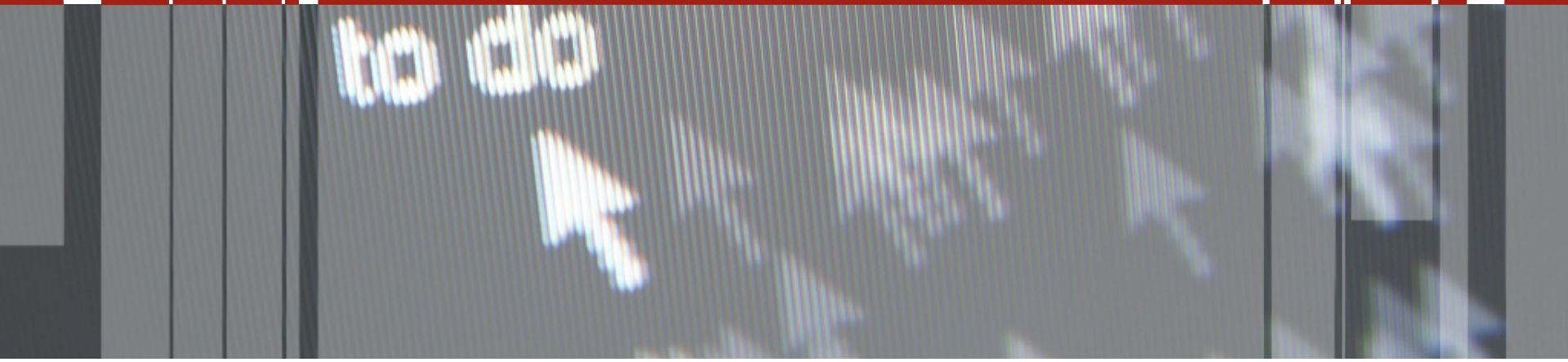


Pizza & Beer: OpenEdge Table Partitioning

Spotkania techniczne dla partnerów aplikacyjnych i klientów technologii Progress

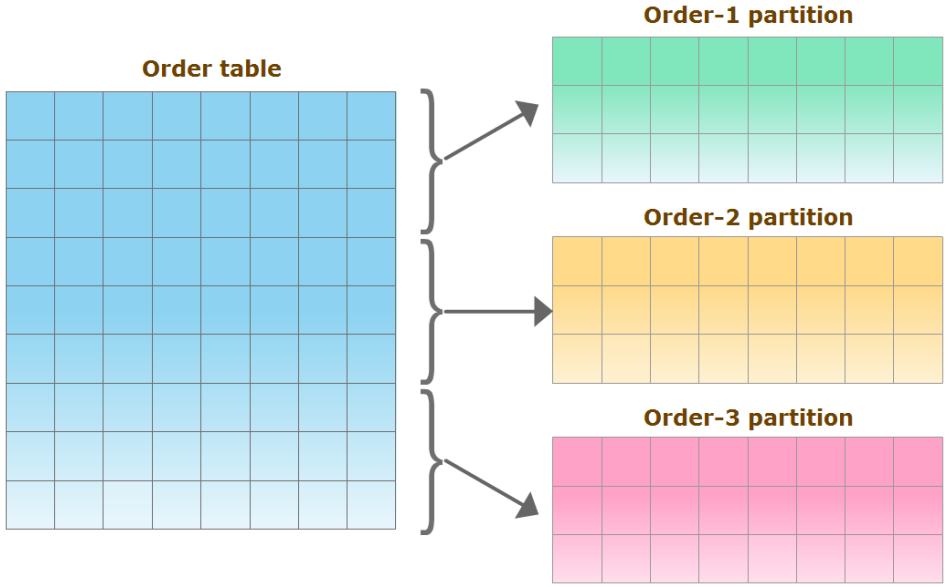
Piotr Tucholski

Agenda

- Co to jest Table Partitioning
- Terminologia
- Rodzaje partycjonowania tabel
- Implementacja TP
- Obsługa TP
- Testy wydajności

Partycjonowanie tabel (TP) - wprowadzenie

- Podział jedej tabeli na mniejsze części pod względem logicznym
- Te małe tabele są uporządkowane w oparciu o kluczowe pola oryginalnej tabeli
- Wszystkie dane są transparentne dla aplikacji
- Partycje mogą znajdować się w różnych obszarach
- Partycjonowane mogą być także indexy – tzw. Indexy lokalne
- Partycjonowanie Tabel nie może być zaimplementowane dla tabel Multi-Tenant
- Od OE 11.4



Kiedy stosować TP

Duża liczba jednoczesnych operacji CUD

Tabele zawierają dane historyczne, gdy trzeba okresowo archiwizować stare dane w miarę dodawania nowych danych.

Duże tabele, które muszą podlegać okresowym operacjom na rekordach i indexach

Tabele zawierają kolumny, wg których można logicznie pogrupować dane

Tabele zawierające dane z częstymi zapytaniami poprzez TABLE-SCAN a nie index

Tabele, które będą rosły do bardzo dużych rozmiarów

Zalety TP

Wysoka dostępność – gdy jedna partycja nie jest dostępna inne są dostępne

Łatwiejsza administracja – dump, load, odbudowa indexów szybsze dla partycji niż całej tabeli

Lepsza wydajność – powyższe operacje mogą być wykonywane równolegle dla różnych partycji

Restrykcje

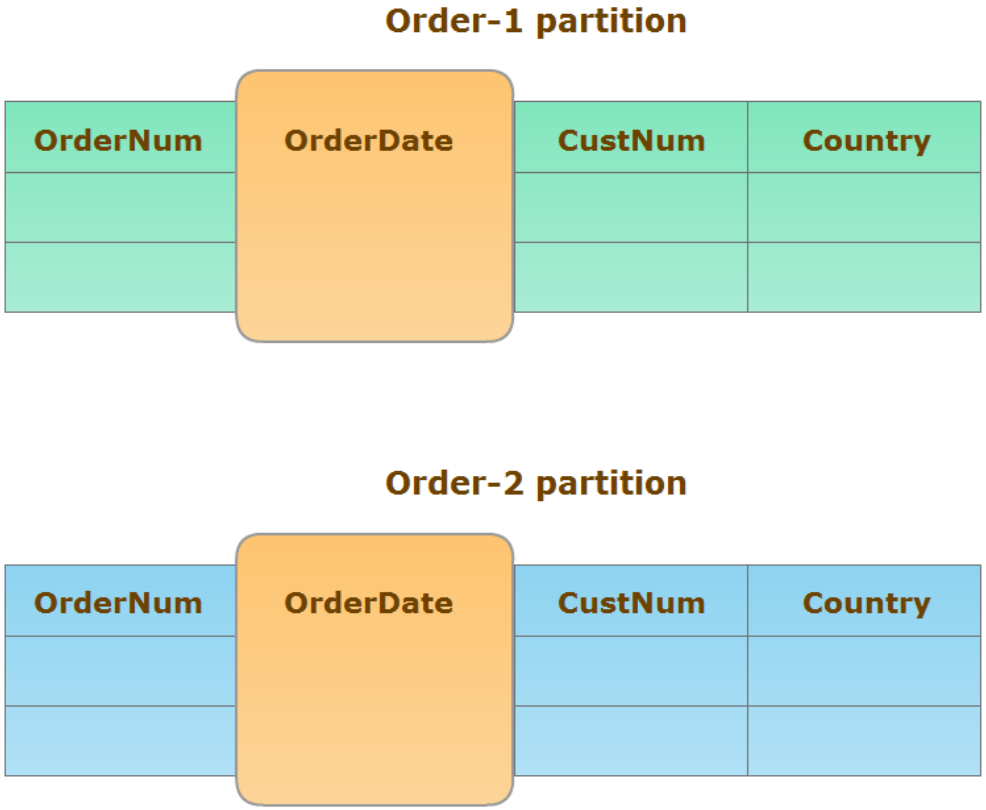
Partycje tabel dostępne TYLKO dla obiektów obszarów TYP II

Nie ma obsługi tabel tymczasowych

Nie ma obsługi dla tabel Multi-Tenant

Terminologia

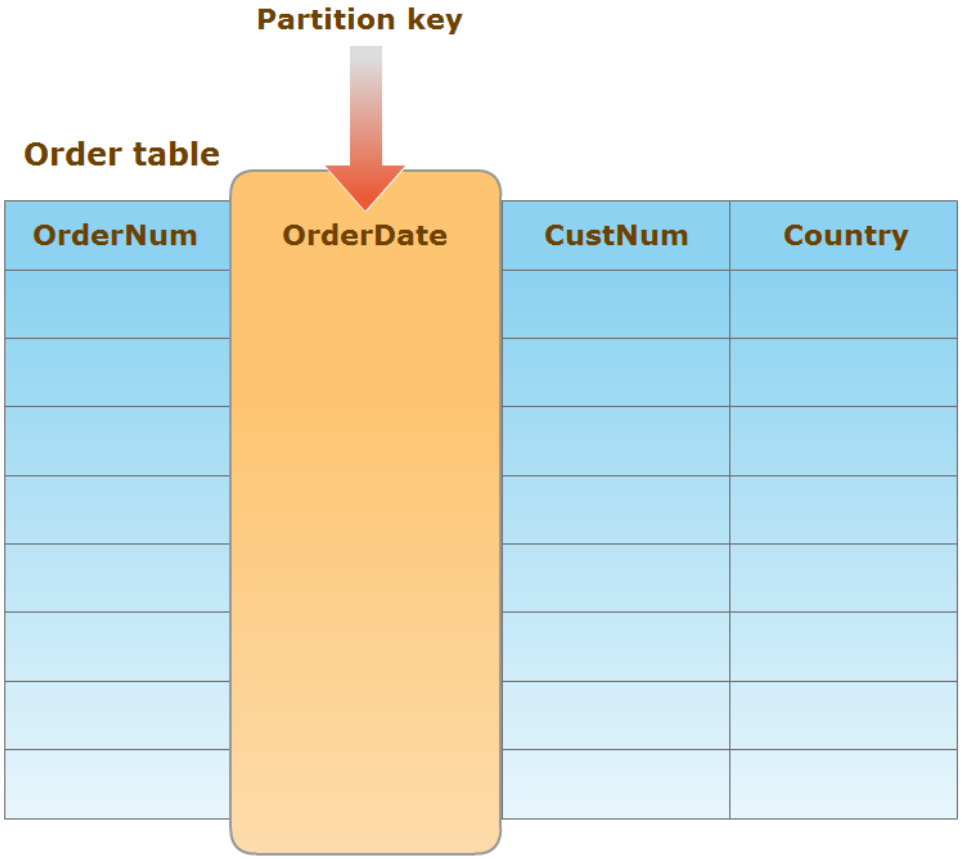
Partycjonowanie – proces który używa jednej kolumny do jednoznacznej identyfikacji każdej partycji. Podział może być według zakresu (zakresy wartości kolumn) lub według listy (lista odrębnych wartości kolumn).



Terminologia c.d.

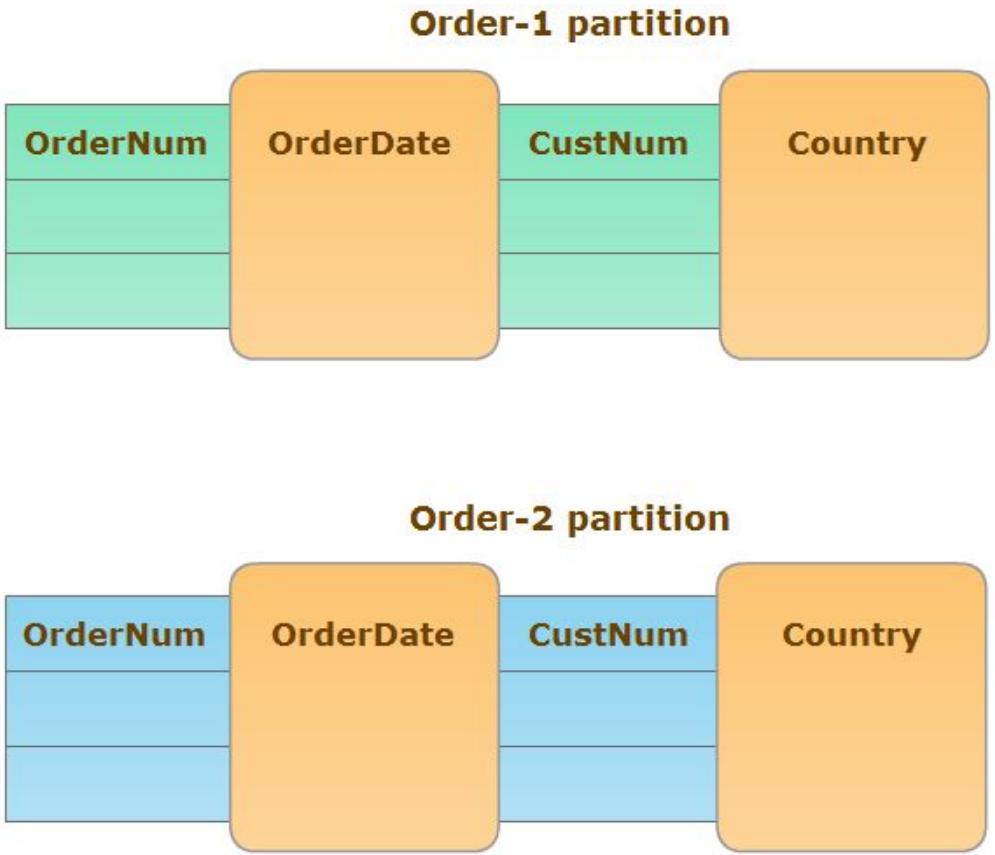
Klucz partycji: kolumna(y) jednoznacznie identyfikująca każdą partycję podzielonej na partycje tabeli.

- Kolumny partycji zawsze muszą być wiodącym komponentem indeksu. Tylko dane indeksowane (z wyjątkiem identyfikatorów RECID i ROWID) można podzielić na partycje
- Kolumny partycji muszą mieć znane wartości. Partycjonowanie tabel OpenEdge nie obsługuje wartości UNKNOWN "?" w kolumnach partycji.



Terminologia c.d.

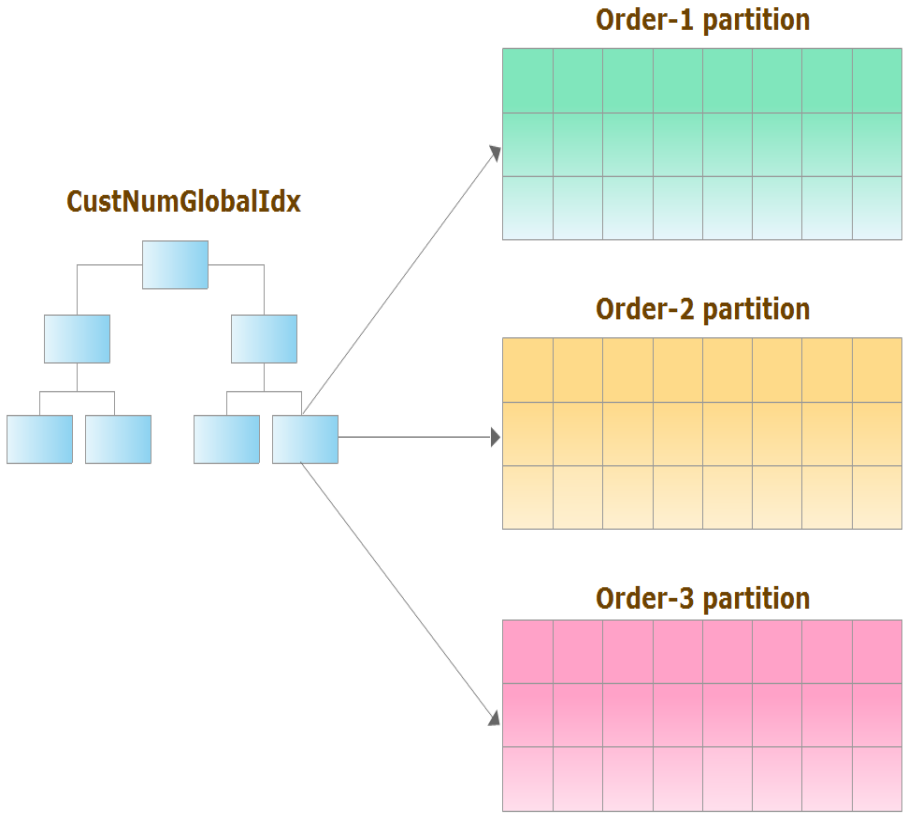
Subpartycje - użycie dwóch lub więcej kolumn do jednoznacznej identyfikacji każdej partycji. Można użyć do 15 kolumn.



Terminologia c.d.

Indeks globalny - indeks zawierający pozycje indeksu dla wszystkich rekordów we wszystkich partycjach tabeli. Indeksy globalne są takie same jak indeksy w tabelach niepartycjonowanych.

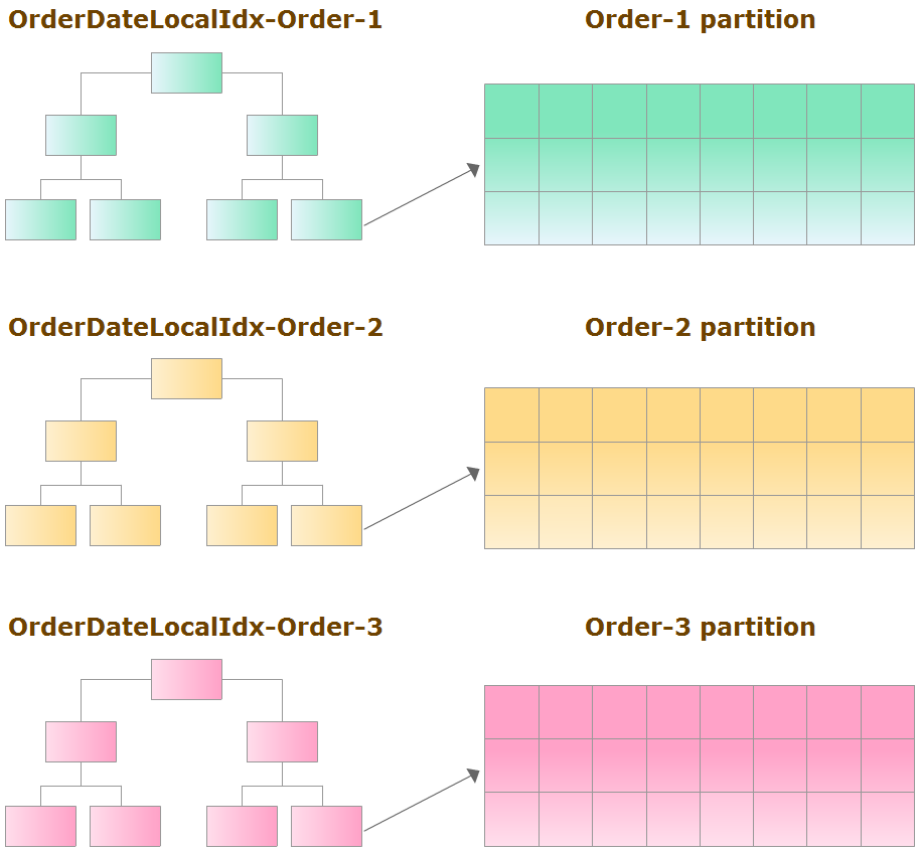
- Stosowany, aby wymusić unikalność w obrębie pwszystkich artycji lub posortować dane na podstawie kolumn niepartycjonowanych.
- Indeks globalny nie musi używać istniejącego klucza partycji. Może być oparty na innych danych zawartych w tabeli. Nie musi być unikalny.



Terminologia c.d.

Indeks lokalny - indeks oparty na tym samym kluczu partycji co tabela, ale zawarty w określonej partycji i stosowany tylko do rekordów tej partycji.

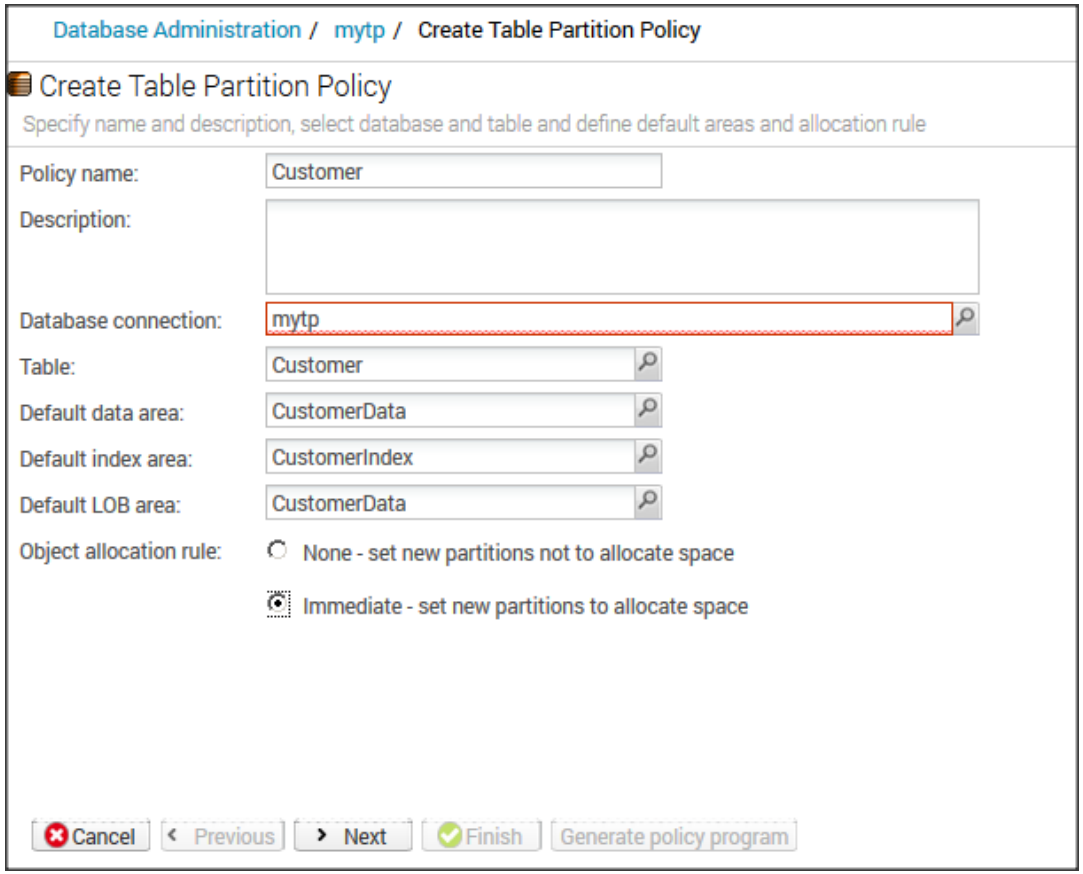
- Indeks lokalny może być również oparty na innych kolumnach w tej samej tabeli. Klucz partycji musi być wiodącym komponentem tego indeksu.
- Kiedy zdefiniowany jest indeks lokalny dla dowolnej partycji tabeli, OpenEdge automatycznie tworzy indeks lokalny dla każdej partycji tabeli.



Terminologia c.d.

Zasada partycjonowania (Partition policy) - jest rekordem meta-schematu bazy danych, który określa sposób partycjonowania tabeli.

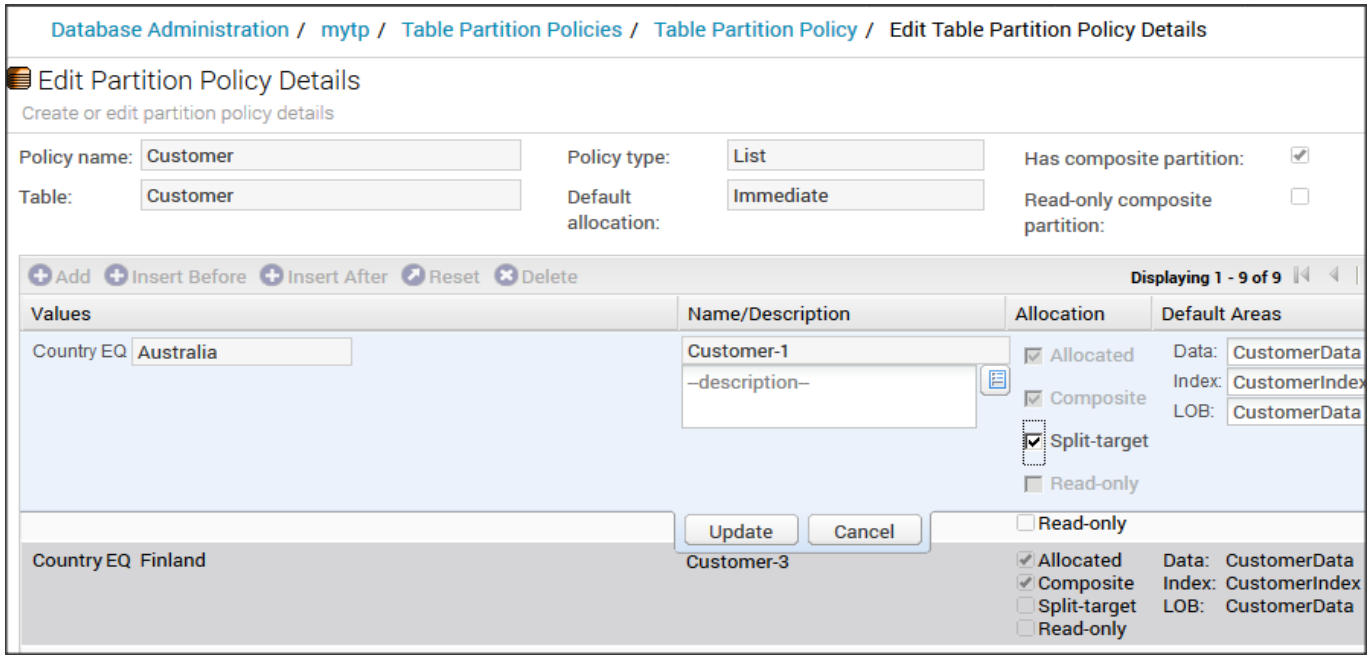
- Zawiera informacje takie jak nazwa zasady, tabela, obszary przechowywania danych, indeksy i pola LOB, reguły alokowania obiektów i typ partycji.
- Każda tabela podzielona na partycje musi mieć zdefiniowaną zasadę partycjonowania.



Terminologia c.d.

Szczegóły zasady partycjonowania (Partition policy detail) - rekord meta-schematu bazy danych, który definiuje wartości dla kolumn zdefiniowanych w strategii dotyczącej partycji. Zawiera informacje takie jak nazwa, określona wartość kolumny lub zakres wartości kolumn oraz obszary przechowywania obiektów (dane, indeks i LOB).

- **Każdy rekord szczegółów zasady partycji definiuje jedną partycję w tabeli.**
- **Można mieć maksymalnie 32 765 partycji na tabelę.**

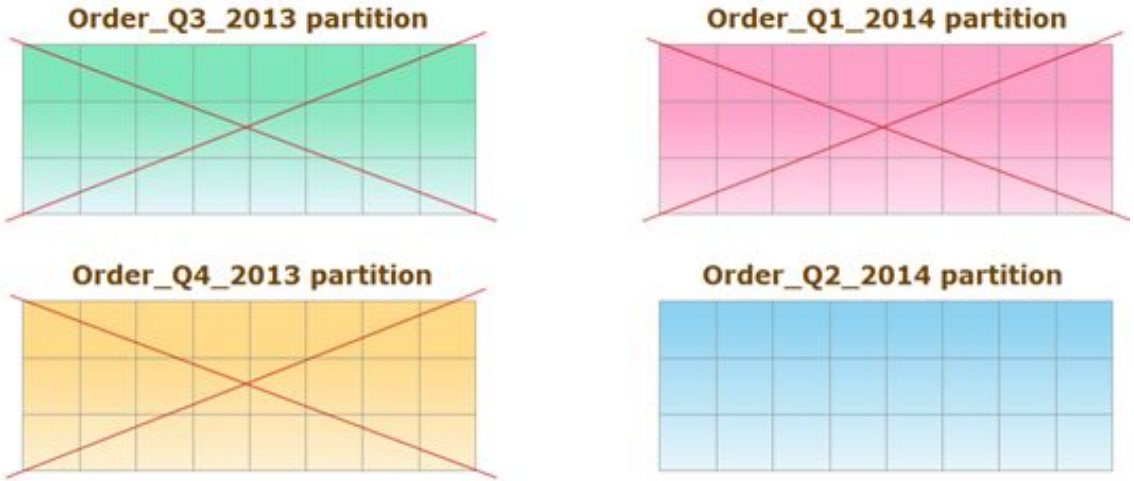


Terminologia c.d.

„Okrajanie" partycji (pruning) - proces runtime, w którym OpenEdge RDBMS wykonując instrukcje CRUD, analizuje tylko te partycje, które zawierają odpowiednie dane - w efekcie niepotrzebne partycje nie są brane pod uwagę.

- **OpenEdge RDBMS stosuje pruning zawsze gdy klauzula WHERE filtruje dane. „Okrajanie" partycji wpływa na poprawę wydajności.**
- **Zamiast skanować całą tabelę klauzula WHERE wymusza wyszukiwanie tylko w mniejszej partycji.**

```
FOR EACH Order WHERE  orderdate >= 04/01/2014 AND orderdate <= 6/30/2014:
        DISPLAY ordernum custnum orderdate.
```



Typy partycji

Mamy do wyboru dwa rodzaje partycji

- Lista - tabela oparta na liście odrębnych wartości w jednej kolumnie. Kolumna musi być typu: character, date, datetime, datetime-tz, decimal, integer, int64 lub logical.
- Zakres - tabela oparta na zakresach wartości w jednej kolumnie. Kolumna musi być typu: character, date, datetime, datetime-tz, decimal, integer, int64 lub logical. Zakres jest najczęstszym typem partycjonowania tabel.
- Podczas korzystania z partycji Zakresu należy ustawić tylko górną granicę każdej partycji. OpenEdge RDBMS automatycznie tworzy dla tabeli zestaw nie pokrywających się partycji.

Kryteria wyboru kolumn dla subpartycji

Mamy do wyboru dwa rodzaje partycji

- Kolumna partycji reprezentuje najlepszy sposób na podział danych w tabeli.
- Kolumna partycji jest często używana jako kryterium filtru w większości zapytań uruchamianych względem tabeli – wykorzystanie partition pruning.
- Kolumna partycji zawiera wartości, które są względnie statyczne w czasie. Zmniejsza to potrzebę przenoszenia rekordów do nowej partycji po zmianie wartości kolumn – co jest kosztowną operacją pod względem wydajności.

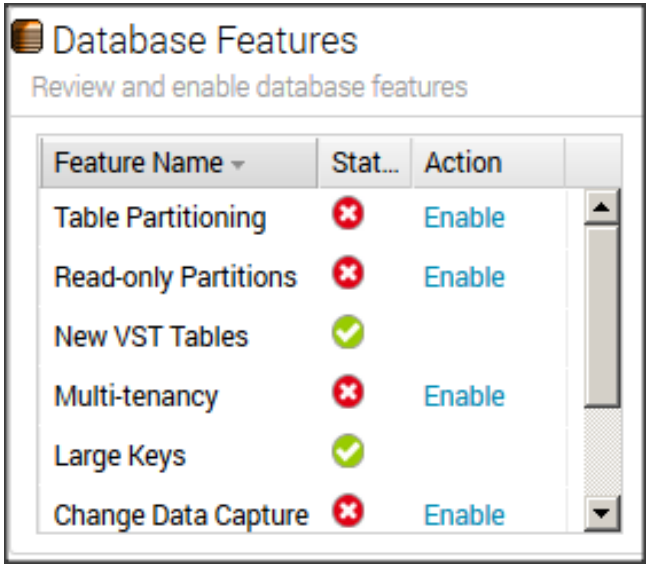
Przykłady zastosowań typów partycji

Mamy do wyboru dwa rodzaje partycji

- Partycjonowanie List jest często stosowane do orientowania danych według regionu, stanu, prowincji, przedstawiciela handlowego lub podobnych grup danych w tabeli.
- Partycjonowanie Zakres jest często stosowane do orientowania danych według okresu czasu (np. miesiące, kwartały, lata) lub zakresu wartości danych.

Implementacja partycjonowania

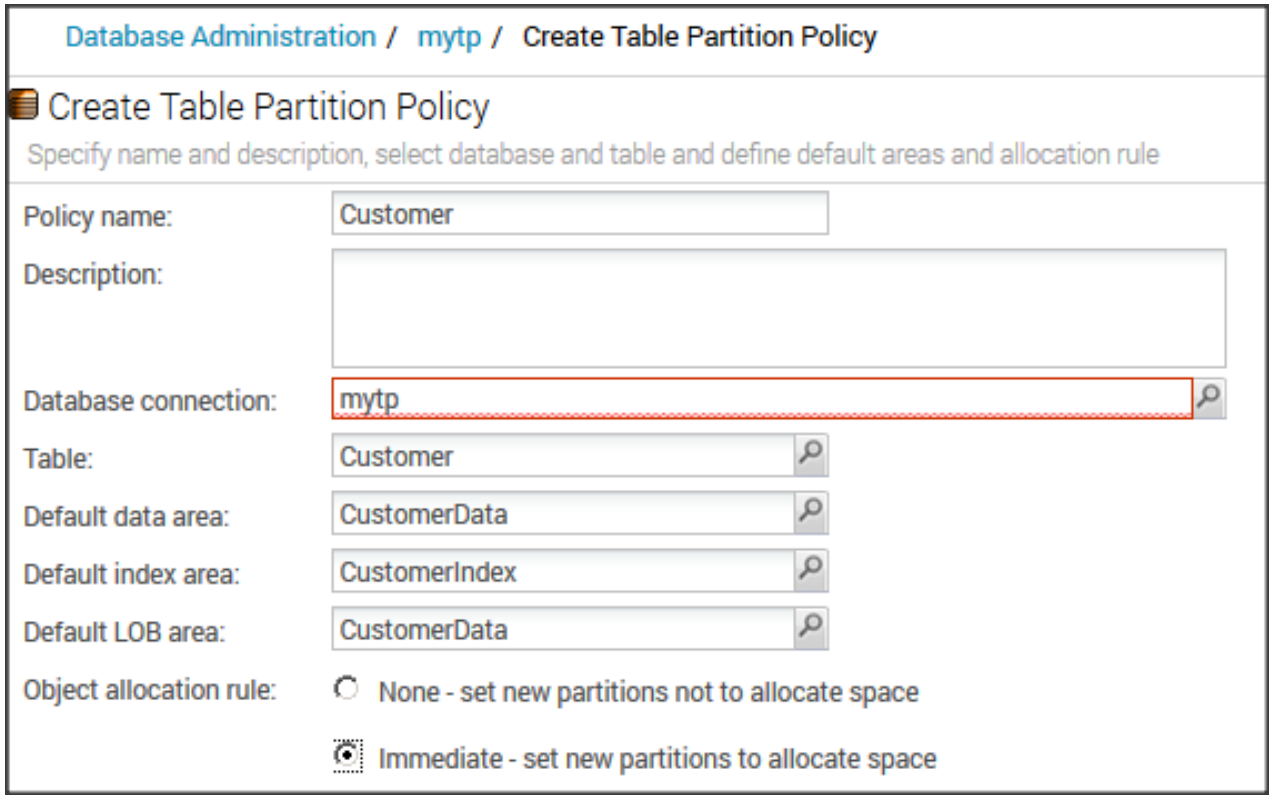
- Tabele i indeksy muszą znajdować się w obszarze typu II.
- W razie potrzeby można przenieść obiekty poleceniem **proutil tablemove** i **indexmove** z obszaru typu I do obszaru typu II.
- Można przerzucić dane dump i load.
- Należy włączyć funkcję partycjonowania tabel w bazie w OpenEdge Mangement lub: **proutil <dbname> - C enabletablepartitioning**
- Obie te operacje można wykonać na bazie online.



Implementacja partycjonowania c.d.

Definiowanie zasad partycjonowania

- Data Dictionary / Data Administration
- SQL
- OpenEdge Management



Obsługa partycji

- Indexbuild online
- Dodawanie partycji
- Podział i zmiana nazwy partycji
- Scalanie partycji
- Tworzenie partycji tylko do odczytu
- Przenoszenie partycji
- Zrzut partycji (dump)
- Przycinanie partycje (truncate)
- Usuwanie partycji

Obsługa partycji c.d.

proutil *db-name* -C partitionmanage split table *table-name*
{ partition *table-partition-name* | composite initial }
[useindex *index-name*] [recs *numrecs*]

proutil db-name -C partitionmanage merge table table-name
partition table-partition-name partition table-partition-name
[partition table-partition-name] [useindex index-name]
[recs numrecs]

proutil db-name -C partitionmanage truncate table table-name
{ partition table-partition-name | composite initial }
[recs numrecs] [deallocate]

proutil db-name -C partitionmanage view
[table table-name [partition table-partition-name | composite initial]]
{ list | state | status }

Partycjonowanie - wzrost i spadek wydajności

Wzrost wydajności obserwuje się najczęściej w przypadku operacji wstawiania, aktualizacji i usuwania danych, gdy umiarkowana liczba równoczesnych użytkowników uzyskuje dostęp do tabeli (tabel).

Spadek wydajności obserwuje się gdy dane przesuwają się z partycji do partycji ze względu na zmiany wartości klucza głównego.

Test wydajności

- Test wydajności został przeprowadzony przez Richa Banville'a i Dapeng Wu.
- Przebieg został zaprojektowany w celu zbadania wpływu różnych strategii partycjonowania na system baz danych.
- Test został skonfigurowany do mierzenia wydajności dla operacji READ, WRITE i DELETE w tabeli "Order", a wyniki porównane gdy partycjonowanie tabel było włączone i wyłączone.
- Zostały użyte dwie strategie partycjonowania gdy tabela jest podzielona na partycje:
  1. Partycja zakresu z kolumną "order-date" jako klucz partycjonowania
  2. Subpartycja wg kolumn "region" i "order-date" jako klucze partycjonowania.
- Dla każdej strategii partycjonowania dla porówania był wykonany test oparty na tabeli bez partycjonowania.
- Zmiany wydajności są obliczane wg różnic między konfiguracjami partycjonowanymi i niepodzielonymi na partycje.
- Uwzględniono różne liczby równoczesnych użytkowników, pokazując zmiany wydajności wraz ze wzrostem obciążenia.

Test wydajności - konfiguracja

Obszary Typ II

- Dane i indexy odseparowane
- Blok 8 KB z klastrem o wielkości 512 (dane) i 64 (indeksy)
- Wszystkie partycje o oddzielnych obszarach

Dane

- Średnia wielkość rekordu 257, RPB (32)
- 50 000 rekordów do 10 000 000 na przebieg (w zależności od #użytkowników)
- 3 indeksy globalne i 2 indeksy lokalne

BI

- Blok 8 KB z klastrem o wielkości 128 MB

Test wydajności – konfiguracja c.d.

Parametry serwera bazy

- Buffer pool: -B 50000 -lruskips 250
- Lock table: -L 100000 -lkwtmo 3600
- Transaction: -TXERetryLimit 1000
- BI: -bibufs 4000 -bwdelay 20
- Latching: -spin 50000 -napmax 10
- Page writers: 1 BIW 3 APWs

Maszyna do testów

- 16 sparcv9, częstotliwość procesora 3600 Mhz
- Rozmiar pamięci: 32768 MB

Test wydajności – metodologia

- Skalowanie użytkowników: 1, 2, 5, 10, 25, 50, 100, 200
  - Unikanie konfliktów po stronie aplikacji
  - Monitorowanie konfliktów wewnętrznych zasobów
- Wykonane operacje: Create, Read, Delete
- Zmienny zakres transakcji: 10, 100, 500 rekordów na transakcję
- Zmienny schemat partycjonowania
  - Brak partycjonowania
  - Partycjonowanie Zakresu {order-date}
  - Subpartycjonowanie {region (9), order-date}
- Dbanalys wykonano przed i po każdym etapie
- Baza danych odtworzona z tym samym plikiem .st dla każdego przebiegu

Test wydajności – wyniki

- Zastosowano dwie strategie partycjonowania do mierzenia zmian wydajności między konfiguracją partycjonowaną i niepartycjonowaną.
- Pierwszą strategią jest użycie subpartycjonowania wg kolumn „Region" i „Order-date", drugim jest użycie partycjonowana zakresu wg kolumny „Order-date".
- W obu strategiach operacje WRITE, READ i DELETE są wykonywane w tabeli "Order", a czas zakończenia każdej operacji służy do pomiaru wydajności tej operacji.
- Następnie, aby pokazać różnicę wydajności czasy są porównywane z wartościami z konfiguracji niepartycjonowanej.
- Aby pokazać wpływ wydajności na skalowalność systemu dla każdej konfiguracji testy są wykonywane dla różnej liczby użytkowników.

Test wydajności – wyniki c.d.

Test porównawczy między dwoma schematami partycjonowania dla 100 transakcji

- Wybór właściwego schematu partycjonowania może mieć znaczący wpływ na wydajność.
- W przykładzie testowym przy użyciu najpierw kolumny region, a następnie zakresu dat, wydajność znacząco się poprawiła (~ 138%), gdy 25 lub więcej użytkowników jednocześnie wprowadza zmiany w danych.
- Dla porównania, gdy zastosowano tylko zakres dat, nie zaobserwowano poprawy, a w niektórych przypadkach pogorszenie wydajności.

Test wydajności – wyniki c.d.

Konfiguracja: region i order-date

	1	2	5	10	25	50	100	200
Deletes (100)	3.52	4.76	-1.5	3.16	138.15	97.98	126.16	100.14
Writes(100)	-4.01	-0.58	-4.24	-3.49	104.65	125.14	117.55	134.72
Reads(100)	-2.63	3.24	-2.27	-1.3	-0.06	0.19	1.83	2.09

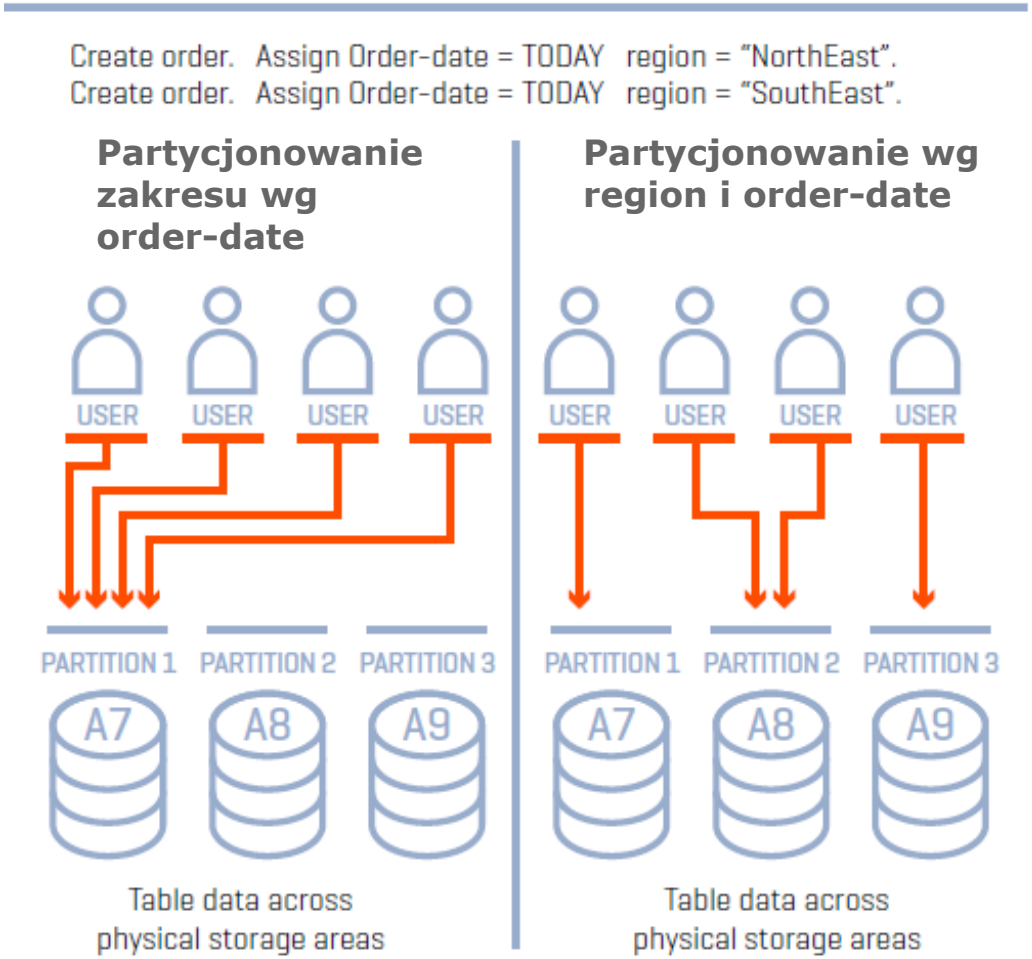
Test wydajności – wyniki c.d.

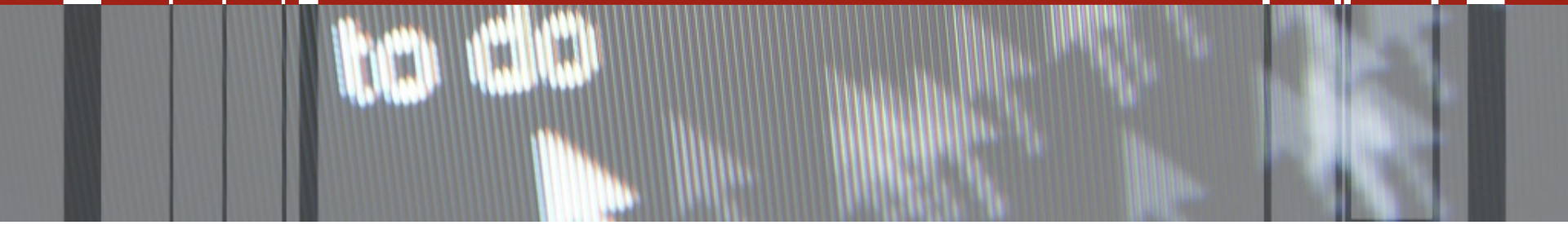
Konfiguracja: tylko order-date

	1.00	2.00	5.00	10.00	25.00	50.00	100.00	200.00
Deletes (100)	0.68	0.05	(1.25)	(1.94)	(3.07)	(6.86)	(4.56)	(7.28)
Writes(100)	(7.61)	(3.81)	0.06	(3.10)	1.68	2.83	(0.38)	(0.46)
Reads(100)	0.66	2.56	3.17	0.10	0.57	(0.63)	(0.20)	0.18

Test wydajności – wyniki c.d.

Gdzie leży przyczyna poprawy dla pierwszej konfiguracji?





Dziękuję za uwagę