



**Wydział Elektroniki
i Technik Informatycznych**

POLITECHNIKA WARSZAWSKA

Bazy Danych 1

edycja 21L

Laboratorium 7

Przebieg laboratorium

 Porównywanie wt. NULL

 Widoki

 Sekwencje

 Synonimy

 [VIEWS - Oracle docs](#)  [Sequences - Oracle docs](#)  [Synonyms - Oracle docs](#)

Porównywanie wt. NULL

- Jedynie operatory **IS NULL**, **IS NOT NULL** przy porównywaniu z wt. NULL zwracają wt. logiczne TRUE lub FALSE. **Zastosowanie pozostałych operatorów zwraca wt. logiczną UNKNOWN.**
- Przykład 1: założmy, że zmienna A ma wartość 10:
 - $A = \text{NULL} \rightarrow \text{UNKNOWN}$
 - $A > \text{NULL} \rightarrow \text{UNKNOWN}$
 - $A < \text{NULL} \rightarrow \text{UNKNOWN}$
 - $A \text{ IS NULL} \rightarrow \text{FALSE}$
 - $A \text{ IS NOT NULL} \rightarrow \text{TRUE}$
- Przykład 2: założmy, że zmienna A ma wartość NULL:
 - $A = \text{NULL} \rightarrow \text{UNKNOWN}$
 - $A \text{ IS NULL} \rightarrow \text{TRUE}$

Rezultaty operacji logicznych

	TRUE	FALSE	UNKNOWN
NOT	FALSE	TRUE	UNKNOWN

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

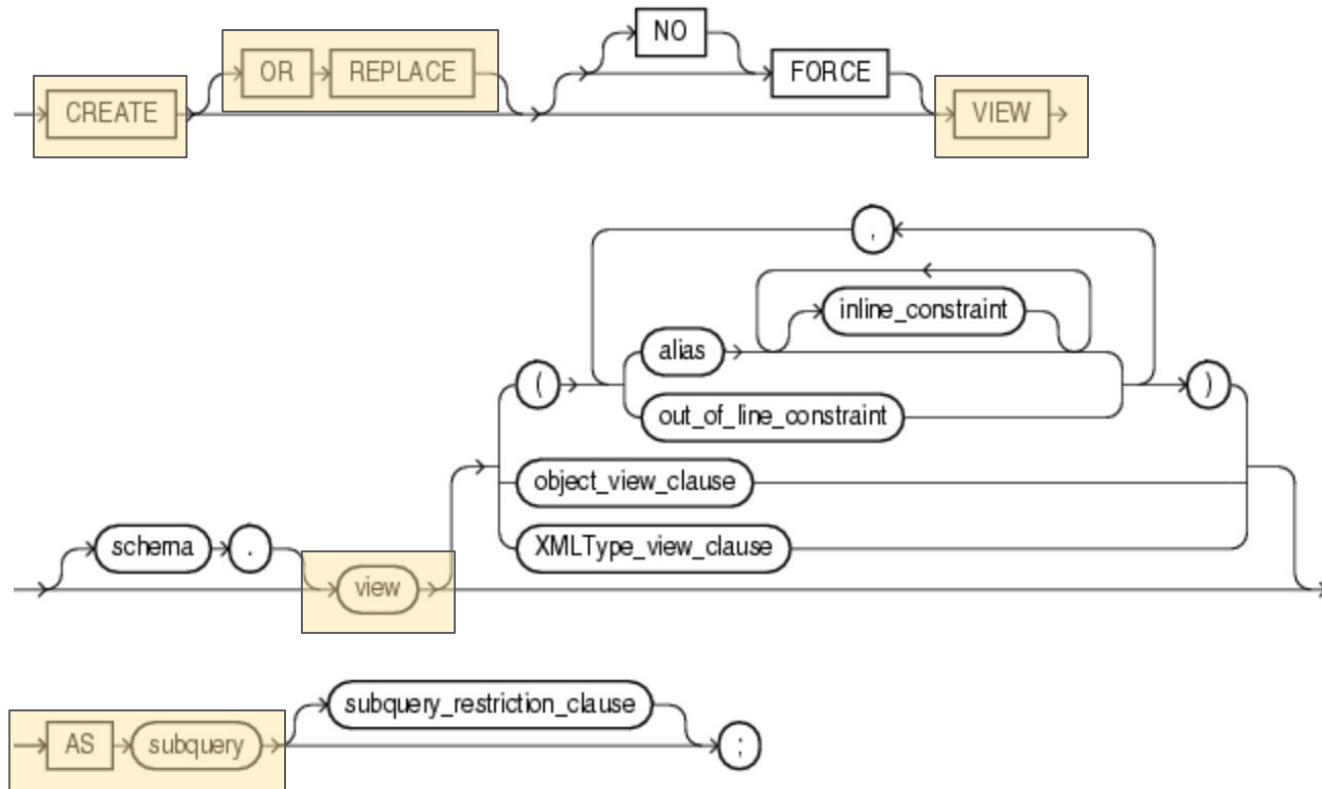
Widoki

- Widoki (perspektywy) są logicznymi reprezentacjami tabeli lub złączeń wielu tabel (tzw. tabel bazowych).
 - Tabele bazowe mogą być zarówno rzeczywistymi tabelami, jak i widokami.
- Dlaczego stosujemy widoki:
 - ze względów bezpieczeństwa (nie udostępniamy całego schematu bazy danych).
 - bo nie chcemy powtarzać skomplikowanych zapytań do BD wiele razy.
 - widoki (zmaterializowane) mogą skrócić czas oczekiwania na złożone zapytania zwracające wiele wierszy.

Widoki - dozwolone operacje

- Do widoków można zawsze wydawać zapytania SELECT.
- Za pomocą niektórych widoków można wprowadzać, modyfikować i usuwać dane w tabelach bazowych poleceniami INSERT, UPDATE i DELETE.
- Modyfikowanie danych za pomocą widoków nie jest możliwe w następujących przypadkach:
 - Widok jest zdefiniowany jako WITH READ ONLY.
 - Dla widoków zdefiniowany jako WITH CHECK OPTION możliwe jest modyfikowanie jedynie tych wierszy, które spełniają warunki klauzuli WHERE z definicji widoku.
 - Definicja widoku nie uwzględnia kolumn NOT NULL z tabeli bazowej.
 - Widok jest zdefiniowany na podstawie niektórych zapytań SELECT np. posiadającego klauzulę GROUP BY.
 - ...

Widoki - składnia



Widoki - przykład

- W definicji widoku powinno pojawić się zapytanie, które odwoła się to tabel lub innych widoków.
- Przykład: *Utwórz widok, który będzie zawierać imię, nazwisko, zarobki oraz nazwę zajmowanego stanowiska dla wszystkich pracowników.*

```
CREATE VIEW emp_pos_names  
AS
```


Utworzenie widoku

```
SELECT e.name as name, surname, salary, p.name as position  
FROM employees e JOIN positions p USING (position_id);
```

```
SELECT position, COUNT(*)  
FROM emp_pos_names  
WHERE salary > 2100  
GROUP BY position  
HAVING COUNT (*) >= 2  
ORDER BY 2 DESC;
```

Wydanie zapytania
z wykorzystaniem
widoku

Script Output x Query Result x

 All Rows Fetched: 43 in 0,093 seconds

E_NAME	SURNAME	SALARY	POSITION
1 Piotr	Maciejewski	3000	Programista
2 Piotr	Janowski	3000	Konsultant
3 Andrzej	King	1500	Kadrowy
4 Lex	Kochhar	2000	Manager marketingowy
5 Alexander	Hunold	3000	Reprezentant
6 Tracy	Hunold	2000	Techniczny
7 David	Ernst	4500	Wdrozeniowiec

Widoki - modyfikowanie danych

→ Przykład: Widok *emp_view* przechowuje id, imie, nazwisko oraz datę urodzenia danego pracownika. Wykorzystaj ten widok do dodania nowego pracownika do tabeli *employees*.

```
CREATE VIEW emp_view AS  
    SELECT employee_id, name, surname, birth_date  
    FROM employees;
```

```
INSERT INTO emp_view VALUES (301, 'Piotr', 'Kowalski', SYSDATE);  
UPDATE emp_view SET name = 'PIOTR' WHERE name LIKE 'Piotr';
```

Widoki - ograniczenie na modyfikowanie danych

- Oprócz tego, że można tworzyć widoki tylko “do odczytu”, to możliwe jest także definiowanie widoków, które pozwolą na modyfikowanie tych wierszy w tabeli bazowej, do których odwołuje się definicja widoku.
- Przykład: *Widok emp_view_names udostępnia kolumny employee_id, name, surname dla pracowników, których imiona rozpoczynają się od litery ‘P’, ‘R’, ‘M’:*

```
CREATE VIEW emp_view_names AS
  SELECT employee_id, name, surname, birth_date FROM employees
  WHERE name like 'P%' OR name like 'R%' OR name like 'M%'
  WITH CHECK OPTION;
```

- Ponieważ widok został stworzony z opcją **WITH CHECK OPTION**, to możliwe jest za jego pomocą dodawanie, modyfikowanie i usuwanie tylko tych wierszy tabeli Employees, które spełniają warunek:

```
WHERE name like 'P%' OR name like 'R%' OR name like 'M%'
```

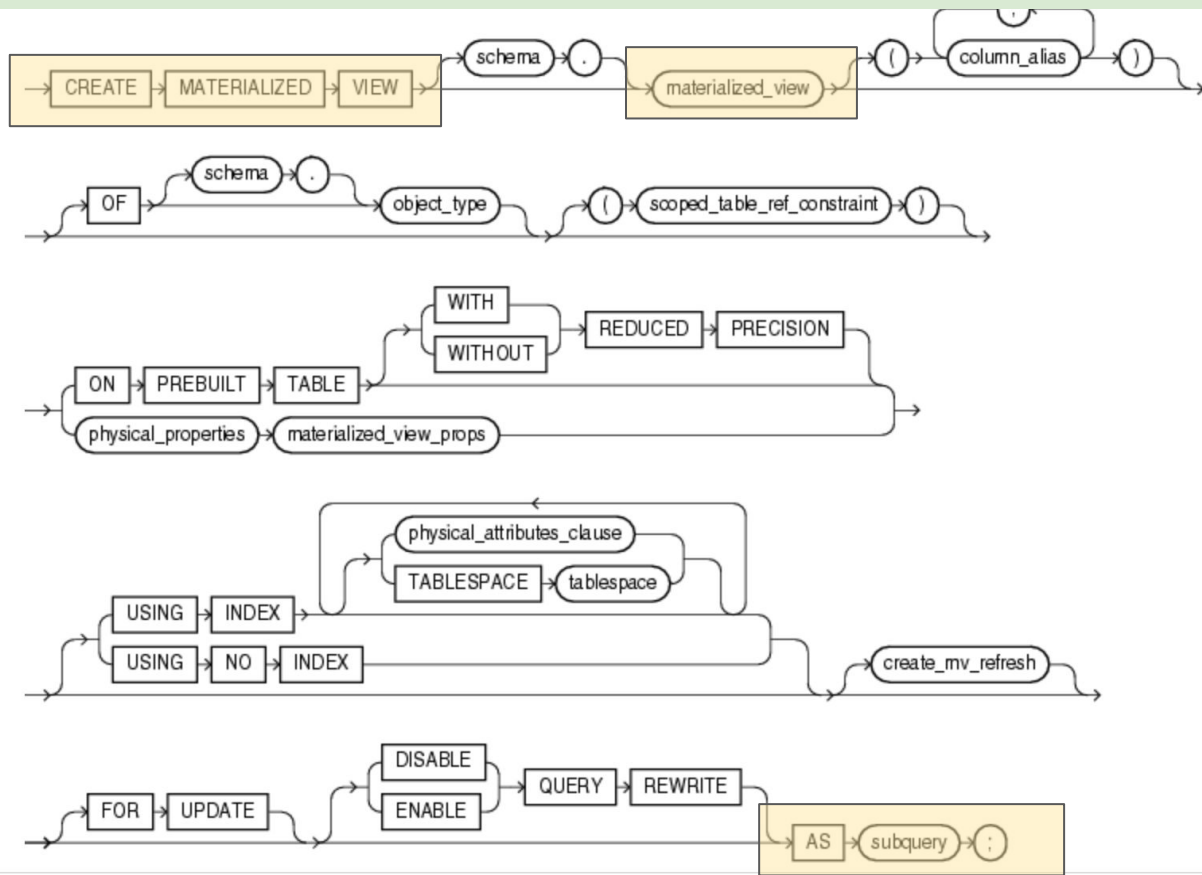
Widoki - ćwiczenia

1. Zdefiniuj widok, który będzie zwracać imię, nazwisko, zarobki, nazwę stanowiska, nazwę departamentu oraz imię i nazwisko managera dla wszystkich pracowników w tabeli employees.
2. Zdefiniuj widok typu WITH CHECK OPTION przechowujący id stanowisk (position_id) oraz nazwę, minimalne zarobki wszystkich stanowisk rozpoczynających się od litery 'P', 'K' lub 'M'. Następnie spróbuj zwiększyć minimalne zarobki dla stanowiska 'Rekruter' o 1000 i przeanalizuj komunikat błędu.
3. Wykonaj polecenia DROP VIEW, aby usunąć jeden z wcześniej utworzonych widoków.

Widoki zmaterializowane

- Rezultaty zapytań definiujących widoki w poprzednich przykładach nie są na stałe przechowywane w bazie danych. Oznacza to, że każdorazowe odwołanie do widoku powoduje na nowo wykonanie zapytania definiującego ten widok.
- W przypadku widoków zmaterializowanych rezultaty zapytań definiujących widoki są na stałe przechowywane w bazie danych.
- Należy odświeżać widoki zmaterializowane, aby zobaczyć aktualne dane w tabelach bazowych (automatycznie lub na żądanie).

Widoki zmaterializowane - składnia



Widoki zmaterializowane - przykład

→ Przykład: Widok zmaterializowany `emp_pos_dept_MView` przechowuje imię, nazwisko pracownika, jego zarobki, nazwę stanowiska oraz nazwę departamentu w którym pracuje:

```
CREATE MATERIALIZED VIEW emp_pos_dept_mview AS
```

```
SELECT e.name name, e.surname, e.salary, p.name position, d.name department  
FROM employees e JOIN positions p USING(position_id)  
JOIN departments d USING (department_id);
```

```
SELECT * FROM emp_pos_dept_mview;
```



The screenshot shows a database query result with 4 rows. The columns are NAME, SURNAME, SALARY, POSITION, and DEPARTMENT. The data is as follows:

	NAME	SURNAME	SALARY	POSITION	DEPARTMENT
1	Piotr	Maciejewski	3000	Programista	Administracja
2	Jose Manuel	Sciarra	3036	Grafik	Administracja
3	Piotr	Janowski	3000	Konsultant	Marketing
4	Andrzej	King	1500	Kadrowy	Marketing

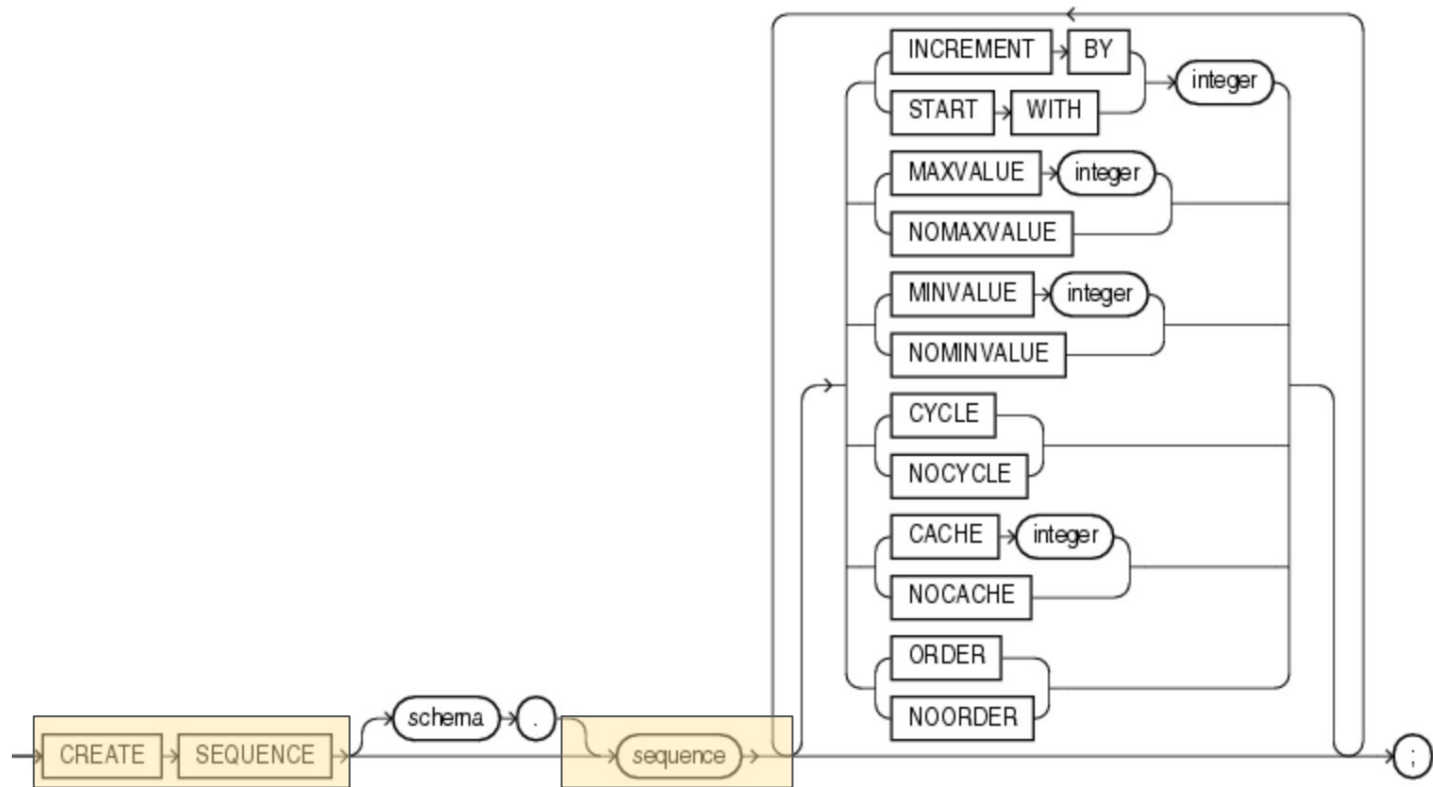
Widoki zmaterializowane - ćwiczenia

1. Zdefiniuj widok, zmaterializowany, który będzie przechowywał imię i nazwisko kierowników i liczbę jego podwładnych.
2. Zdefiniuj widok zmaterializowany przechowujący informacje o sumie budżetów (`estimated_budget`) projektów prowadzonych przez dany departament.
3. Do widoku stworzonego w powyższym poleceniu dodaj kolumnę z informacją o procentowym udziale sumy budżetów projektów w rocznym budżecie danego departamentu. Zaokrąglij procentowy udział do 2 miejsc po przecinku. Posortuj malejąco względem tego procentowego udziału.
4. Wykonaj polecenia `DROP MATERIALIZED VIEW`, aby usunąć jeden z wcześniej utworzonych widoków.

Sekwencje

- Sekwencja jest obiektem bazy danych, który pozwala na iteracyjne uzyskiwanie liczb całkowitych.
- Sekwencje są na ogół wykorzystywane do generowania wartości kluczy głównych.

Sekwencje - składnia



Sekwencje - opcje definiowania

- Istnieje szereg parametrów pozwalających określić charakterystykę sekwencji. Niektóre z nich zostały pokazane w tabeli:

INCREMENT BY liczba1	Określa, o ile mają być inkrementowane lub dekrementowane kolejne wartości sekwencji.
START WITH liczba1	Określa początkową wartość sekwencji.
MAXVALUE liczba1	Określa maksymalną wartość sekwencji.
NOMAXVALUE	Brak maksymalnej wartości sekwencji (w rzeczywistości maks. to liczba: $10^{28} - 1$)
MINVALUE liczba1	Określa minimalną wartość sekwencji.
NOMINVALUE	Ustawia wartość minimalną wartość 1 dla sekwencji rosnących i $-10^{28} - 1$ dla sek. malejących.
CYCLE	Wskazuje, że po osiągnięciu maksymalnej (minimalnej) wartości sekwencja zacznie generować liczby począwszy od minimalnej (maksymalnej) wartości.
NOCYCLE	Po osiągnięciu wartości maksymalnej (minimalnej) sekwencja przestaje generować wartości.

Sekwencje - przykład

- *Zdefiniuj sekwencję o nazwie `int_seq1`, która będzie generowała wartości całkowite z przedziału $100 \div 1000$:*

```
CREATE SEQUENCE int_seq1 start with 100 maxvalue 1000;
```

- Po zdefiniowaniu sekwencji możliwe jest odwoływanie się do jej aktualnej wartości (CURRVAL) oraz następnej wartości (NEXTVAL):

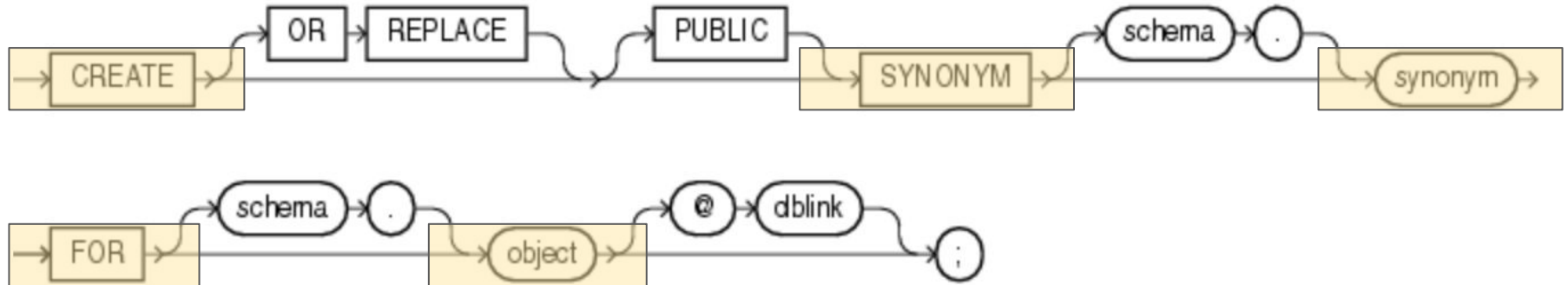
```
SELECT int_seq1.NEXTVAL FROM dual;
```

```
SELECT int_seq1.CURRVAL FROM dual;
```

Synonimy

- Synonimy są alternatywnymi nazwami nadawanymi obiektom bazy danych.
- Możliwe jest tworzenie synonimów m.in. dla następujących obiektów BD:
 - tabele, widoki, inne synonimy, pakiety, procedury i funkcje języka PL/SQL, widoki zmaterializowane, sekwencje, obiekty zdefiniowane przez użytkownika.
- Można zdefiniować synonimy:
 - *prywatne* - dostępnych jedynie dla właściciela oraz użytkowników, którzy mają uprawnienia do tabel bazowych
 - *publiczne* - widoczne dla wszystkich użytkowników bazy danych (jednakże wymagane jest nadanie dodatkowych uprawnień do takich obiektów).
- Korzystając z synonimu można wykonywać wszystkie operacje DML.
- Dlaczego stosujemy synonimy:
 - maskowanie schematu i nazwy tabeli (wygoda, czytelność)
 - wprowadzenie niezależności między aplikacją kliencką (konsument danych z tabeli) a fizyczną lokalizacją danych

Synonimy - składnia



Synonimy - przykład

- Przykład: zdefiniuj synonim prywatny o nazwie *emps* dla tabeli *employees*:

```
CREATE SYNONYM emps FOR employees;
```

- Przykład: zdefiniuj synonim prywatny o nazwie *seq_syn* dla sekwencji zdefiniowanej na slajdzie 16:

```
CREATE SYNONYM seq_syn FOR int_seq1;
```

- Przykład: zdefiniuj synonim prywatny o nazwie *MView_1* dla widoku zmaterializowanego zdefiniowanego na slajdzie 11:

```
CREATE SYNONYM MView_1 FOR emp_pos_dept_mview;
```

Sekwencje, synonimy - ćwiczenia

1. Zdefiniuj sekwencję, która: (i) będzie posiadała minimalną wartość 10; (ii) rozpocznie generowanie wartości od 12; (iii) będzie posiadała maksymalną wartość 17; (iv) będzie cykliczna. Następnie wygeneruj kilkanaście wartości za pomocą tej sekwencji i obserwuj rezultaty.
2. Zdefiniuj sekwencję, która będzie generowała malejąco liczby parzyste z przedziału $100 \div 0$.
3. Nadaj synonim dla dowolnej z dwóch poprzednio zdefiniowanych sekwencji i pobierz z niej wartość za pomocą synonimu.



Praca domowa

1. Utwórz widok zawierający złączenia kilku tabel (np. employees, positions, departments, addresses) i spróbuj go wykorzystać do wprowadzenia nowych danych do tych tabel. Jaki komunikat błędu zaobserwowałeś?
Wyszukaj w dokumentacji możliwe rozwiązanie tego problemu.
2. Zdefiniuj widok łączący tabele countries, regions i addresses. Następnie wykorzystując złączenie pomiędzy tym widokiem, a tabelą departments wyświetl kraje oraz regiony położenia wszystkich departamentów.
3. Wyszukaj informacje w Internecie lub w dokumentacji bazy Oracle, w jaki sposób możliwe jest automatyczne wykorzystanie sekwencji do generowania wartości klucza głównego przy dodawaniu danych do tabeli.



Wydział Elektroniki
i Technik Informatycznych

POLITECHNIKA WARSZAWSKA

Bazy Danych 1

edycja 21L

Laboratorium 11

Wprowadzenie i przebieg laboratorium



Wydajność w systemach bazodanowych

Co można stroić?

- Serwer bazy danych
- Schemat bazy danych
- Aplikację
 - strukturę aplikacji (gruby klient, cienki klient)
 - sposób współpracy z bazą danych
 - zapytania SQL



[Oracle database performance - docs](#)

Strojenie serwera bazy danych

- Strojenie SGA (System Global Area)
- Strojenie zasobów dyskowych
- Klastrowanie bazy danych

 [Oracle database architecture - docs](#)

Strojenie schematu bazy danych

- Normalizacja schematów relacyjnych
- Dobór typów danych
- Indeksowanie danych
- Klastrowanie tabel *(do samodzielnego opracowania)*
- Partycjonowanie - *operacja dzielenia dużej tabeli na mniejsze części. Zapytania SQL i złączenia są szybsze, ze względu na zmniejszoną liczbę operacji we/wy. Partycjonowanie może być poziome (na podstawie zakresu wartości) lub pionowe (oddzielenie kolumn). Dostęp do partycjonowanej tabeli przy użyciu zapytań SQL i instrukcji DML nie wymaga modyfikacji.*

Normalizacja schematów relacyjnych

Poziomy normalizacji (N^{th} Normal Form):

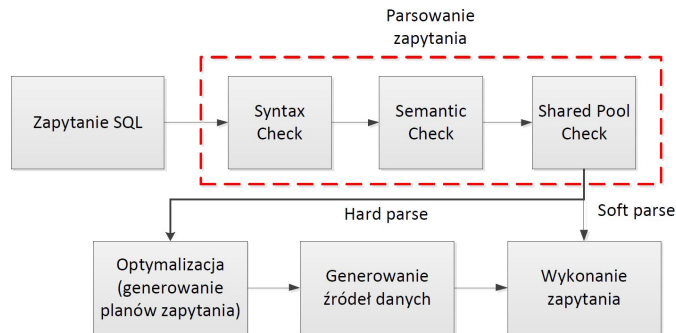
- Schematy nieznormalizowane
- Pierwsza postać normalna (1^{st} NF)
 - Normalizacja 1^{st} NF polega na podziale nieatomowych kolumn na atomowe
- Druga postać normalna (2^{nd} NF)
- Trzecia postać normalna (3^{rd} NF; Boyce-Codd NF)
- Czwarta postać normalna (4^{th} NF)
- Piąta postać normalna (5^{th} NF)
 - Normalizacja 2^{nd} NF ÷ 5^{th} NF zawsze polega na podziale tabel na mniejsze

Dobór typów danych

- Strony kodowe
 - ASCII – 1 bajt na znak
 - Unicode – 1 ÷ 4 bajtów na znak
- Typy znakowe
 - CHAR, VARCHAR (VARCHAR2), BLOB, CLOB
 - NCHAR, NVARCHAR2, NCLOB – Unicode
- Typy numeryczne
 - NUMBER – typ Oracle (do 21 bajtów)
 - BINARY_FLOAT (4 bajty), BINARY_DOUBLE (8 bajtów)

**Typy danych należy dobierać tak,
aby minimalizować przestrzeń
zajmowaną przez dane.**

Przetwarzanie zapytań SQL



- Zapytania po sprawdzeniu składni są interpretowane
- Zapytanie można wykonać na wiele sposobów opisanych za pomocą planów wykonania
- Optymalizator kosztowy wybiera plan wykonania o minimalnym koszcie (procesor, pamięć operacyjna, operacje dyskowe)

Do wersji 10 Oracle był używany również optymalizator regułowy

Optymalizator zapytań

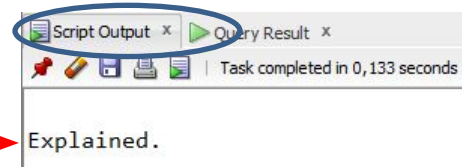
Tryby pracy określają rozmiar wynikowej puli wierszy zapytania, do której koszt dostępu należy zminimalizować:

- ALL_ROWS – (**domyślny**) wszystkie
- FIRST_ROWS – kilka pierwszych
- FIRST_ROWS_N – pierwszych N
 $N \in \{1, 10, 100, 1000\}$

```
alter session set optimizer_mode = first_rows;  
alter system set optimizer_mode = first_rows_10;
```

Plan zapytania (explain plan)

explain plan for
select * from dual;



SELECT PLAN_TABLE_OUTPUT FROM TABLE(DBMS_XPLAN.DISPLAY());

Script Output x Query Result x
All Rows Fetched: 8 in 0,054 seconds

PLAN_TABLE_OUTPUT

1 Plan hash value: 272002086

2

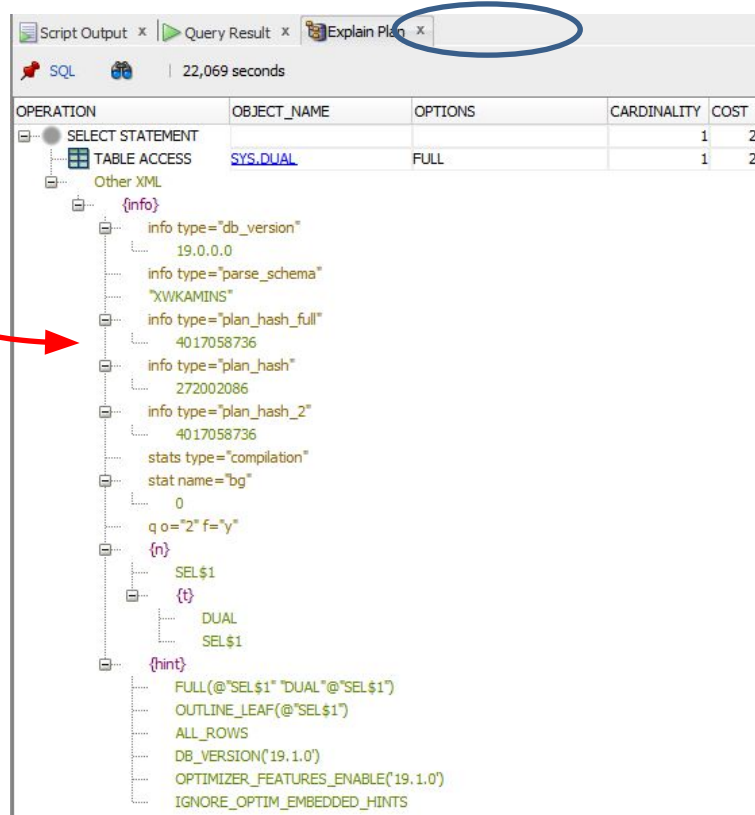
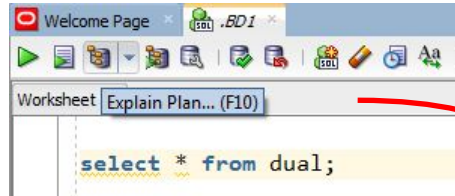
3

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	2	2 (0)	00:00:01
1	TABLE ACCESS FULL	DUAL	1	2	2 (0)	00:00:01

8

Plan zapytania (explain plan)

SQLDeveloper



The 'Explain Plan' window displays the execution plan for the query 'select * from dual;'. The plan is shown in a tree view with the following structure:

- SELECT STATEMENT
 - TABLE ACCESS (SYS_DUAL) FULL
 - Other XML
 - {info}
 - info type="db_version" 19.0.0.0
 - info type="parse_schema" "XWKAMINS"
 - info type="plan_hash_full" 4017058736
 - info type="plan_hash" 272002086
 - info type="plan_hash_2" 4017058736
 - stats type="compilation"
 - stat name="bg" 0
 - q o="2" f="y"
 - {n}
 - SEL\$1
 - {t}
 - DUAL
 - SEL\$1
 - {hint}
 - FULL(@"SEL\$1" "DUAL"@"SEL\$1")
 - OUTLINE_LEAF(@"SEL\$1")
 - ALL_ROWS
 - DB_VERSION('19.1.0')
 - OPTIMIZER_FEATURES_ENABLE('19.1.0')
 - IGNORE_OPTIM_EMBEDDED_HINTS

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	2
TABLE ACCESS	SYS_DUAL	FULL	1	2

Explain plan - ćwiczenia

Zapoznaj się z planami dla następujących zapytań:

1. `select * from employees;`
2. `select sysdate from dual;`
3. `select * from employees where employee_id > 130;`
4. `select * from employees where surname = 'Himuro';`
5. `select * from employees where salary between 1000 and 5000;`
6. `select department_id, count(*) from employees group by department_id;`
7. `select department_id, count(*) from employees where manager_id is not null group by department_id;`
8. `select department_id, count(*) from employees where manager_id is not null group by department_id having count (*) > 2;`

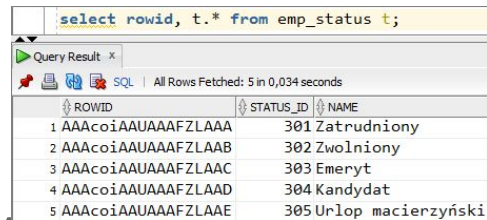
Plan zapytania: sposoby dostępu do danych

Ścieżka dostępu do danych	Tabele Heap-organized	Indeksy B-Tree	Indeksy bitmapowe	Klastry tabel
Full Table Scans	✓			
Table Access by Rowid	✓			
Sample Table Scans	✓			
Index Unique Scans		✓		
Index Range Scans		✓		
Index Full Scans		✓		
Index Fast Full Scans		✓		
Index Skip Scans		✓		
Index Join Scans		✓		
Bitmap Index Single Value			✓	
Bitmap Index Range Scans			✓	
Bitmap Merge			✓	
Cluster Scans				✓
Hash Scans				✓

Plan zapytania: sposoby dostępu do danych

Pokazują sposób dostępu do danych podczas realizacji zapytania.

- **Table Access Full** – przegląd danych **całej tabeli**.
- **Table Access by ROWID** – dostęp do wiersza tabeli za pomocą wskaźnika ROWID (zazwyczaj pobieranego z indeksu).
 - ROWID – wewnętrzna reprezentacja lokalizacji wiersza w bazie danych (plik danych + blok danych + pozycja w bloku)
 - **ROWID może się zmieniać!**
- **Sample Table Scan** – pobranie losowej próbki danych
`select * from employees sample (20);`



```
select rowid, t.* from emp_status t;
```

ROWID	STATUS_ID	NAME
1 AAACoiAAUAAAFZLAAA	301	Zatrudniony
2 AAACoiAAUAAAFZLAAB	302	Zwolniony
3 AAACoiAAUAAAFZLAAC	303	Emeryt
4 AAACoiAAUAAAFZLAAD	304	Kandydat
5 AAACoiAAUAAAFZLA AE	305	Urlop macierzyński

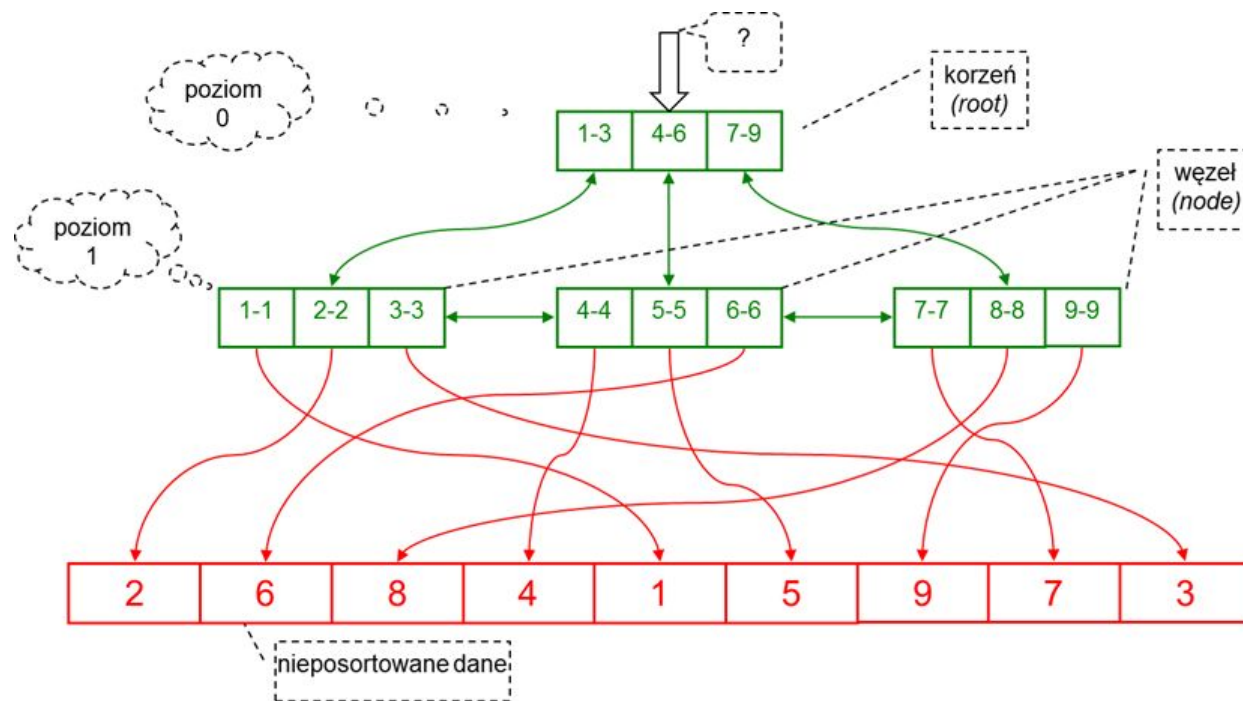
Indeksowanie

- Korzyści:
 - przyspieszenie dostępu do danych
 - pojedyncze rekordy
 - grupy rekordów
 - ułatwienie pewnych operacji
 - sortowanie
- Wady:
 - spowolnienie operacji manipulowania danymi (wstawienie, modyfikacja, usuwanie)
 - zużycie dodatkowych zasobów (pamięć dyskowa, pamięć operacyjna, procesor)
 - degradacja przy modyfikacji danych

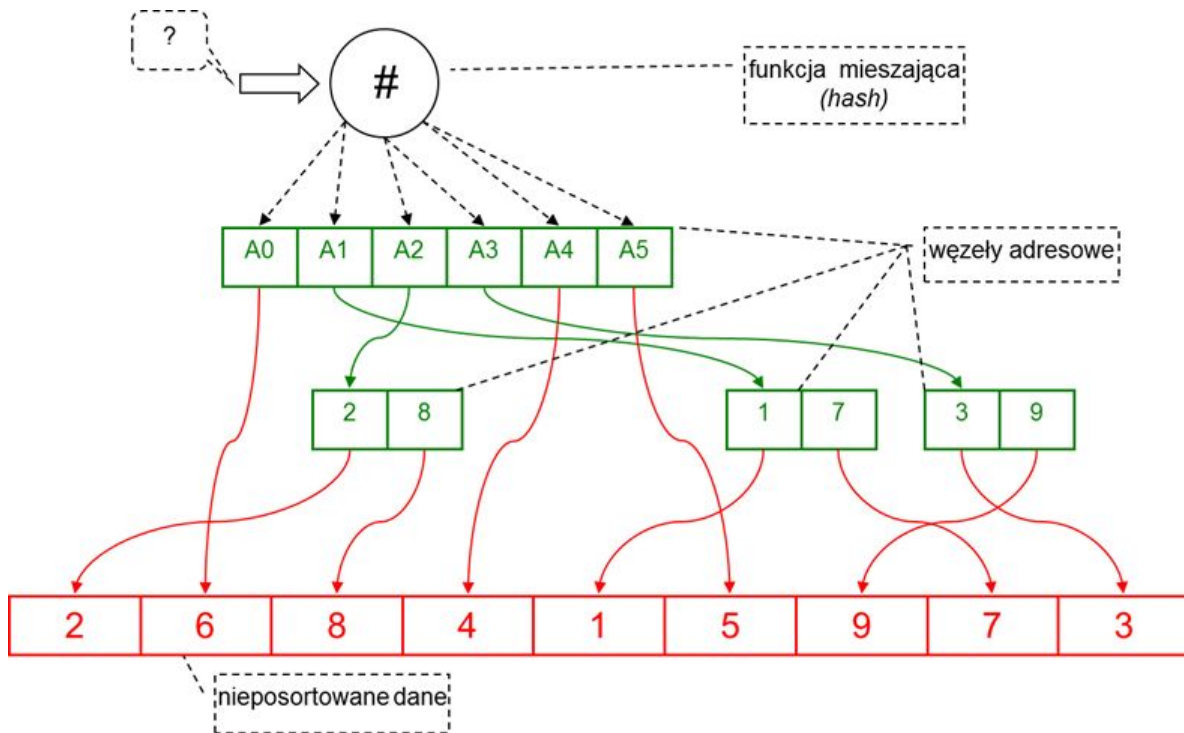
Indeksowanie

- Rodzaje indeksów:
 - sortujące – B-drzewa (*B-tree*)
 - (+) możliwość strojenia
 - (+) możliwość sortowania danych
 - (–) duże zużycie pamięci dyskowej
 - niesortujące – indeksy z funkcją mieszającą (*hash*), indeksy bitmapowe
 - (+) bardzo szybki dostęp do danych
 - (+) stosunkowo małe zużycie pamięci dyskowej
 - (–) brak możliwości sortowania

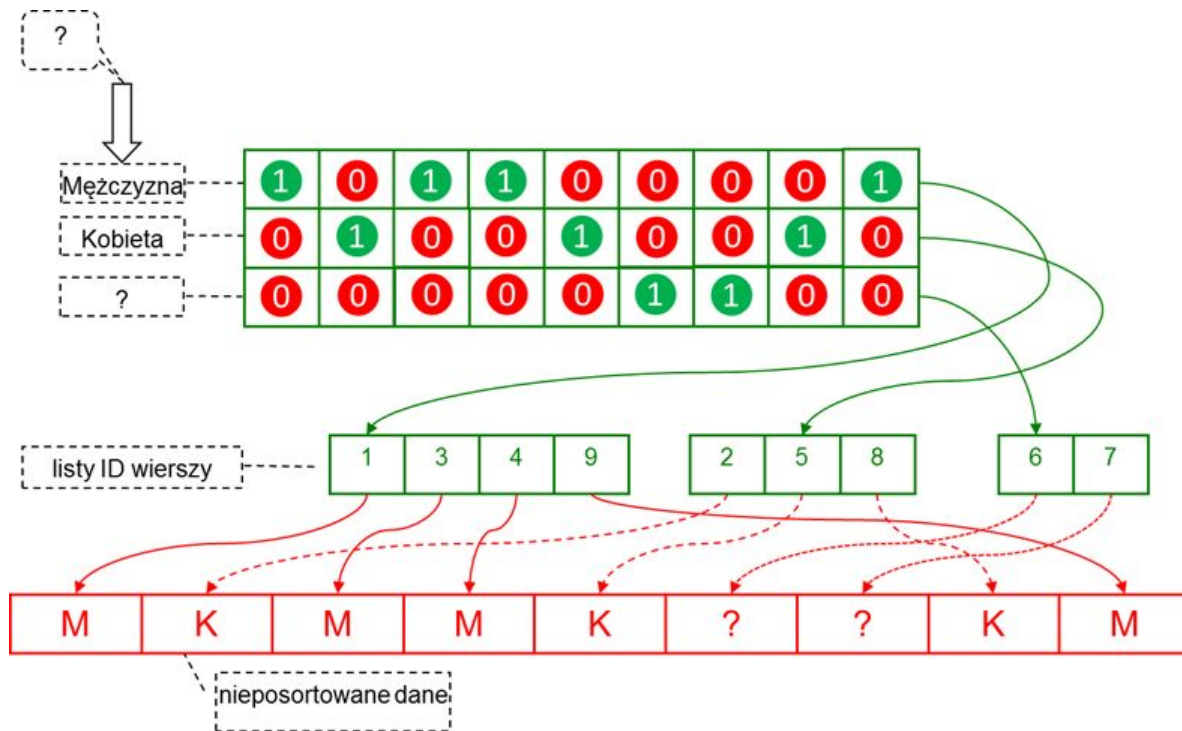
Idea indeksu B-drzewo



Idea indeksu haszującego



Idea indeksu bitmapowego



Indeksy w Oracle

- Rodzaje:
 - B-tree
 - Bitmap and bitmap join indexes
 - Function-based indexes
 - Application domain indexes

-- Sprawdzenie jakie są indeksy na tabeli

```
SELECT * FROM ALL_INDEXES where owner = 'HR' and table_name = 'EMPLOYEES';  
SELECT * FROM USER_INDEXES where table_name = 'EMPLOYEES';
```

Plan zapytania: sposoby dostępu do danych (2)

- **Index Unique Scans** – dostęp do pojedynczego wiersza tabeli z wykorzystaniem unikalnego indeksu przy warunkach selekcji na równość.
- **Index Range Scans** – dostęp do pojedynczego lub wielu wierszy tabeli z wykorzystaniem nieunikalnego indeksu przy warunkach selekcji na równość lub definiujących zakres.
- **Index Full Scans** – dostęp do wierszy w porządku wyznaczonym przez indeks w celu eliminacji oddzielnej operacji sortowania.
- **Index Fast Full Scans** – odczyt danych wynikowych zapytania bezpośrednio z indeksu a nie z tabeli.

Obsługa indeksów (B-tree)

- Tworzenie indeksu

```
CREATE [UNIQUE] INDEX nazwa_indeksu  
ON nazwa_tabeli (nazwa_kolumny {, nazwa_kolumny});
```

- Modyfikacja indeksu

```
ALTER INDEX nazwa_indeksu [ VISIBLE | INVISIBLE | UNUSABLE |  
REBUILD ];
```

- Usuwanie indeksu

```
DROP INDEX nazwa_indeksu;
```

Indeksy - ćwiczenia

1. Załóż indeks na kolumnie surname w tabeli employees.
2. Przeanalizuj plan zapytania z warunkiem równościowym na tej kolumnie. Przeanalizuj zapytanie z warunkiem równościowym na kluczu głównym tabeli employees. Porównaj uzyskane plany.
3. Zaindeksuj kolumnę tabeli zawierającą NULL-e i sprawdź plan wykonania przy zapytaniu z warunkiem „kolumna IS NULL” oraz „kolumna IS NOT NULL”.
4. Utwórz indeks B-tree na dwóch kolumnach (np. w tabeli positions na min_salary i max_salary) i sprawdź plan wykonania dla wszystkich kombinacji występowania kolumn indeksujących w klauzuli WHERE. Kiedy wykorzystany jest indeks?
5. Sprawdź plany wykonania zapytań z frazą ORDER BY dotyczącą kolumn zaindeksowanych null/not null.
6. Sprawdź plany wykonania zapytań z/bez fraz(ą/y) ORDER BY dotyczącą kolumn zaindeksowanych indeksami unikalnymi i nieunikalnymi, przy czym lista selekcji ma być ograniczona wyłącznie do tych kolumn.
7. Jakie czynniki mają wpływ na to czy indeks zostanie wykorzystany?

Obsługa indeksów funkcyjnych

- Tworzenie indeksu

```
CREATE [UNIQUE] INDEX nazwa_indeksu  
ON nazwa_tabeli (nazwa_funkcji(nazwa_kolumny));
```

- Modyfikacja indeksu

```
ALTER INDEX nazwa_indeksu  
ON nazwa_tabeli (nazwa_funkcji(nazwa_kolumny));  
  
ALTER INDEX nazwa_indeksu [ ENABLE | DISABLE ];
```



- DROP INDEX nazwa_indeksu;

Usunięcie indeksu

Indeksy funkcyjne - ćwiczenia

1. Wykonaj zapytanie na tabeli employees z warunkiem

surname = 'Himuro'

oraz z warunkiem

UPPER(surname) = 'HIMURO'.

Kiedy jest wykorzystany indeks?

2. Stwórz indeks funkcyjny na tabeli employees na kolumnie surname poddanej działaniu funkcji upper.

3. Wykonaj zapytanie z warunkiem upper(surname) = 'HIMURO'.

Czy jest wykorzystany indeks?

Plan wykonania: złączenia

- **Nested Loops**

- Zbiory danych dzieli się na zewnętrzne i wewnętrzne
- Dla każdego wiersza ze zbioru zewnętrznego wyszukuje się wiersze ze zbioru wewnętrznego (możliwe użycie indeksów w zbiorze wewnętrznym)

- **Sort Merge**

- Odmiana Nested Loops
- Zestawy danych są sortowane (operacje SORT JOIN)
- Dla każdego wiersza w pierwszym zestawie danych sprawdzany jest drugi zestaw danych pod kątem zgodnych wierszy i łączy je (operacja MERGE JOIN)
- Pozycja początkowa iteracji bazuje na dopasowaniu dokonanym w poprzedniej iteracji

- **Hash Join**

- Dla mniejszego zbioru danych budowana jest tablica mieszająca na kluczu złączenia przechowywana w pamięci
- Większy zbiór danych jest przeszukiwany ze względu na dopasowanie do wartości w tablicy mieszającej

Plan wykonania

złączenie (zaindeksowane tylko klucze główne)

The screenshot displays the SQL Developer interface with a query in the Query Builder and its execution plan in the Explain Plan tab.

Query:

```
select * from employees e inner join departments d on e.department_id = d.department_id;
```

Execution Plan:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			51	4
HASH JOIN			51	4
Access Predicates				
E.DEPARTMENT_ID=D.DEPARTMENT_ID				
TABLE ACCESS	DEPARTMENTS	FULL	19	2
TABLE ACCESS	EMPLOYEES	FULL	51	2
Filter Predicates				
E.DEPARTMENT_ID IS NOT NULL				
Other XML				

The execution plan shows a HASH JOIN operation. The join is based on the predicate E.DEPARTMENT_ID=D.DEPARTMENT_ID. The tables DEPARTMENTS and EMPLOYEES are accessed using FULL scans. The cardinality of the join result is 51, and the cost is 4. The filter predicate E.DEPARTMENT_ID IS NOT NULL is also shown.

Wskazówki (hints)

Wskazówki:

- umożliwiają wpływanie na wybór planu wykonania przez optymalizator
- Są przekazywane za pomocą specjalnie sformatowanych komentarzy w treści zapytania

```
SELECT /*+ hint_text */ ...
```

```
/*+ NO_INDEX(employees emp_manager) */
```

```
/*+ INDEX(emp_dep_pk prac_kod_s) */
```

```
/*+ use_nl (emp dep) */ albo use_merge, use_hash
```

Złączenia - ćwiczenia

1. Jak zmieni się plan wykonania, jeśli w poprzednim przykładzie w definicji złączenia zastosujemy frazę USING?
2. Dodaj indeks B-tree na kolumnie klucza obcego z poprzedniego przykładu i sprawdź plan wykonania.
Jakiego rodzaju musi być to indeks (unikalny, nieunikalny)?
Czy wydajność złączenia uległa poprawie?
3. Za pomocą wskazówek zmień typ złączenia tabel.
4. Zmień warunek złączenia na nierównościowy, zweryfikuj jak zmienił się plan wykonania.

⚡ Następne zajęcia - sprawdzian ⚡

Zakres:

- widoki, zmaterializowane widoki, sekwencje, synonimy
- aspekty wydajności bazy danych



Praca domowa

1. Wykonaj ćwiczenia z przebiegu laboratorium
2. Co to jest klastrowanie tabel?
3. Do schematu bazy danych dodaj klaster tabel EMPLOYEES_CL oraz DEPARTMENTS_CL o strukturze odpowiadającej tabelom EMPLOYEES oraz DEPARTMENTS.
4. Przekopiuj dane z tabel EMPLOYEES oraz DEPARTMENTS do odpowiednich tabel w klastrze.
5. Sprawdź plany wykonania zapytań do pojedynczych tabel klastra z warunkami do różnych kolumn.
6. Sprawdź plan wykonania operacji złączenia sklastrowanych tabel.
7. Stwórz indeks bitmapowy na kolumnie status_id w tabeli pracownicy. Wykonaj zapytania na tej tabeli i sprawdź, czy indeks jest wykorzystywany.