POLITECHNIKA GDAŃSKA

Wydajność w Oracle

Optymalizacja struktury bazy i zapytań
Optymalizacja działania instancji

Wydajność

Obszary optymalizacji

Zmiana struktury bazy

Zmiana zapytań

Wskazówki

Partycje i klastry

Śledzenie wydajności

TK_PROF

AWR

ADDM

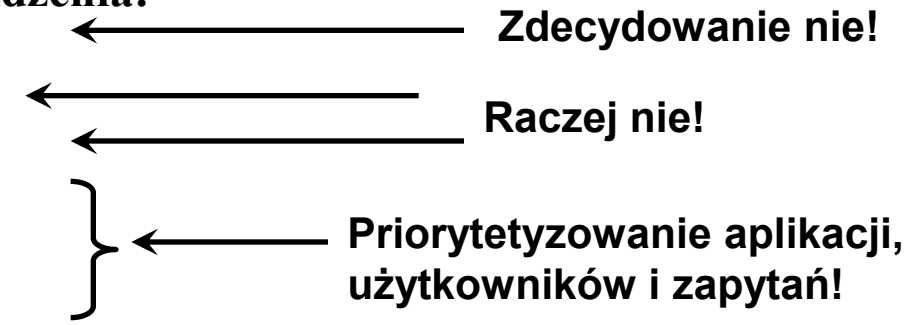
Strojenie instancji

Opracowanie: Agnieszka Landowska

POLITECHNIKA GDAŃSKA

Optymalizacja wydajności

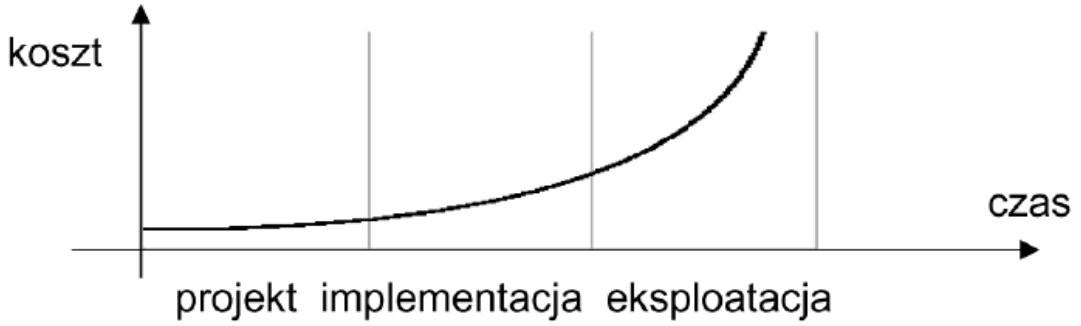
- **Cele optymalizacji**
  - Właściwe wykorzystanie zasobów (których jest zawsze za mało)
  - Zapobieganie awariom i okresowej niedostępności
  - Skalowalność
- **Poziomy optymalizacji**
  - Na poziomie zapytań i aplikacji
  - Na poziomie logicznego projektu bazy danych
  - Na poziomie dopasowania Oracle
    - struktury składowania
    - instancja (pamięć, procesy, zasoby)
  - Na poziomie dopasowania środowiska uruchamiania
    - Sprzęt (dyski, pamięć, procesor)
    - Oprogramowanie (system operacyjny, wersja Oracle)
    - Sieć
- **Optymalizacja z punktu widzenia:**
  - Pojedynczego zapytania
  - Pojedynczej aplikacji
  - Pojedynczego użytkownika
  - Wszystkich zapytań
  - Wszystkich aplikacji
  - Wszystkich użytkowników



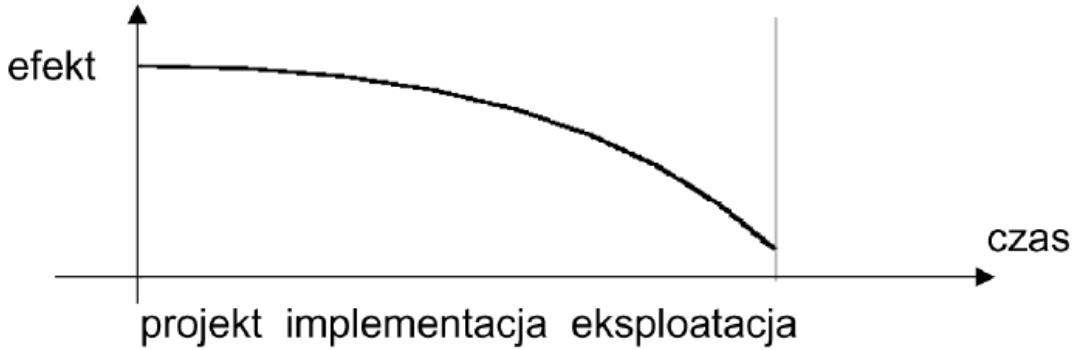
POLITECHNIKA GDAŃSKA

Kiedy stroić?

Koszt strojenia w trakcie budowy i eksploatacji systemu informatycznego



Efekt strojenia w trakcie budowy i eksploatacji systemu informatycznego



- **Eliminacja wąskich gardeł czy aktywne monitorowanie?**

Źródło rysunku: PLOUG, Jezierski, Monitorowanie Oracle …, 2003

POLITECHNIKA GDAŃSKA

Literatura

- **Dokumentacja Oracle**
  - **Oracle Database Performance Tuning Guide**
- **Antognini Christian „Troubleshooting Oracle Performance”**
  - **9i, 10g, 11g,**
  - **Skupia się na optymalizacji struktury bazy i zapytań**
- **Artykuły PLOUG**
  - **Karwowski, Metodyka strojenia baz danych Oracle 9i**
  - **Wojciechowski, Zakrzewicz, Kosztowy optymalizator zapytań**
  - **Figas, Enterprise Manager 9.2 jako narzędzie wspomagające proces strojenia i przewidywania problemów, 2003**
  - **Świerzy, Optymalizacja aplikacji OLAP pracujących w oparciu o SZBD Oracle9i**
  - **Karwowski, Kiedy tworzymy indeksy?, 2003**
  - **Kołodziej, Czego mądrzy ludzie z Oracle’m nie robią, 2003**

POLITECHNIKA GDAŃSKA

Przestrzenie optymalizacji

- Projekt logicznej bazy danych
  - Czasem trzeba zrezygnować z normalizacji
- Ocena projektu aplikacji
- Optymalizacja zapytań
  - Specjaliści twierdzą, że ok. 80% problemów z wydajnością to nieoptymalne zapytania
- Strojenie pamięci instancji (buforów SGA, obszarów PGA)
- Strojenie mechanizmów przechowywania danych (struktur składowania)
- Optymalizacja operacji odczytu/zapisu na dyskach fizycznych
- Eliminowanie dodatkowych zdarzeń
  - Blokady i przestoje
  - Wprowadzone przez użytkowników wyzwalacze, alerty, zadania (jobs)
- Zmiany w konfiguracji systemu operacyjnego
- Zmiany w konfiguracji ruchu sieciowego
- Zmiany sprzętowe

POLITECHNIKA GDAŃSKA

Narzędzia optymalizacji

- **Moduły Oracle**
  - **AWR – Automatic Workload Repository**
  - **ADDM – Automatic Database Diagnostic Monitor**
  - **TKPROF – TraceKernel Profiler – analiza logów**
- **Pakiety dodatkowe Oracle (często z poprzednich wersji) np.**
  - **DBMS_STATS – moduł gromadzący statystyki dla tabel**
- **Doradcy Oracle**
  - **Memory Advisors**
  - **SQL Tuning Advisors**
  - **SQL Access Advisors**
  - **Partitioning Advisor**
  - **SQL Performance Analyzer**
- **Alerty Oracle**
  - **Pochodzą z modułów lub/i doradców Oracle**
  - **Niektóre są generowane niezależnie od ustawień (domyślnie), inne trzeba włączyć**
- **Uniwersalne narzędzie optymalizacji – MYŚLENIE ☺**

POLITECHNIKA GDAŃSKA

Optymalizacja „akcyjna”

- „Zapytania do bazy nie wykonują się tak szybko, jak kiedyś”
- „Zapytanie X wykonuje się za długo”
- Zbadaj problem wydajnościowy
  - Opinia użytkownika zgłaszającego problem – co jest zbyt wolne (która funkcja/zapytanie)? kiedy i w jakich warunkach (powtarzalność)? jakiej poprawy oczekuje (cel optymalizacji)?
  - Zbierz opinie innych użytkowników
  - Zadaj pytanie: „co się zmieniło od poprzedniej wersji, kiedy działało to dobrze?”
  - Zadaj pytanie: „czy wykonywanie tego zapytania jest konieczne?”
  - Użyj automatycznych narzędzi (AWR, ADDM, Oracle Advisors, Top Sessions View, Top SQL View) – analizuj właściwy przedział czasowy!
    - Podobno ADDM znajduje 9 z 10 problemów wydajnościowych
- Szukaj przyczyny - kolejność???
  - Aplikacje i zapytania
  - Środowisko działania Oracle (dyski, pamięć, sieć, system operacyjny, oprogramowanie)
  - Instancja i serwer Oracle (struktury pamięci, struktury składowania, parametry pracy serwera)
- Zaplanuj i wprowadź zmiany
  - Jedna na raz!
- Sprawdź wprowadzone zmiany i stabilność instancji oraz korzystających z niej aplikacji
  - Często najpierw na serwerze testowym, a potem na produkcyjnym
  - Na serwerze produkcyjnym działanie optymalizacji trzeba sprawdzić ponownie

POLITECHNIKA GDAŃSKA

Optymalizacja proaktywna – sugerowana kolejność

- **Analiza obciążenia instancji w czasie**
  - **Redukcja obciążenia**
  - **Balansowanie obciążenia (w czasie)**
  - **Zrównoleglenie obciążenia**
- **Identyfikacja zapytań wykonywanych w krytycznym czasie**
  - **Dynamiczny widok v$sql**
  - **Wskazania AWR**
  - **Wskazania ADDM**
- **Korekta planów wykonywania zapytań**
  - **Analiza planu wykonania zapytania**
  - **Korzystanie z Doradców Oracle**
  - **Zastosowanie dobrych praktyk (z rozsądkiem i umiarem) np. indeksy**
- **Sprawdzenie efektów działań**

POLITECHNIKA GDAŃSKA

Przetwarzanie zapytania SQL (powt)

- dokonywane przez proces serwera z udziałem procesów i pamięci instancji
- kroki:
  1. Analiza (parse)
     - poszukiwanie wśród wcześniejszych poleceń
     - kontrola składni,
     - kontrola poprawności nazw obiektów,
     - zakładanie blokad na obiektach używanych w fazie analizy
     - kontrola uprawnień
     - stworzenie planu wykonania zapytania (drzewo czynności)
     - zapisanie zapytania i planu w obszarze współdzielonym
  2. Wykonanie (execute)
     - identyfikacja wybranych rekordów (najpierw poszukiwanie w buforze danych, jeżeli danych tam nie ma, pobranie następuje z plików, kopia pobranych rekordów jest zapisywana w buforze - algorytm LRU Least Recently Used)
     - otwarcie kursora
     - pobranie
     - sortowanie
  3. Sprowadzenie - wysłanie rekordów z kursora do użytkownika
- przy kolejnym wywołaniu tego samego polecenia - tylko sprawdzenie uprawnień

POLITECHNIKA GDAŃSKA

Przetwarzanie zapytania DML (powt)

1. **Analiza - jak w zapytaniach**
2. **Wykonanie**
   - **jeżeli dane nie są w buforze, następuje odczyt danych z plików i zapisanie ich do buforów danych i wycofania;**
   - **założenie blokad na rekordy;**
   - **modyfikacja bufora danych - nowe wartości i bufora wycofania - poprzednie wartości; ("brudny blok" - dane w pamięci różnią się od danych w plikach na dysku).**
   - **rejestracja zmian w buforze dziennika powtórzeń (które bloki w obu buforach są zmienione);**

- **Bufor i segment wycofania**
  - **do odtwarzania w przypadku rollback,**
  - **do pobierania danych przez inne transakcje przed zatwierdzeniem bieżącego polecenia**
  - **odtworzenie w przypadku awarii**

POLITECHNIKA GDAŃSKA

Przetwarzanie polecenia COMMIT (powt)

- mechanizm "fast commit"
- numer SCN - system change number
- kroki:
  1. Zapis o commit z numerem SCN w buforze dziennika powtórzeń
  2. LGWR zapisuje bufor dziennika powtórzeń do pliku
  3. Użytkownik jest informowany o zakończeniu commit
  4. Zapis o zakończeniu commit.
- zapis danych do plików jest niezależny od polecenia commit, może nastąpić przed lub po wykonaniu commit.

POLITECHNIKA GDAŃSKA

Optymalizacja zapytań (1)

- **Podstawy:**
  - **Pomiar czasu wykonania zapytania**
  - **Podgląd planu wykonania zapytania**
- **Pojedyncze zapytanie oderwane od kontekstu całej bazy danych może nie budzić żadnych zastrzeżeń,**
- **Często ważna jest wiedza o danych: rozmiar tabeli, selektywność zapytań**
- **Uwaga! Tworzenie struktur pomocniczych tylko dla danego zapytania ma sens tylko wtedy, gdy zapytanie jest wykonywane często, jego czas wykonania jest krytyczny, a inne metody nie dają poprawy jego wydajności**
  - **Każda dodatkowa struktura oznacza koszty związane z istnieniem danej struktury (spowolnienie zapytań DML i innych zapytań)**

POLITECHNIKA GDAŃSKA

Optymalizacja zapytań (2)

- **Możliwości optymalizacji:**
  - **Zmiana zapytania (w 80% przypadków wystarcza)**
    - **Rozbijanie dużych zapytań na kilka mniejszych (albo odwrotnie)**
    - **Zmiana kolejności złączeń, sposobu złączania**
    - **Zmiana warunków wyszukiwania, sortowania, grupowania**
  - **Zbieranie statystyk dla tabel**
    - **ANALYZE TABLE x COMPUTE STATISTICS;**
    - **Niekiedy działa odwrotnie, niż zamierzono!**
  - **Zastosowanie indeksów (właściwych indeksów)**
  - **Partycjonowanie**
  - **Klastrowanie**
  - **Wskazówki dla optymalizatora kosztowego Oracle**
  - **Wprowadzanie struktur nadmiarowych np. zastosowanie widoków zmaterializowanych**

POLITECHNIKA GDAŃSKA

Pomiar czasu wykonania zapytań

- **Polecenia – SQLPlus:**
  - **SET TIMING ON**
  - **SELECT/INSERT…**
- **Uwaga! Czas pierwszego wykonania zapytania i kolejnych może być znacząco różny**
  - **Dlaczego?**
  - **Który brać pod uwagę przy optymalizacji?**
- **Uwaga! Należy zwracać uwagę na czas (termin) testowania i diagnozowania zapytań (w zależności od aktualnego obciążenia instancji/bazy zmierzony czas może być znacząco różny)**

POLITECHNIKA GDAŃSKA

Plan wykonania zapytania - optymalizatory

- optymalizator zapytań: kosztowy/regułowy
  - optymalizator regułowy – do wersji 9i
  - optymalizator kosztowy – od wersji 7, stopniowo wypierał regułowy
  - bierze pod uwagę zebrane statystyki dla danych zgromadzonych w tabelach
  - uwzględnia konfigurację i wydajność instancji
- optymalizator regułowy
  - plan zapytań formułowany na podstawie analizy składni zapytania
  - prosty w realizacji
  - czuły na sposób sformułowania zapytania
  - nie bierze pod uwagę charakterystyki danych
- optymalizator kosztowy
  - generuje wszystkie możliwe plany wykonania zapytania,
  - ocenia koszt każdego z planów: statystyki danych
  - trudniejszy do zrealizowania
  - mniej czuły na sposób sformułowania zapytania
  - może mieć różne cele optymalizacji

POLITECHNIKA GDAŃSKA

Podgląd planu wykonania zapytania

- **administrator (sys):**
  - **utworzenie roli PLUSTRACE oraz nadanie jej użytkownikowi:**
  - **{ORACLE_HOME}/sqlplus/adm/plustrce.sql;**
- **utworzenie tablicy PLAN_TABLE (użytkownik)**

  START $ORACLE_HOME/rdbms/admin/utlxplan.sql
- **wyświetlenie planu w SQLPlus:**

  SET AUTOTRACE ON [TRACE EXPLAIN]
- **wyświetlenie planu dla zapytania:**

  EXPLAIN PLAN [SET STATEMENT_ID=„x”]

  FOR SELECT …;

  SELECT * FROM PLAN_TABLE; ← problem z formatowaniem

  pakiet DBMS_XPLAN np. metoda DISPLAY ← lepiej
- **Uwaga! Tabelę z planami wykonań należy czyścić**
- **Uwaga! Baza może działać do 25% wolniej**

POLITECHNIKA GDAŃSKA

Plan wykonania zapytania

- **Plan wykonania – wiersz zawiera:**
  - **numer czynności,**
  - **rodzaj operacji**
  - **oszacowany koszt realizacji**
- **Drzewiasta struktura (wiersze wchodzące w skład planu mają strukturę hierarchiczną)**
  - **pierwszy wypisany wiersz to korzeń, który posiada jedno lub wiele dzieci**
  - **dziecko ma zawsze jednego rodzica, któremu przekazuje efekty swojej pracy**
- **Interpretacja:**
  - **plan wykonania czytamy z góry na dół**
  - **plan wykonania czytamy od największego zagłębienia (od najdalszego liścia), czyli od prawej do lewej**
- **Plan zawiera:**
  - **Kolejność operacji na tablicach wymienionych w zapytaniu**
  - **Metoda dostępu do każdej z tablic**
  - **Metoda realizacji złączeń tablic**
  - **Operacje takie jak filtrowanie, sortowanie, agregacje**

POLITECHNIKA GDAŃSKA

Operacje planu wykonania zapytania

- **Około 200 rodzajów operacji, ok. 20 podstawowych**
- **Metody dostępu do tabel (stand-alone operations):**
  - **Table access full**
    - **Pełen przegląd tabeli polegający na sekwencyjnym pobieraniu kolejnych wierszy, aż do osiągnięcia tzw. high-water mark. Jest operacją mocno obciążającą bazę, jednak można zmniejszyć jej negatywne oddziaływanie na wydajność poprzez ustawienie wieloblokowego odczytu oraz równoległego przetwarzania.**
  - **Table access by index rowid**
    - **Pobieranie wierszy po ich fizycznym adresie, zwanym pseudokolumną ROWID. Dostęp ten jest wyjątkowo szybki. Z tego względu indeksy przechowują ten adres jako swego rodzaju wskaźnik na konkretny wiersz.**
  - **Index unique scan**
    - **W wypadku gdy na kolumnę znajdującą się w warunku zapytania jest założony index, optymalizator może zadecydować o jego użyciu w trakcie realizacji zapytania. Index to struktura bazodanowa zawierająca ROWID oraz zawartość kolumny do której jest przypisany, przechowywany w taki sposób by maksymalnie przyspieszyć wyszukiwanie. Oracle wspiera dwa typy indeksów: B-drzewo oraz bitmapowe.**
  - **Index range scan**
    - **Operacja wykonywana w przypadku, gdy zapytanie oparte jest na zakresie wartości lub używa indeksu nieunikatowego. Czynność tego typu wymaga odszukania więcej, niż jednej wartości z indeksu, dlatego są wolniejsze od index unique scan, a mogą się zdarzyć sytuacje, że i od table acces full.**
  - **Hash key access**
    - **Metoda dostępu oparta na funkcji haszującej, czyli matematycznego przekształcenia wyznaczającego lokalizacje danych na podstawie wartości.**

Analiza planu zapytania - przykłady

```
EXPLAIN PLAN
  SET statement_id = 'ex_plan2' FOR
SELECT last_name FROM employees
 WHERE last_name LIKE 'Pe%';

SELECT PLAN_TABLE_OUTPUT
  FROM TABLE(DBMS_XPLAN.DISPLAY(NULL, 'ex_plan2','BASIC'));

------------------------------------------
| Id  | Operation         | Name         |
------------------------------------------
|   0 | SELECT STATEMENT  |              |
|   1 |  INDEX RANGE SCAN | EMP_NAME_IX  |
------------------------------------------
```

```
EXPLAIN PLAN
  SET statement_id = 'ex_plan1' FOR
SELECT phone_number FROM employees
 WHERE phone_number LIKE '650%';

---------------------------------------
| Id  | Operation          | Name     |
---------------------------------------
|   0 | SELECT STATEMENT   |          |
|   1 |  TABLE ACCESS FULL | EMPLOYEES |
---------------------------------------
```

Źródło przykładów: Dokumentacja Oracle 10g Performance Tuning Guide

POLITECHNIKA GDAŃSKA

Rodzaje złączeń w planie zapytania

- **Nested loop**
  - Operacja zwana pętlą zagnieżdżoną łączy rekordy dwóch tabel w pętli. Zazwyczaj w mniejszej stosowany jest pełny przegląd tablicy i dla każdej krotki używany jest indeks drugiej tabeli aby znaleźć pasującą krotkę umożliwiającą połączenie.
  - Operacja ta wymaga co najmniej dwukrotnego dostępu do danych.
  - Bardzo szybka w przypadku tabel o różnych rozmiarach, gdy tabela prowadząca jest mała.
  - W innym przypadku należy bardzo uważać przy wystąpieniu tej operacji.
  - Warto przed wykonaniem zapytania sygnalizującego plan wykonania takiej czynności, zebrać statystyki dla wszystkich tabel biorących udział.
- **Sort-Merge join**
  - Operacja polega na osobnym przetworzeniu i posortowaniu danych z oddzielnych tabel i dopiero na końcu ich złączeniu. Optymalizator często ją wykonuje przy braku dostępnych indeksów.
- **Hash join**
  - Operacja zakłada stworzenie dla większej z tabel pomocniczej tabeli (w pamięci) z wartościami funkcji haszującej na kolumnie podlegającej łączeniu, zaś na drugiej mniejszej wykonanie pełnego przeglądu, w którym będzie wykonywane połączenie w oparciu o utworzone wartości funkcji. Dobrze się sprawdza, gdy tabela haszowana może w całości być utworzona w pamięci.
- **Cartesian join**
  - Konieczność stworzenia produktu kartezjańskiego (każdy wiersz z każdym!)

POLITECHNIKA GDAŃSKA

Przykład - złączenia

- **Przykład:**

```
Rows       Execution Plan
--------   ----------------------------------------------------
      12   SORT AGGREGATE
       2    SORT GROUP BY
   76563     NESTED LOOPS
   76575      NESTED LOOPS
      19       TABLE ACCESS FULL CN_PAYRUNS_ALL
   76570       TABLE ACCESS BY INDEX ROWID CN_POSTING_DETAILS_ALL
   76570        INDEX RANGE SCAN (object id 178321)
   76563      TABLE ACCESS BY INDEX ROWID CN_PAYMENT_WORKSHEETS_ALL
11432983       INDEX RANGE SCAN (object id 186024)
```

- **Interpretacja zależy od liczby wierszy i selektywności operacji!**

POLITECHNIKA GDAŃSKA

Po co analizujemy plan zapytania?

- **Poszukiwanie błędów zapytania (błędów człowieka)**
- **Poszukiwanie błędów optymalizatora (wady Oracle)**
- **Poszukiwanie przestrzeni optymalizacji (możliwości Oracle)**
- **Czego szukamy?**
  - **Pełnego skanowania tablic (full-scan)**
  - **Złączeń typu CARTESIAN JOIN dla dużych tabel**
  - **Nieselektywnego skanowania przedziałowego (unselective range scans)**
  - **Spóźnionych filtrów (late predicate filters)**
  - **Złej kolejności złączeń (wrong join order)**
  - **Złej kolejności w NESTED LOOPS**
  - **Nieefektywnych filtrów**

POLITECHNIKA GDAŃSKA

Przykład – warunek

```
select * from publikacje where upper(kod_typu_publikacji) = 'MONO';
Plan wykonywania
----------------------------------------------------------------------
0      SELECT STATEMENT Optimizer=CHOOSE
1    0   TABLE ACCESS (FULL) OF 'PUBLIKACJE'


select * from publikacje where kod_typu_publikacji in ('MONO', 'mono');
Plan wykonywania
----------------------------------------------------------------------
0      SELECT STATEMENT Optimizer=CHOOSE
1    0   CONCATENATION
2    1     TABLE ACCESS (BY INDEX ROWID) OF 'PUBLIKACJE'
3    2       INDEX (RANGE SCAN) OF 'PUB_TYP_FK_I' (NON-UNIQUE)
4    1     TABLE ACCESS (BY INDEX ROWID) OF 'PUBLIKACJE'
5    4       INDEX (RANGE SCAN) OF 'PUB_TYP_FK_I' (NON-UNIQUE)
```

- **Źródło: Traczyk T.: Wybrane zagadnienia administrowania bazami danych**

POLITECHNIKA GDAŃSKA

Ogólne reguły optymalizacji zapytań

1. Jak najwcześniejsze wykonanie selekcji.
2. Jak najwcześniejsze wykonanie projekcji.
3. Jak najpóźniejsze wykonanie iloczynu kartezjańskiego.
4. Wykonywanie operacji sumy zbiorów rekordów (UNION) po realizacji złączeń wszelkiego typu (zbiór wynikowy operacji sumowania zbiorów nie jest mniejszy od żadnego z jej zbiorów wejściowych).
5. Realizacja ciągu złączeń w kolejności wynikającej z oszacowanego rozmiaru zbioru wynikowego dla każdego złączenia (jako pierwsze zostanie wykonane to złączenie, które da rezultat o najmniejszej liczebności - najbardziej *selektywne*).
   - Jak przewidzieć liczebność zbioru wynikowego?
   - Jak przewidzieć selektywność?

POLITECHNIKA GDAŃSKA

Wspomaganie optymalizatora

- **Zebranie statystyk dla tabel**
  - **ANALYZE TABLE temp2 COMPUTE STATISTICS;**
- **Wskazówki dla optymalizatora**
  - **SELECT /*+ wskazówka */ atr1, atr2 FROM temp2;**
- **Wybrane wskazówki:**
  - **/*+ INDEX(nazwa_indeksu) */ - wymusza użycie indeksu**
  - **/*+ FULL(nazwa_tabeli) */ - wymusza pełne skanowanie tabeli (mimo istnienia indeksów)**
  - **/*+ USE_MERG(tab1, tab2) */ - wymusza użycie MERGE JOIN**
  - **/*+ USE_HASH(tab1) */ - wymusza użycie HASH JOIN**
  - **/*+ USE_NL(tab2) */ - wymusza użycie NESTED LOOPS**
  - **/*+ ALL_ROWS */ - nacisk na otrzymanie wszystkich wyników**
  - **/*+ FIRST_ROWS(n) */ - nacisk na otrzymanie najpierw części wyników**
  - **/*+ PARALLEL(tab1, n) */ - uruchamia n procesów równolegle liczących**
  - **/*+ NO_PARALLEL */ - wymusza brak zrównoleglenia**
  - **/*+ APPEND */ - przy insert – brak zapisów w dzienniku powtórzeń, dokłada dane do nowych bloków, a nie zapełnia miejsc w dotychczasowych**

POLITECHNIKA GDAŃSKA

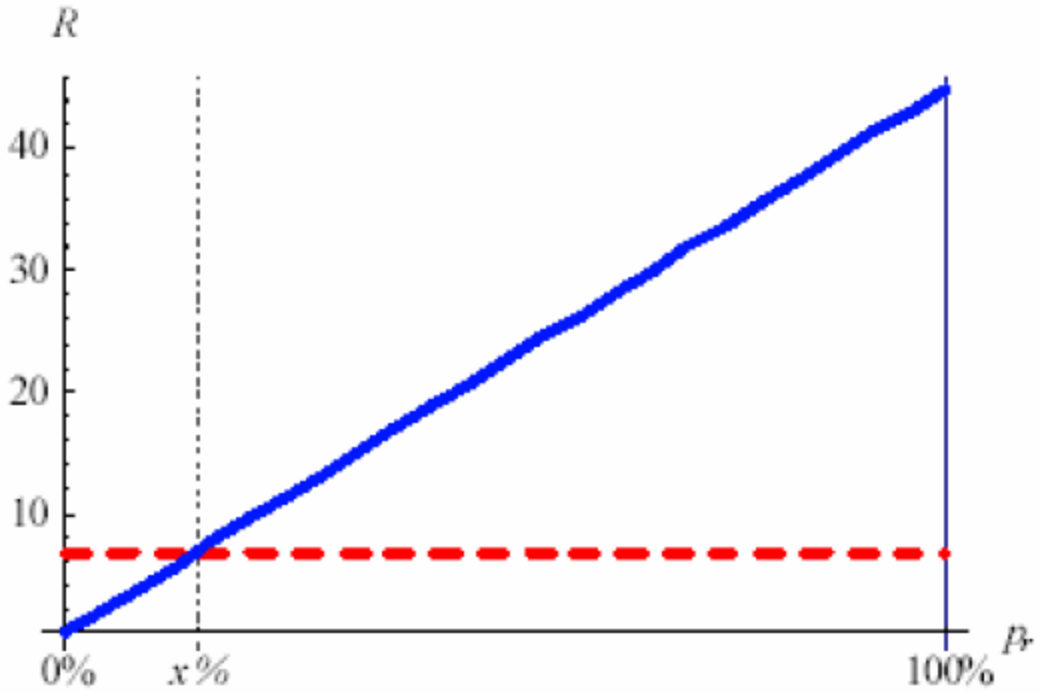
Indeksy

- **Optymalizacja zapytania poprzez:**
  - **Utworzenie**
  - **Usunięcie**
  - **Przebudowę**
- **Indeksy: oparte o B-drzewa lub bitmapowe**
- **Zalecenia:**
  - **Przebudowa często używanych indeksów – jeżeli wydajność zapytań stopniowo słabnie**
  - **Indeksowanie wielu kolumn w jednej tablicy – lepiej mniej indeksów wielokolumnowych**
  - **Dane przechowywać tak, jak najczęściej będą pobierane**

Indeksy – kiedy tworzyć

- Indeks twórz wtedy, gdy dane zapytanie zwraca mniej niż x % wierszy tabeli
- Jeśli zbiór źródłowy będący wynikiem zapytania zawiera tylko jeden wiersz, skanowanie przedziału indeksu trwa krócej, niż skanowanie całej tabeli.
- Jeśli zbiór źródłowy będący wynikiem zapytania zawiera wszystkie wiersze tabeli, skanowanie całej tabeli trwa krócej niż skanowanie przedziału indeksu
- 1<x<100%
- Ile wynosi x?

  (punkt zrównania)



Źródło rysunku: PLOUG, Karwowski, Kiedy tworzymy indeksy?

POLITECHNIKA GDAŃSKA

Indeksy - wskazania

- **Oficjalne punkty zrównania dla różnych wersji Oracle**

Wersja Oracle	Zalecana w dokumentacji Oracle wartość x
5	10-15
6	10-15
7	2-4
8	2-4
8i	15
9i	15

- **Różnice wynikają z problemu dysproporcji między selektywnością wierszy i selektywnością bloków**
- **Selektywność wierszy = liczba wierszy spełniających warunek zapytania/liczba wierszy ogółem**
- **Selektywność bloków = liczba bloków spełniających warunek zapytania/liczba bloków ogółem**

POLITECHNIKA GDAŃSKA

Indeksy – kiedy stosować

- Przykład: zapytanie:

  SELECT id, data FROM temp2 WHERE znacznik='x'

- Ile bloków tabeli zawiera przynajmniej jeden wiersz zwracany w wyniku zadanego zapytania where?
- Tabela temp2:
  - zawiera 1 000 000 wierszy zapisanych w 10 000 blokach bazy Oracle
  - tylko 10 000 wierszy spełnia kryterium znacznik='x‘ (selektywność wierszy 1%)
- Ale:
  - fizyczna dystrybucja wierszy w tabeli dostawy jest taka, że każdy blok tej tabeli zawiera dokładnie jeden wiersz, w którym znacznik='x'.
  - zwrócenie wyników zapytania wymaga dostępu do każdego bloku tabeli dostawy, bez względu na to, czy dla kolumny znacznik użyjemy indeksu
  - Selektywność bloków dla tego warunku wynosi 100%
- Wnioski:
  - W tym przypadku skanowanie całej tabeli będzie trwało krócej niż skanowanie przedziału indeksu, mimo że zapytanie zwraca tylko 1% wierszy
  - Jeśli „interesujące” wiersze są rozproszone jednolicie po tabeli, indeks może się okazać niewydajny nawet w przypadku bardzo „dobrych” wartości, na podstawie których są wybierane wiersze.

POLITECHNIKA GDAŃSKA

Indeksy - wnioski

- indeks może znacznie przyspieszyć działanie bazy nawet w przypadku zapytań na tabeli z jednym wierszem.
  - Badania: w Oracle8i ok. dziesięciokrotne przyspieszenie reakcji na zapytania dual po zbudowaniu indeksu
- decyzję o tym, czy tworzyć indeks, podejmuje się na podstawie punktu zrównania, uzależnionego od selektywności bloków (niekoniecznie wierszy)
- wartości kolumn często są naturalnie pogrupowane lub jednolite. Posiadając taką informację, można podjąć lepszą decyzję o tworzeniu indeksu.
- zapisywanie danych w porządku fizycznym pozwala na zbudowanie bardzo szybkiej bazy.
- Na przydatność indeksu przy skanowaniu przedziału z użyciem operacji table access by rowid wpływają następujące parametry:
  - Wielkość bloku danych bazy Oracle. (większe bloki danych → niższe wartości x)
  - Wielkość bloku indeksu bazy Oracle (większy blok indeksu → mniej operacji LIO → większe wartości x)
  - Wielkość wiersza (większe wiersze → wyższe wartości x)
  - Dystrybucja danych (lepsze fizyczne rozmieszczenie danych tabeli → zmniejszenie wartości x)
  - Parametr db_file_multiblock_read_count
- Można definiować tabele oparte o index (index-organized tables)

POLITECHNIKA GDAŃSKA

Wprowadzanie indeksów

- **Definiowanie kluczy głównych**
  - **ALTER TABLE temp1 ADD CONSTRAINT PK_tmp1 PRIMARY KEY (ID);**
  - **ALTER TABLE temp2 ADD CONSTRAINT PK_tmp2 PRIMARY KEY (ID) USING INDEX;**
- **Definiowanie kluczy obcych**
  - **ALTER TABLE temp1 ADD CONSTRAINT FK_tmp2 FOREIGN KEY(tmp2_ID) REFERENCES temp2(ID);**
- **Definiowanie ograniczeń unikatowości**
  - **ALTER TABLE temp2 ADD CONSTRAINT UQ_tmp2 UNIQUE (atr1, atr2, atr3);**
- **Bezpośrednie tworzenie indeksu**
  - **CREATE INDEX IDX_temp2 ON temp2(atr2);**
- **Monitorowanie użycia indeksu**
  - **ALTER INDEX IDX_temp2 MONITORING USAGE;**
  - **SELECT TABLE_NAME, INDEX_NAME_MONITORING, USED FROM V$OBJECT_USAGE;**
- **Wyłączenie monitorowania**
  - **ALTER INDEX IDX_temp2 NOMONITORING USAGE;**

POLITECHNIKA GDAŃSKA

Partycjonowanie

- Wsparcie dla zarządzania bardzo dużymi tabelami i indeksami poprzez ich dekompozycję na mniejsze, łatwiej zarządzalne części (partycje)
- Zapytania SQL i DML mogą działać tak samo, jak działają dla całych tabel (przezroczyste dla aplikacji)
- Ale można je też przekształcać w celu uzyskania dostępu do konkretnych partycji
- OLTP – poprawa zarządzalności i dostępności
- OLAP – poprawa wydajności i zarządzalności
- Każda partycja tabeli lub indeksu musi mieć taką samą strukturę (nazwy kolumn, typy danych, ograniczenia)
- Ale możliwe jest zróżnicowanie fizycznych parametrów składowania (np. PCTFREE, PCTUSED), każda partycja może być (powinna?) składowana w innej przestrzeni tabel

POLITECHNIKA GDAŃSKA

Zalety partycjonowania

- operacje takie jak: ładowanie danych, tworzenie indeksów, przebudowa indeksów, tworzenie kopii zapasowych i odtwarzanie – na poziomie partycji, a nie całej tabeli
- Technika **partition pruning** w wykonywaniu zapytań pozwala na wykonywanie zapytań tylko dla wybranych partycji
- Niezależność partycji pozwala na zrównoleglenie operacji wykonywanych na różnych partycjach tej samej tabeli (SELECT, DML)
- Można także wykonywać czynności administracyjne (odtwarzanie) na jednej z partycji, nie blokując dostępu do pozostałych partycji tej samej tabeli
- Zwiększenie dostępności dla baz danych krytycznych pod względem bezpieczeństwa lub wydajności poprzez ograniczenie okna czasowego na czynności administrowania, redukcję czasu odtwarzania, redukcję wpływu awarii (failures)
- Przezroczystość dla aplikacji (nie ma potrzeby przepisywania zapytań)

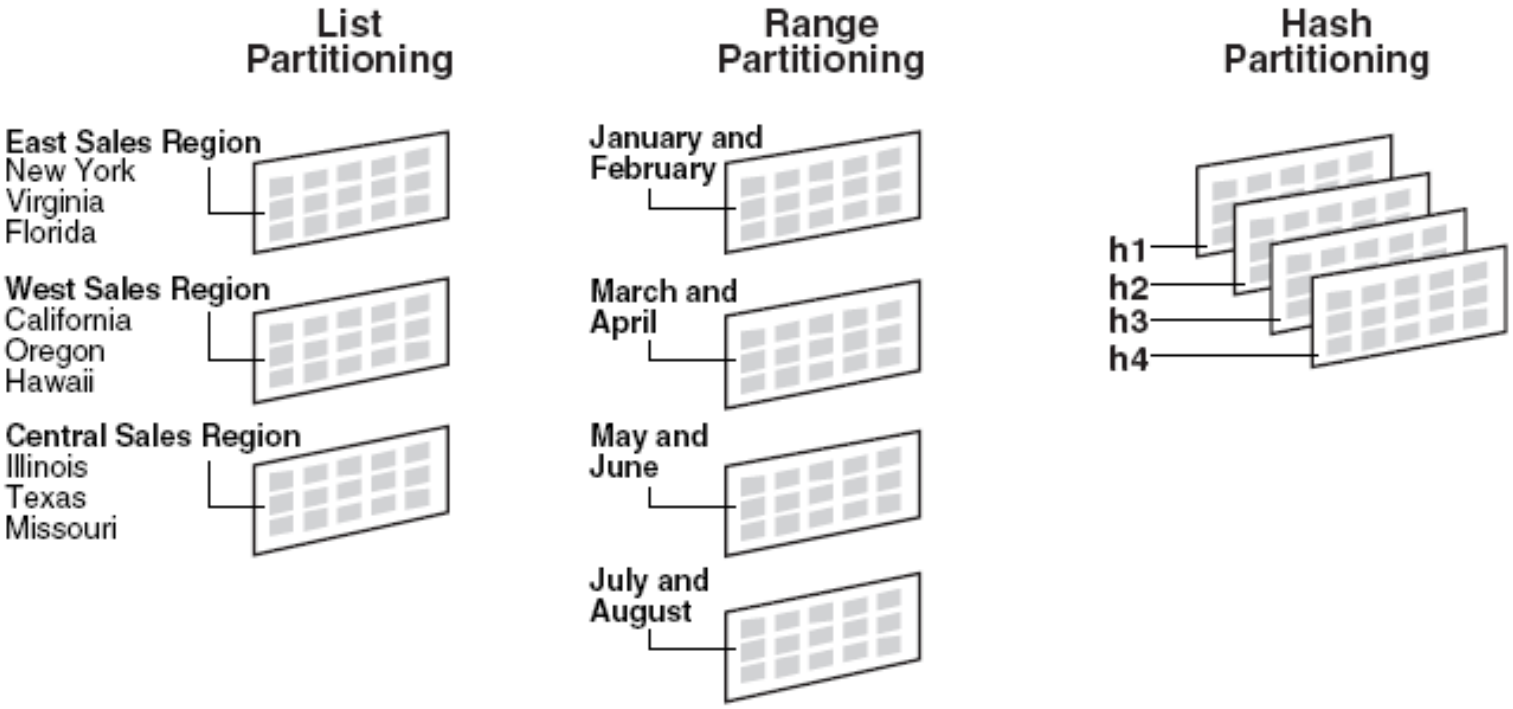
POLITECHNIKA GDAŃSKA

Kryterium partycjonowania

- Każdy wiersz tabeli jest przypisany do jednej konkretnej partycji
- Klucz partycji (Partition Key)
  - Jeden lub zbiór atrybutów, które określają przynależność każdego wiersza do określonej partycji
  - Za pomocą tego klucza Oracle automatycznie wstawia wiersz do konkretnej partycji
- Klucz partycji:
  - Składa się z uporządkowanej listy do 16 kolumn
  - Nie może zawierać pseudokolumn LEVEL, ROWID ani kolumny o typie ROWID
  - Może zawierać kolumny o wartościach pustych NULL
- Nie partycjonowana tablica może mieć partycjonowany albo nie partycjonowany indeks
- Partycjonowana tablica może mieć partycjonowany albo nie partycjonowany indeks
- Jedną tabelę można podzielić na do 64 tysięcy partycji
- Tabela partycjonowana nie może zawierać kolumn o typie LONG lub LONG RAW (ale może zawierać kolumny CLOB i BLOB)

Rodzaje partycjonowania

- Range Partitioning – w oparciu o zakres wartości
- List Partitioning – w oparciu o listę wartości
- Hash Partitioning – w oparciu o funkcję haszującą
- Composite Partitioning – złożone



Źródło rysunku: Dokumentacja Oracle 10g Performance Tuning Guide

POLITECHNIKA GDAŃSKA

Partycjonowanie w oparciu o zakres (RANGE)

- Przypisuje dane do partycji na podstawie przedziałów wartości klucza partycji
- Przedziały są ustalane dla każdej partycji
- Najczęściej dokonywane w oparciu o atrybuty daty i czasu np. miesięczne zestawienia sprzedaży
- Zasady:
  - Każda partycja powinna mieć klauzulę VALUES LESS THAN, wyznaczającą górną granicę zawartości partycji
  - Każda partycja (poza pierwszą) będzie miała również niejawnie podaną dolną granicę (wyznaczoną przez pozostałe partycje)
  - MAXVALUE – podanie może spowodować, że pewien wiersz nie zmieści się w żadnej partycji
  - Co z wartościami NULL?
- Nie wiadomo, jaka będzie dystrybucja wierszy w partycjach
- **Przykład**

```
CREATE TABLE sales_range
   (salesman_id NUMBER(5), salesman_name VARCHAR2(30), sales_amount NUMBER(10),
   sales_date DATE)
   PARTITION BY RANGE(sales_date)
   (  PARTITION sales_jan2000 VALUES LESS THAN
                              (TO_DATE('02/01/2000','MM/DD/YYYY')),
      PARTITION sales_feb2000 VALUES LESS THAN
                              (TO_DATE('03/01/2000','MM/DD/YYYY')),
      PARTITION sales_mar2000 VALUES LESS THAN
                              (TO_DATE('04/01/2000','MM/DD/YYYY')),
      PARTITION sales_apr2000 VALUES LESS THAN
                              (TO_DATE('05/01/2000','MM/DD/YYYY')));
```

POLITECHNIKA GDAŃSKA

Partycjonowanie w oparciu o listę wartości (LIST)

- Dokładniejsze kontrolowanie przydziału do partycji na podstawie podanej listy wartości klucza partycji
- Tylko dla jednego atrybutu
- Bardzo naturalne grupowanie zbiorów danych zgodnie z logiką ich wykorzystania
- **Przykład**

```
CREATE TABLE sales_list
(salesman_id NUMBER(5), salesman_name VARCHAR2(30),  sales_state
    VARCHAR2(20), sales_amount NUMBER(10), sales_date DATE)
PARTITION BY LIST(sales_state)
   (PARTITION sales_west VALUES('California', 'Hawaii'),
   PARTITION sales_east VALUES ('New York', 'Virginia', 'Florida'),
   PARTITION sales_central VALUES('Texas', 'Illinois'),
   PARTITION sales_other VALUES(DEFAULT));
```

- DEFAULT – aby nie specyfikować wszystkich możliwych wartości
- Jeżeli nie ma partycji DEAFULT, wartość nie wymieniona na liście wygeneruje błąd

POLITECHNIKA GDAŃSKA

Partycjonowanie haszujące (HASH)

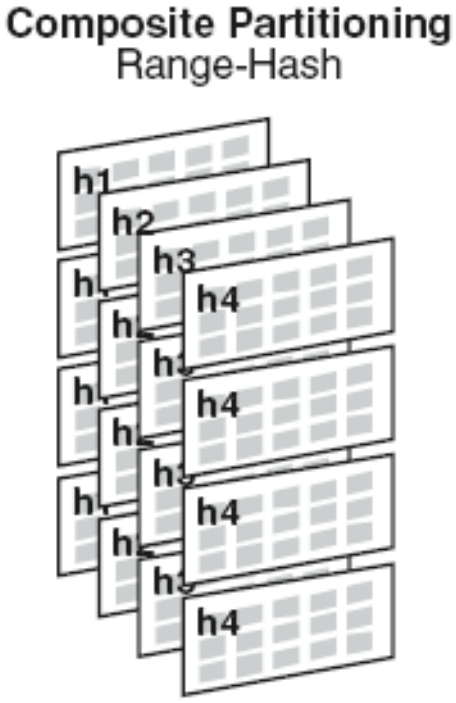
- Dla danych, które nie mają logicznego podziału według zakresu lub wartości,
- Łatwiejsze i prostsze w implementacji
- Kiedy stosować partycjonowanie haszujące:
  - Nie wiadomo jaka ilość danych pasuje do poszczególnych przedziałów
  - Rozmiar partycji zdefiniowanych w inny sposób byłby nieproporcjonalny
  - Nie ma warunków pozwalających utrzymać zrównoważoną dystrybucję między partycjami
  - Przedziałowe mapowanie niepotrzebnie klastrowałoby dane (brak możliwości równoległego przetwarzania zapytań)
  - Istotna jest możliwość zastosowania równoległych zapytań DML, wykluczania partycji albo złączeń partition-wise
- Partycje tego typu mogą być dodawane i defragmentowane
- **Przykład**

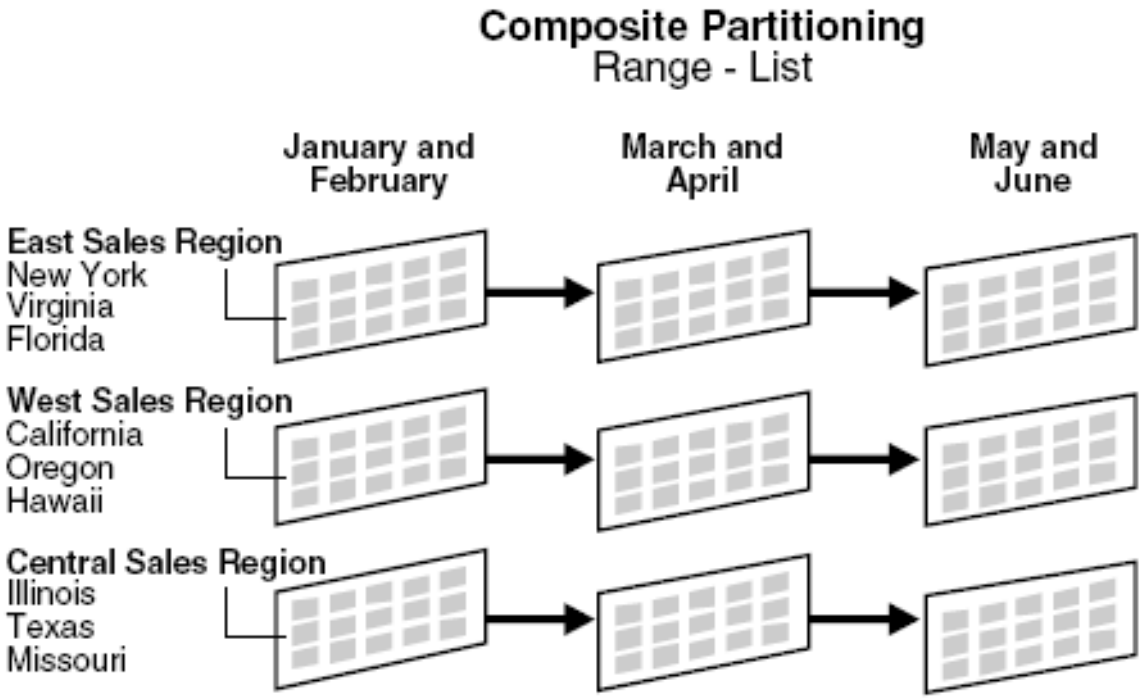
```
CREATE TABLE sales_hash
(salesman_id NUMBER(5), salesman_name VARCHAR2(30), sales_amount NUMBER(10), week_no
    NUMBER(2))
PARTITION BY HASH(salesman_id)
PARTITIONS 4
STORE IN (ts1, ts2, ts3, ts4);
```

- Algorytm round-robin rozłoży wiersze równomiernie w czterech przestrzeniach tabel.

Przykłady partycjonowania złożonego

- Dwie metody: RANGE + HASH lub RANGE + LIST
- RANGE + HASH – grupowanie logiczne w połączeniu z ulepszoną dystrybucją pomiędzy subpartycje dzięki funkcji HASH
- RANGE + LIST – bardzo dokładne kontrola użytkownika nad podziałem danych
- Partycjonowanie złożone wspiera zarządzanie operacjami historycznymi, pozwala na jeszcze lepsze zrównoleglenie zapytań i instrukcji DML, pozwala też na lepszą granularność i dystrybucję danych





Źródło rysunku: Dokumentacja Oracle 10g Performance Tuning Guide

POLITECHNIKA GDAŃSKA

Partycjonowanie a indeksy

Figure 18–6 Local Partitioned Index

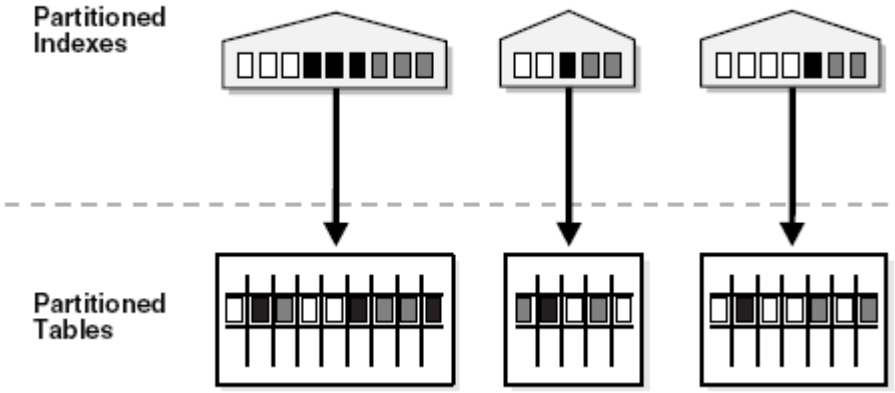


Figure 18–7 Global Partitioned Index

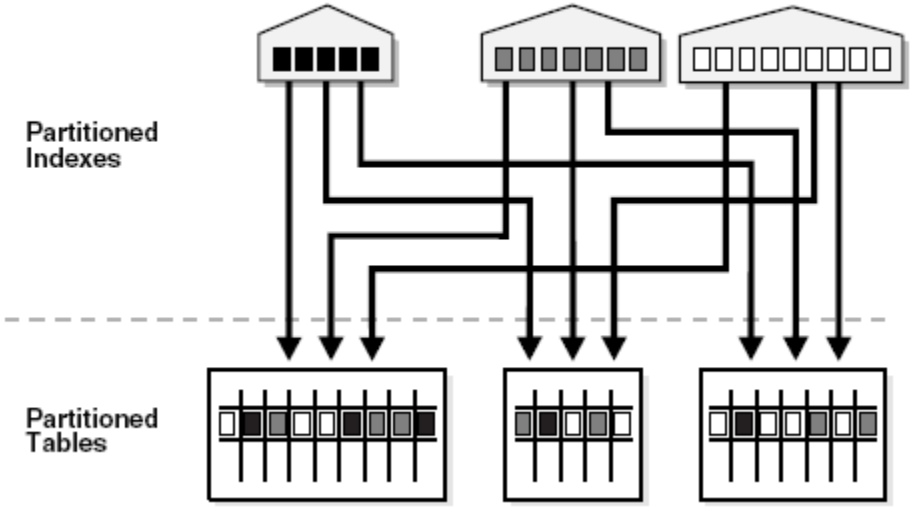
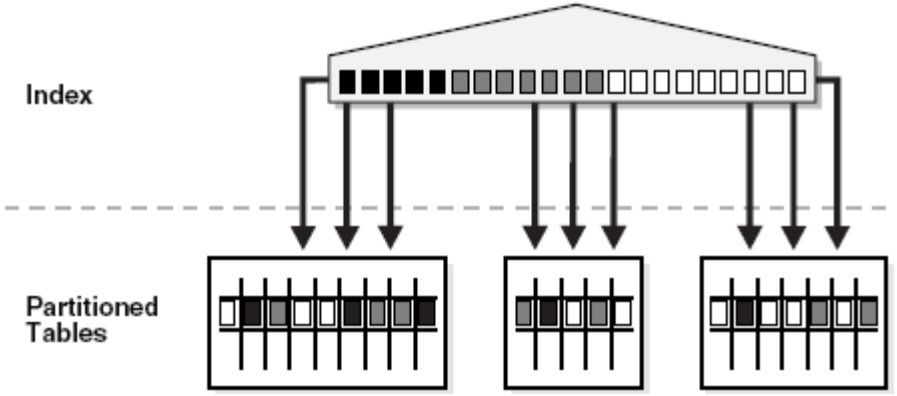


Figure 18–8 Global Nonpartitioned Index



Źródło rysunku: Dokumentacja Oracle 10g Performance Tuning Guide

POLITECHNIKA GDAŃSKA

Kiedy partycjonować?

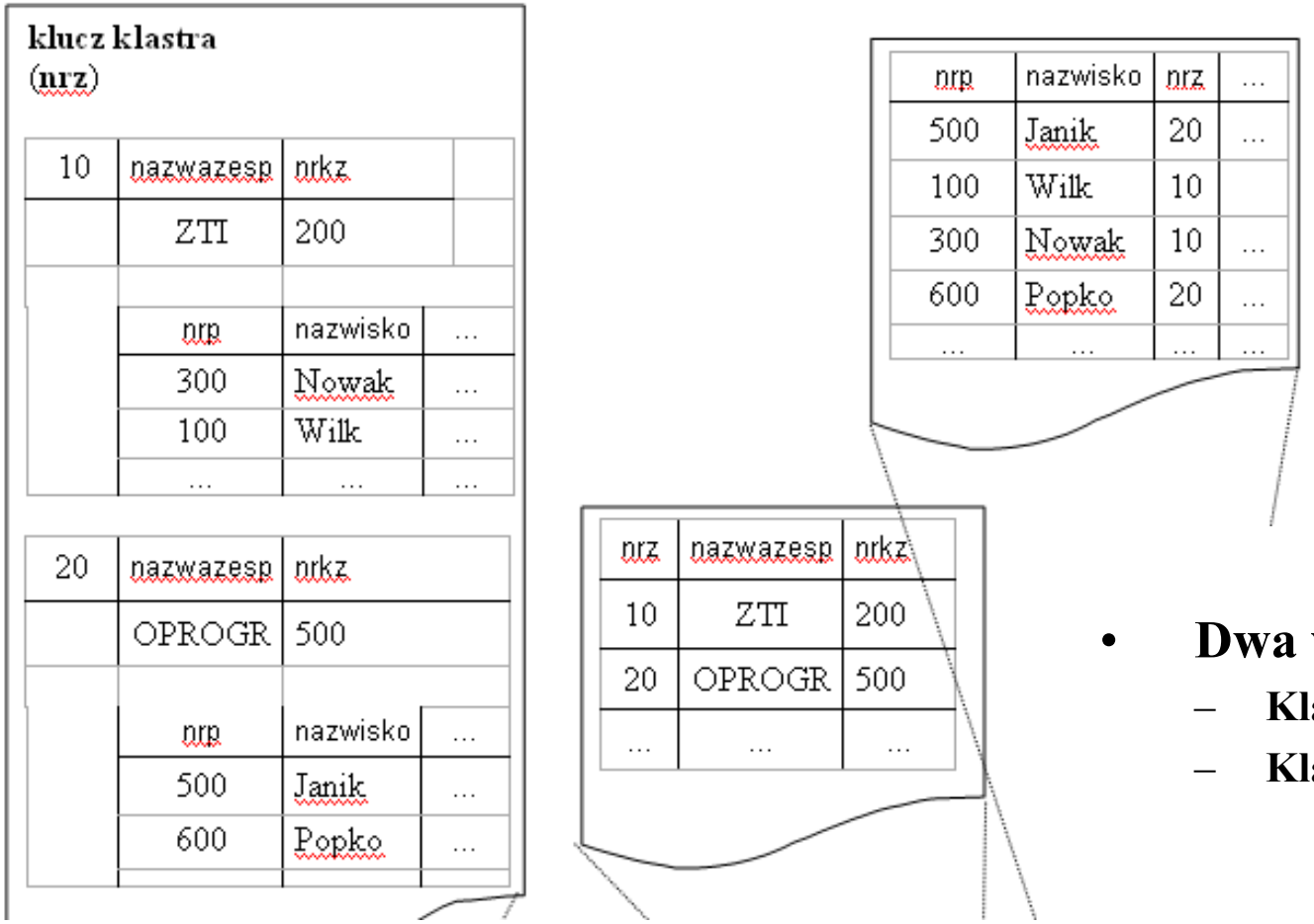
- Zalety:
  - Partition Pruning – wykluczanie partycji z zapytań
  - Partition-wise Joins – złączenia oparte o wybrane partycje w zapytaniach
  - Parallel DML – możliwość zrównoleglenia wykonania zapytań (w tym celu dane partycji muszą być umieszczone w różnych przestrzeniach tabel – w różnych plikach, najlepiej na odrębnych fizycznych dyskach)
- Wskazówki:
  - Dla tabel większych niż 2GB zawsze powinno rozważać się partycjonowanie
  - Dobrze nadają się do partycjonowania tabele zawierające dane historyczne, przy czym nowe wartości powinny być umieszczane tylko w najnowszej partycji np. historia sprzedaży, zamówień, faktur itp., gdzie można zamykać miesiące po ich zakończeniu w partycjach read-only
  - Dla tabel, do których często dostęp (zapytania) dotyczy tylko konkretnych przedziałów wierszy
  - Dla tabel wypełnianych i odpytywanych przez różne aplikacje (różnych użytkowników) w odmienny sposób
  - W bazach OLAP w celu przyspieszenia dużych zestawień, gdy można zastosować równoległe zapytania DML
- Partycjonowanie indeksów:
  - OLTP: Indeksy globalne, a nie lokalne
  - OLAP: Indeksy lokalne, a nie globalne

POLITECHNIKA GDAŃSKA

Klastry w Oracle

- **Grupowanie (ang. clustering), klasteryzacja**
  - **rozmieszczenie blisko siebie na dysku obiektów w jakiś sposób ze sobą powiązanych**
  - **powstają struktury fizyczne określane jako klaster lub grono**
- **Cel:**
  - **zwiększenie efektywności przetwarzania zapytań przez zmniejszenie ilości koniecznych operacji WE/WY koniecznych do odczytania potrzebnych danych**
  - **dotyczy danych, które są składowane rozłącznie, a często wykorzystywane razem → wówczas bardziej efektywne może być ich połączenie w klaster i składowanie w tych samych strukturach fizycznych**
- **Zalety:**
  - **Poprawa wydajności wybranych! zapytań**
- **Wady:**
  - **Zmniejszenie wydajności operacji DML (aktualizacji)**
  - **Zmniejszenie możliwości zarządzania i administrowania**
  - **Zmniejszenie wydajności innych zapytań (analiza struktury obiektów i wzorca dostępu stosowanego przez aplikacje)**

Klastry w Oracle (2)



- **Dwa warianty:**
  - **Klastry indeksowe**
  - **Klastry haszujace**

w jednym segmencie umieszcza się krotki należące do różnych relacji, o ile posiadają one wspólne atrybuty, które będą równe → efektywne operacje złączenia

POLITECHNIKA GDAŃSKA

Klaster indeksowy

- **Klaster indeksowy**
  - **Do znalezienia bloku danych w klastrze indeksowym, potrzebna jest jedna lub więcej operacji WE/WY związanych z indeksem klastra.**
  - **Aby zlokalizować rekord zawierający dany klucz klastra, najpierw sprawdzany jest indeks i odczytywany odpowiadający identyfikator rekordu, a dopiero w następnym kroku pobierany jest rekord.**
- **Zalety i wady klasterów indeksowych:**
  - **Klastry indeksowe umożliwiają szybszy dostęp do wielu rekordów o tej samej wartości klucza klastra.**
  - **Szybsze są operacje wybierania, aktualizacji i usuwania.**
  - **Pojedyncza operacja wstawiania jest nieco wolniejsza niż w przypadku tabeli samodzielnej.**
  - **Ładowanie danych do tabeli jest znacznie wolniejsze.**
  - **Wstawianie rekordu musi być poprzedzone odczytaniem indeksu klastra w celu odszukania bloku, w którym powinien znaleźć się wiersz.**
  - **Nie jest możliwe ładowanie danych do tabel w klastrze bez indeksu.**
  - **Indeks klastra musi być częściej modyfikowany, więc generowanych jest więcej informacji wycofania i dziennika powtórzeń.**
  - **Bloki mogą nie mieścić się w SGA i wtedy muszą być odczytywane wielokrotnie.**
  - **Klastry indeksowe potencjalnie oszczędzają przestrzeń potrzebną na indeksy, gdyż nie potrzebują dodatkowej przestrzeni.**

POLITECHNIKA GDAŃSKA

Klaster haszujący

- **Klaster haszujący**
  - W przypadku zastosowania klastra haszującego, dane przechowywane są w oparciu o wynik funkcji numerycznej, która określa blok danych w klastrze w oparciu o wartość klucza klastra
  - aby znaleźć blok danych w klastrze z funkcją haszującą, klucz klastra wystarcza do określenia jego położenia
  - Klaster haszujący może być użyty dla pojedynczej tabeli, co umożliwia pobranie danych w jednej operacji WE/WY (w odróżnieniu np. od wielokrotnych operacji WE/WY wykonywanych do pobrania tych samych danych w przypadku użycia indeksu opartego na B-drzewie).
  - Klucz klastra i blok danych, w którym będzie przechowany są bezpośrednio powiązane. W trakcie tworzenia klastrów musi być przydzielona przestrzeń, która będzie przez nie wykorzystana. W ten sposób jego rozmiar i liczba rekordów powinny być dokładnie znane.
  - Haszowanie przynosi największe korzyści dla tabel posiadających unikatowe wartości klucza klastra (ze względu na rozkład danych) oraz takich, dla których podstawowymi zapytaniami są zapytania o równość klucza klastra
- **Zalety i wady klastrów haszujących:**
  - Klastry haszujące dostarczają bardzo szybkiej metody dostępu do pojedynczego rekordu lub do wielu rekordów o tej samej wartości klucza klastra.
  - Szybsze są operacje wybierania, aktualizacji i wyszukiwania.
  - Pojedyncze wstawianie nie jest wolniejsze niż w przypadku zwykłej (niepoklastrowanej) tabeli.
  - Ładowanie danych do tabeli jest nieco wolniejsze.
  - Bloki mogą nie mieścić się w SGA i będą musiały być odczytywane wielokrotnie.
  - Przestrzeń musi być zarezerwowana przy tworzeniu klastra, co sprawia, że klastry haszujące mają zastosowanie tylko wtedy, gdy liczba rekordów nadmiernie nie wzrasta.
  - Istnieje potencjalne ryzyko przepełnienia, jeśli funkcja haszująca (np. zdefiniowana przez użytkownika) nie rozkłada danych równomiernie.
  - Klastry haszujące wymagają zwykle więcej przestrzeni niż zwykła tabela szczególnie, jeżeli liczba i/lub rozmiar rekordów z tym samym kluczem klastra jest bardzo zmienna.
  - Niemożliwe jest bezpośrednie ładowanie do tabeli.

POLITECHNIKA GDAŃSKA

Procedura tworzenia klastra

1. stworzyć klaster,
2. stworzyć tabele w klastrze, pamiętając jednocześnie, że nie należy specyfikować parametrów składowania gdyż zostały one określone dla klastra,
3. utworzyć indeks na kluczu klastra,
4. wprowadzić dane do tabel znajdujących się w klastrze.

- **W klastrze można zmodyfikować:**
  - parametry zarządzania alokacji ekstentów (INITIAL, NEXT, MINEXTENTS, MAXEXTENTS, PCTINCREASE),
  - parametry wykorzystania przestrzeni bloku danych (PCTFREE i PCTUSED),
  - średni rozmiar klucza klastra (SIZE)
  - ustawienia zapisów transakcji w blokach danych (INITRANS i MAXTRANS).

POLITECHNIKA GDAŃSKA

Klastry – polecenia SQL

- **Tworzenie klastra haszującego:**

  CREATE CLUSTER ***nazwa klastra***
  HASHKEYS ***liczba kluczy klastra***
  HASH IS ***kolumna będąca kluczem klastra***
  SIZE ***rozmiar przestrzeni potrzebna do przechowania pojedynczego klucza***
  TABLESPACE ***nazwa przestrzeni tabel*****;**

- **Tworzenie klastra indeksowego:**

  CREATE CLUSTER ***nazwa klastra*** **(*pole1 typ, pole2 typ, ...*)**
  [*opcjonalne ustawienie parametrów wykorzystania bloku*]
  [TABLESPACE ***nazwa_ przestrzeni*****]**
  [SIZE rozmiar [K/M]]
  STORAGE (*opcjonalne ustawienie parametrów pamięciowych*);

- **Tworzenie indeksu klastra:**

  CREATE INDEX ***nazwa indeksu*** **ON CLUSTER** ***nazwa klastra*****;**

- **Modyfikowanie klastra:**

  ALTER CLUSTER ***nazwa klastra*** **[SIZE ...] [STORAGE (...)];**

- **Usuwanie klastra:**

  DROP CLUSTER ***nazwa klastra;***

- **Tworzenie tabeli w klastrze:**

  CREATE TABLE ***nazwa_tabeli*** **(atrybut1 typ1, ...)**
  CLUSTER ***nazwa_klastera*** **(*kolumna1 klucza, kolumna2 klucza, ...*);**

POLITECHNIKA GDAŃSKA

Optymalizacja zapytań – należy:

- Zbierać statystyki dla tabel i okresowo je aktualizować,
- Unikać stosowania wielu prywatnych synonimów i aliasów,
- Śledzić plany wykonania zapytań, tylko jeżeli sprawiają problemy,
- Uważnie tworzyć indeksy (na właściwych kolumnach, właściwego typu: B-drzewa, bitmapowe, funkcyjne!) – strategia indeksowania,
  - Okresowo przebudowywać indeksy przy szybkozmiennych danych,
  - Gdy duża liczba kolumn w jednej tabeli ma być indeksowana, efektywniejsze jest stworzenie mniej indeksów, ale wielokolumnowych
- Przechowywać dane w takich strukturach logicznych, w jakich najczęściej będą wykorzystywane (niekoniecznie normalizacja struktury bazy),
  - Ostrożnie wprowadzać dodatkowe struktury, takie jak partycje, klastry, index-organized tables,
- Używać widoków zmaterializowanych z opcją ENABLE QUERY REWRITE,
- Maksymalnie przetwarzać dane po stronie serwera (szybsze, niż przesyłanie dużych wolumenów danych przez sieć) – bloki PL/SQL zamiast zbiorów oddzielnych zapytań,
  - Uruchamiać anonimowe bloki PL/SQL zamiast składowanych,
- Ograniczać użycie mechanizmów, które stanowią dodatkowy narzut na wykonywanie zapytań np. wyzwalacze.