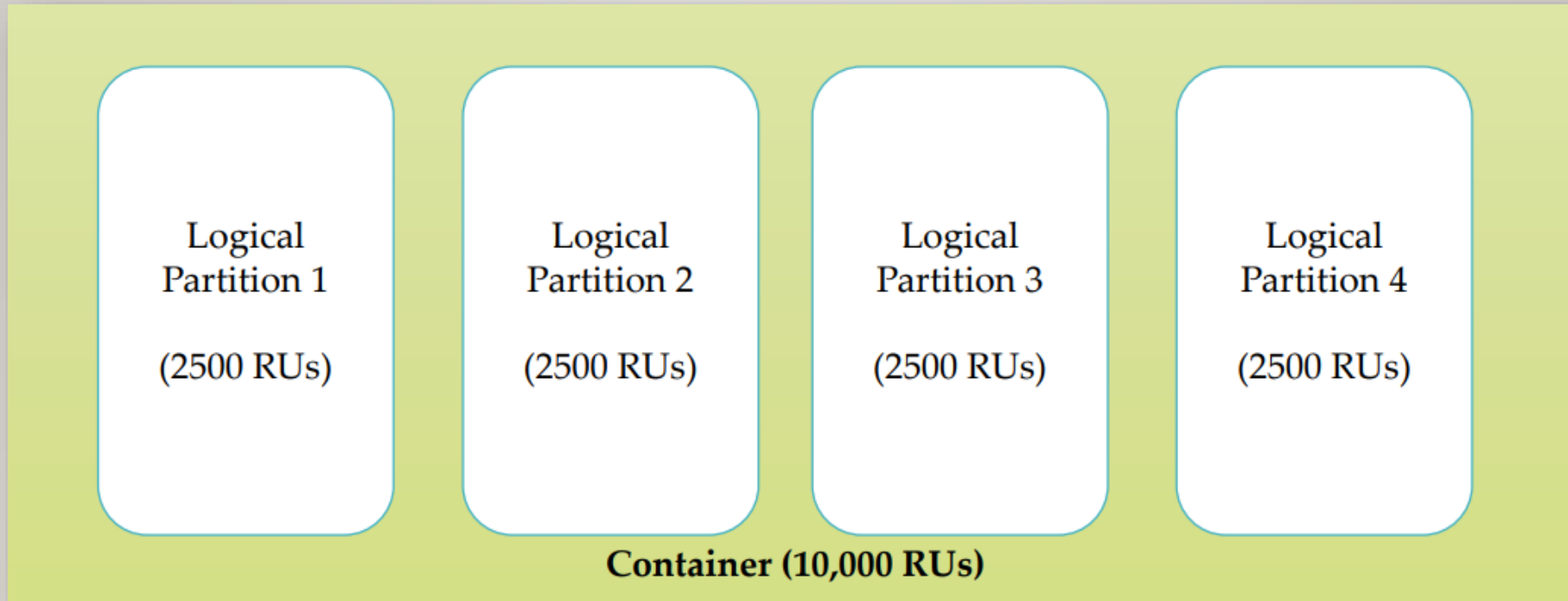
- Możemy ustawiać przepustowość na poziomie bazy danych lub na poziomie kontenera
- Współdzielona przepustowość – to przepustowość na poziomie bazy danych, która jest współdzielona przez wszystkie kontenery
- Przepustowość dedykowana – to przepustowość zdefiniowana dla kontenera, mamy wtedy pewność, że dany kontener zawsze będzie miał zarezerwowaną przepustowość

PRZEPUSTOWOŚĆ DEDYKOWANA | PRZEPUSTOWOŚĆ WSPÓŁDZIELONA



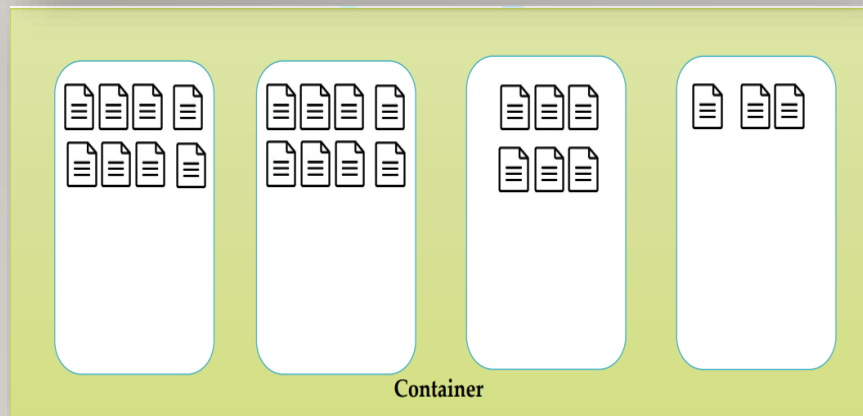
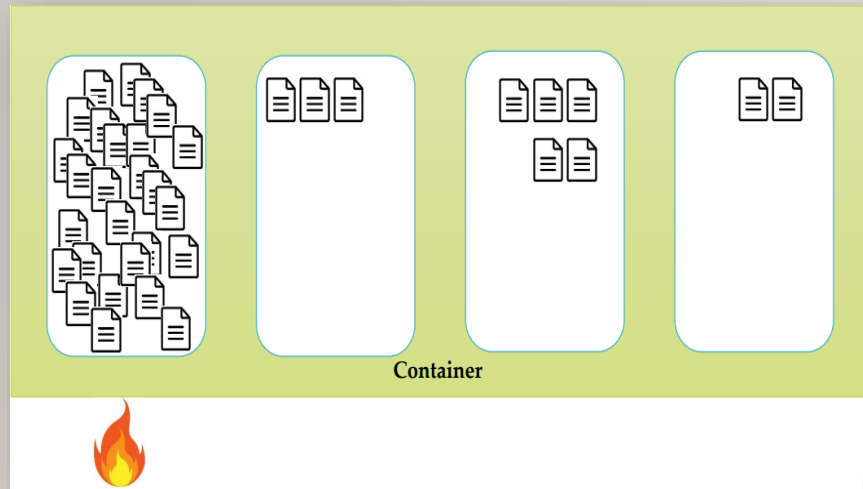
PRZEPUSTOWOŚĆ DEDYKOWANA VS PRZEPUSTOWOŚĆ WSPÓŁDZIELONA

- Możesz ustawić przepustowość na poziomie:
 - Bazy danych – współdzielona przepustowość
 - Kontenera – dedykowana przepustowość
 - Rekomendowane jest ustawianie przepustowości na poziomie kontenera
- Można ustawić alert dla przypadków osiągnięcia 80% przepustowości
- Requests are “throttled” (HTTP 429) – w przypadku przekroczenia przepustowości
- Przepustowość jest wybierana w momencie tworzenia kontenera / bazy danych



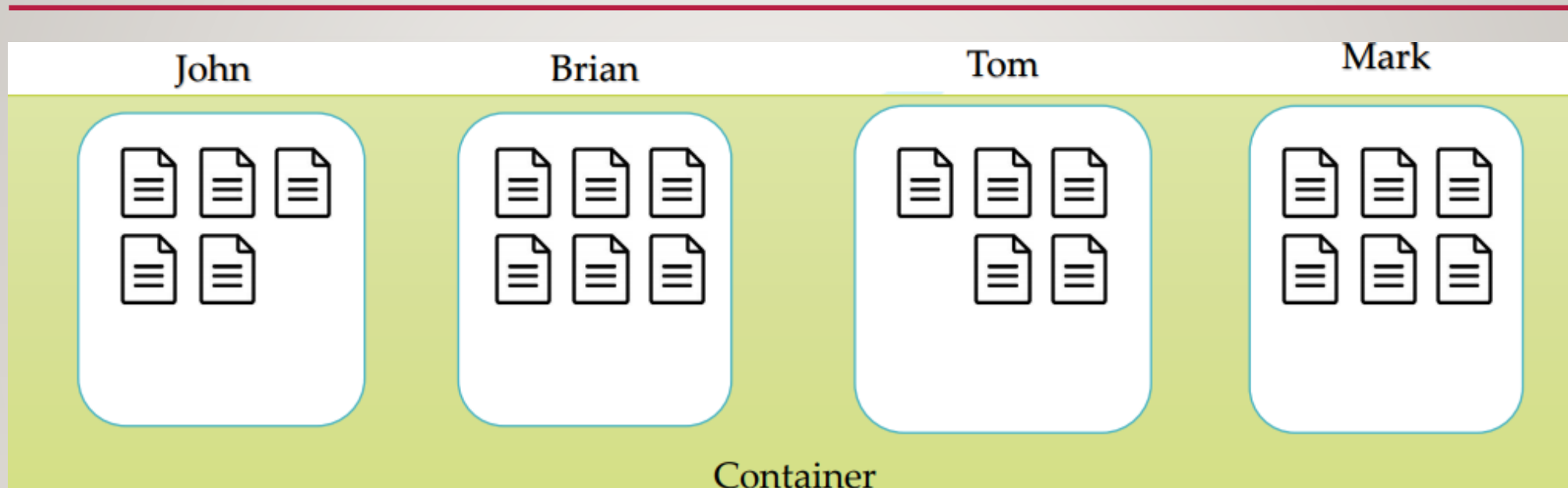
PODZIAŁ PRZEPUSTOWOŚCI NA PARTYCJE LOGICZNE

UNIKANIE GORĄCYCH PARTYCJI LOGICZNYCH



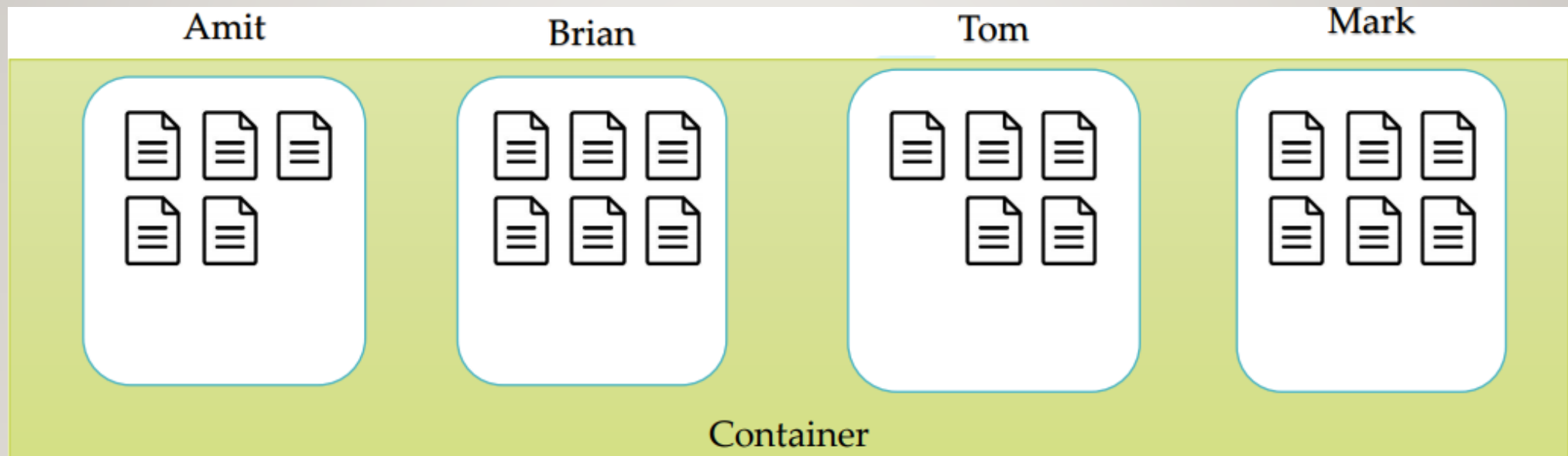
- Partycjonowanie wykonuje się w oparciu o Partition Key
- Przepustowość dla każdej partycji logicznej w danym kontenerze jest taka sama – przykład na poprzednim slajdzie
- Chcemy uniknąć gorących partycji – to znaczy, że zależy nam na tym, żeby dokumenty zostały równo podzielone wśród partycji logicznych oraz na tym, żeby podczas wykonywania zapytań zużycie partycji logicznych było jak najbardziej równe
 - Partition key – zły wybór – Current time
 - Partition key – dobry wybór – User ID, Product ID

ZAPYTANIE WEWNĄTRZ PARTYCJI



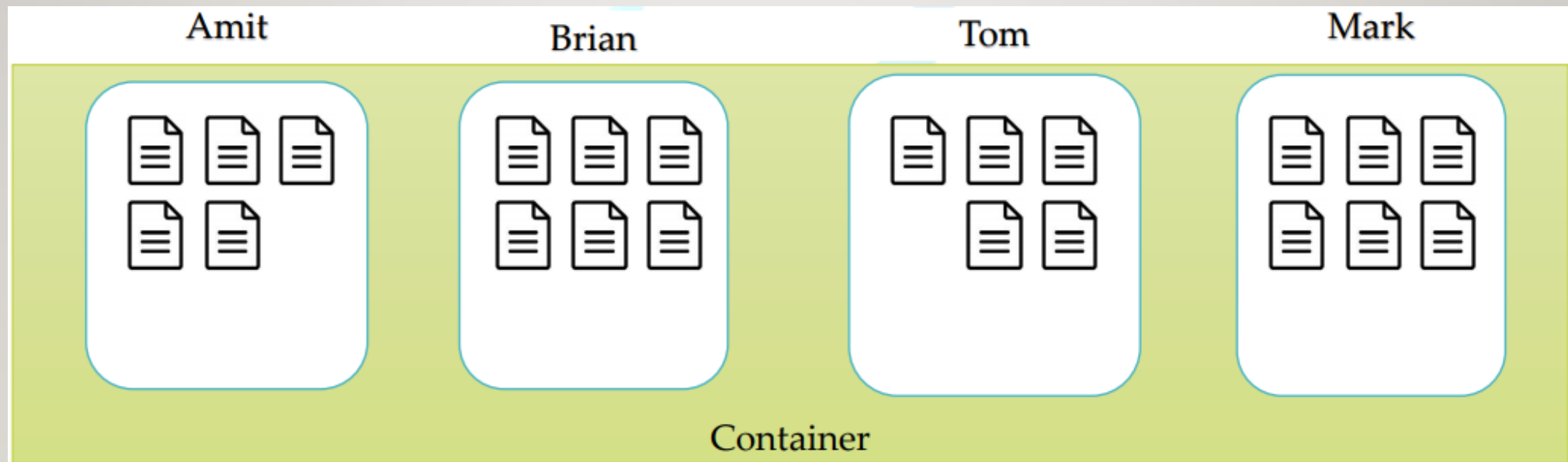
```
SELECT * FROM c WHERE c.username = 'Brian'
```

ZAPYTANIE OBEJMUJĄCE WIELE PARTYCJI



```
SELECT * FROM c WHERE c.favoritecolor= 'Blue'
```

KLUCZ ZŁOŻONY



Composite Key: CustomerName-mmddyyyy

KLUCZ PARTYCJI – NAJLEPSZE PRAKTYKI

- Równomierne rozmieszczenie pamięci
 - Klucz partycji, który nie powoduje powstawania "gorących punktów" w Twoich aplikacjach.
 - Wysoka kardynalność
 - Nie bój się wybrać klucza partycji, który ma dużą liczbę wartości np.: User Id & Product Id
- Równomierne dystrybuowanie żądań
 - RU są równomiernie rozdzielane pomiędzy wszystkie partycje.
 - Przegląd najpopularniejszych zapytań z **where** – tak aby zapytania były optymalne
- Rozważ ograniczenia
 - Maksymalna wielkość dokumentu - **2MB**
 - Maksymalna wielkość partycji logicznej – **20GB**

ZADANIE TEORETYCZNE

- Twoja organizacja planuje użyć Azure Cosmos DB do przechowywania danych telemetrycznych pojazdów generowanych z milionów pojazdów w każdej sekundzie. Jaki klucz partycji będzie optymalny do tego zadania?
- Model pojazdu?
- Numer VIN pojazdu? Potocznie numer nadwozia np.: ABCJ9EF6GHI23456

ODPOWIEDŹ

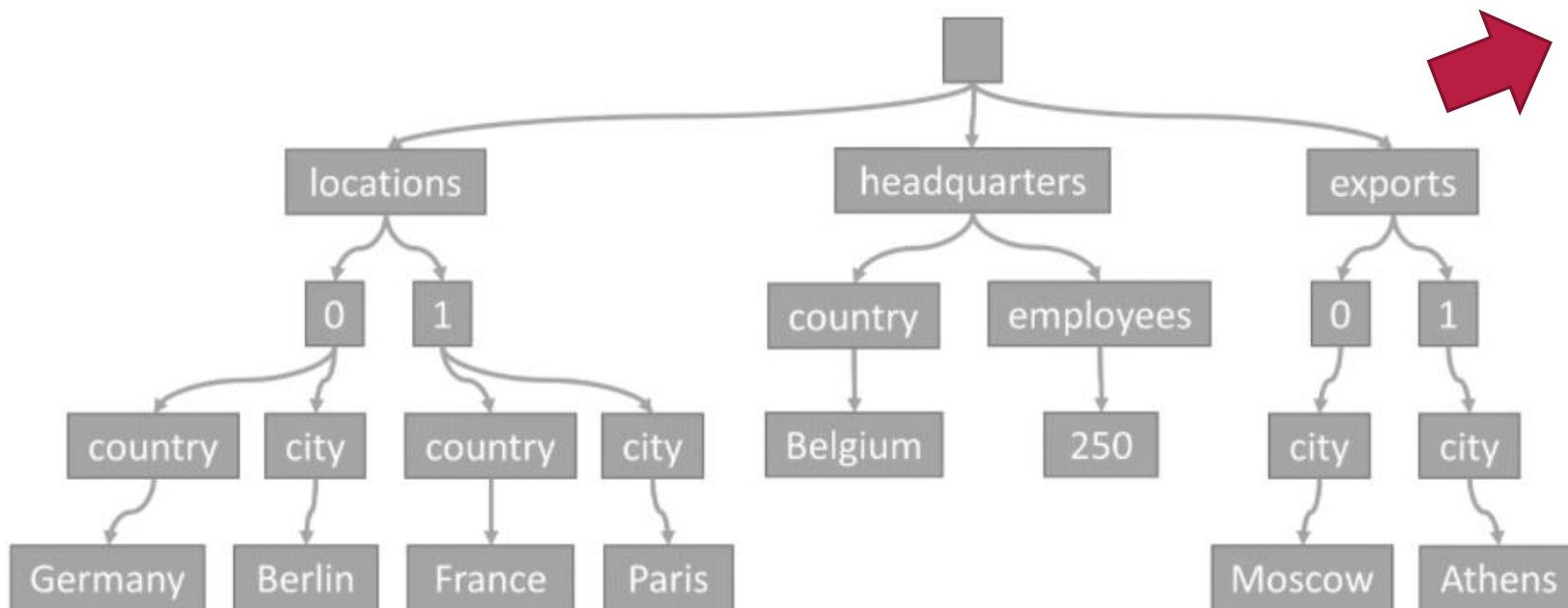
- **Model pojazdu: ŹLE:** Większość producentów samochodów ma tylko kilkadziesiąt modeli. Ta opcja jest potencjalnie najmniej szczegółowa, utworzy tworzy stałą liczbę partycji logicznych i może nie rozdzielać danych równomiernie (niektóre modele są popularniejsze) na wszystkie partycje fizyczne.
- **Numer VIN: DOBRZE:** Producenci samochodów dokonują transakcji w ciągu całego roku. Ta opcja pozwoli na bardziej zrównoważoną dystrybucję pamięci przez wartość klucza partycji.

AUTOMATYCZNE INDEKSOWANIE

- Indeksowanie wszystkich danych bez konieczności zarządzania indeksami
- Każda właściwość każdego rekordu automatycznie indeksowana
- Indeks aktualizuje się synchronicznie podczas tworzenia, aktualizowania lub usuwania elementów
- Nie jest to specyficzne dla SQL, ale dostępne dla wszystkich interfejsów API

JSON

```
{
  "locations": [
    { "country": "Germany", "city": "Berlin" },
    { "country": "France", "city": "Paris" }
  ],
  "headquarters": { "country": "Belgium", "employees": 250 },
  "exports": [
    { "city": "Moscow" },
    { "city": "Athens" }
  ]
}
```



/locations/0/country: "Germany"

/locations/0/city: "City"

/locations/1/country: "France"

/locations/1/city: "Paryż"

/centrala/kraj: "Przejmij"

/centrala/pracownicy: 250

/exports/0/city: "Zamów"

/exports/1/city: "Zamów"

*SELECT location FROM
location IN company.locations
WHERE location.country =
'France'*

TTL

Możesz ustawić czas
wygaśnięcia dla danych
Cosmos DB

Wartość Time to live jest
konfigurowana w
sekundach.

System będzie
automatycznie usuwał
wygasłe elementy na
podstawie wartości TTL

Zużycie tylko pozostałych
RU

Opóźnienie usuwania
danych w przypadku braku
RU

Mimo, że usuwanie danych
jest opóźnione, dane nie są
nie są zwracane przez
żadne zapytania (przez
dowolne API) po upływie
TTL.

KORZYŚCI Z GLOBALNEJ DYSTRYBUCJI

Wydajność:

- Zapewnia wysoką dostępność w wielu regionach
- We wszystkich regionach, przybliża dane do konsumenta.
- Jeżeli zależy nam na wydajności powinniśmy odpowiednio replikować dane – do regionów, które są parami itd. <https://build5nines.com/azure-region-pairs-explained/>

Ciągłość działania

- Przydatne w przypadku poważnej awarii lub klęski żywiołowej.

Funkcja multi master (multi region feature)

Mniejszy czas oczekiwania pozwala włączyć zapisy w wielu regionach w Azure Cosmos DB.
(wybierz resource -> Replicate data globalny).

Click on a location to add or remove regions from your Azure Cosmos DB account.

* Each region is billable based on the throughput and storage for the account. [Learn more](#)



Configure regions

Multi-region writes ⓘ

Disable

Enable

Configure the regions for reads, writes and availability zone (supported in selected regions and can only be configured when a new region is added). [+ Add region](#)

Write Region

Availability Zone

West US

Read Regions

Availability Zone

Action

East US

☐

Japan East

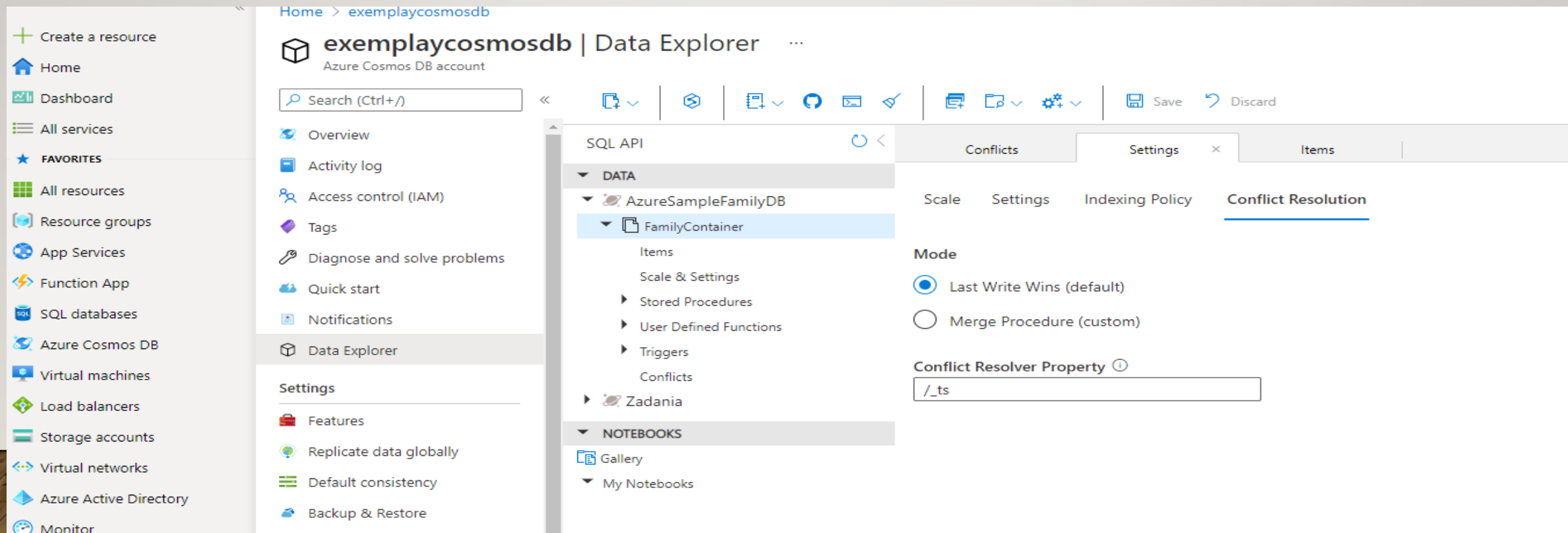
☐

West Europe

☐

Typy konfliktów i zasady rozwiązywania w przypadku korzystania z wielu regionów zapisu

- **Konflikty wstawiania:** te konflikty mogą wystąpić, gdy aplikacja jednocześnie wstawia dwa lub więcej elementów z tym samym unikatowym indeksem w co najmniej dwóch regionach. Na przykład ten konflikt może wystąpić z właściwością ID.
- **Zastąp konflikty:** te konflikty mogą wystąpić, gdy aplikacja aktualizuje ten sam element jednocześnie w co najmniej dwóch regionach.
- **Konflikty usuwania:** te konflikty mogą wystąpić, gdy aplikacja jednocześnie usuwa element w jednym regionie i aktualizuje go w innym regionie.



Failover (przełączenie awaryjne)

Search (Ctrl+/)

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Quick start

Notifications

Data Explorer

Settings

Features

Replicate data globally

Default consistency

Backup & Restore

Firewall and virtual networks

Private Endpoint Connections

CORS


Dedicated Gateway

Save Discard Manual Failover Automatic Failover

When you add a region to your account, you will be billed for the additional RU/s and storage copied to the region. To keep your account completely free, stay in 1 region with 400 RU/s and 5GB of storage. Click here to learn more. →

Click on a location to add or remove regions from your Azure Cosmos DB account.

* Each region is billable based on the throughput and storage for the account. [Learn more](#)



Configure regions

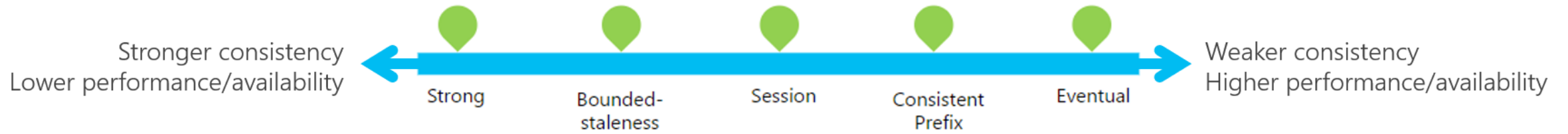
Multi-region writes ⓘ

Disable Enable

Configure the regions for reads, writes and availability zone (supported in selected regions and can only be configured when a new region is added). [+ Add region](#)

Regions	Reads Enabled	Writes Enabled	Availability Zone	Action
Germany West Central	✓	✓		
West US	✓	✓		
Japan West	✓	✓		
Korea South	✓	✓		

5 poziomów spójności



Silna spójność (Strong) - Zawiera gwarancje operacji atomowych, odczyty są gwarantowane do zwrócenia najnowszej zatwierdzonej wersji elementu.

Spójność ostateczna (Eventual) - W przypadku spójności ostatecznej nie ma gwarancji porządkowania dla operacji odczytu. W przypadku braku jakichkolwiek dalszych zapisów repliki staną się ostatecznie zbieżne. Spójność ostateczna to najslabsza forma spójności, ponieważ klient może odczytać wartości starsze niż te, które były wcześniej odczytywane.

Spójny prefiks (Consistent prefix) - Jeśli operacje zapisu zostały wykonane w podanej kolejności A, B, C, klient zobaczy |A|, |A, B| lub |A, B, C|, ale nigdy nie z kolejności permutacji, takich jak A, C lub B, A, C. Spójny prefiks zapewnia opóźnienia zapisu, dostępność i przepływność odczytu porównywalne do spójności ostatecznej, ale z gwarancją kolejności.

Spójność sesji - to najczęściej używany poziom spójności dla jednego regionu, a także aplikacji rozproszonych globalnie. Zapewnia ona opóźnienia zapisu, dostępność i przepływność odczytu porównywalne do spójności ostatecznej, ale również zapewnia gwarancje spójności w ramach sesji.

Powiązana nieaktualność - pobieramy dane w kolejności, ale z opóźnieniem do pewnego momentu. Ten moment wyznaczamy przez czas albo liczbę operacji.



Azure CLI

```
az cosmosdb create --name
                    --resource-group
                    [--capabilities]
                    [--default-consistency-level {BoundedStaleness, ConsistentPrefix, Eventual, Session, Strong}]
                    [--enable-automatic-failover {false, true}]
                    [--enable-multiple-write-locations {false, true}]
                    [--enable-virtual-network {false, true}]
                    [--ip-range-filter]
                    [--kind {GlobalDocumentDB, MongoDB, Parse}]
                    [--locations]
                    [--max-interval]
                    [--max-staleness-prefix]
                    [--subscription]
                    [--tags]
                    [--virtual-network-rules]
```

DZIĘKUJEMY ZA UWAGĘ

Patryk Cebrat

Dawid Szczerba

- Link do repozytorium zawierającym aplikacje w Javie z zadaniami – kilka query do przećwiczenia. Dodatkowo repozytorium zawiera prezentacje i opis demo, które było prezentowane na zajęciach – jak stworzyć kontener, jak dodać elementy do bazy danych, kilka przykładowych query, przykładowe dane do bazy danych - <https://github.com/patrykce/CosmosDBTutorial>