Zaawansowany SQL

Robert A. Kłopotek

r.klopotek@uksw.edu.pl

Wydział Matematyczno-Przyrodniczy. Szkoła Nauk Ścisłych, UKSW

MySQL GREATEST i LEAST

- Zarówno funkcja GEATEST jak i LEAST przyjmują N argumentów i odpowiednio zwracają wartości największe i najmniejsze
- Składnia:
```
GREATEST(value1, value2, ...);
LEAST(value1,value2,...);
```
- Argumenty mogą zawierać mieszane typy danych. Następujące reguły porównania są stosowane do obu funkcji:
  - Jeśli jakikolwiek argument ma wartość NULL, obie funkcje zwracają NULL bez żadnego porównania.
  - Jeśli funkcje są używane w dla liczb INT lub REAL lub wszystkie argumenty są wartością całkowitą lub REAL, są one porównywane odpowiednio jako INT i REAL.
  - Jeśli argumenty składają się zarówno z liczb, jak i łańcuchów znaków, funkcje porównują je jako liczby.
  - Jeśli przynajmniej argument jest ciągiem niebędącym znakiem binarnym, funkcje porównują argumenty jako łańcuchy inne niż binarne.
  - We wszystkich innych przypadkach funkcje porównują argumenty jako ciągi znaków.

GREATEST i LEAST - przykłady

- ```
  SELECT GREATEST(10, 20, 30),  -- 30
         LEAST(10, 20, 30);     -- 10
  ```
- ```
  SELECT GREATEST('2','100'); -- 2
  ```
- ```
  SELECT GREATEST(10, '20', 30), -- blob
  LEAST('10', 20, 30); -- blob
  ```
  ```

CAST

- Funkcja CAST MySQL umożliwia konwertowanie wartości dowolnego typu na wartość o określonym typie.
- Składnia
  `CAST(expression AS TYPE);`
- Funkcja CAST () konwertuje wartość dowolnego typu na wartość określoną.
- Typem docelowym może być dowolny z następujących typów: BINARY, CHAR, DATE, DATETIME, TIME, DECIMAL, SIGNED, UNSIGNED.
- Funkcja CAST () jest często używana do zwracania wartości określonego typu do porównania w klauzulach WHERE, JOIN i HAVING.

CAST - przykłady

```
SELECT 10+'10'; -- 20
SELECT '10'+'10'; -- 20
```

```
SELECT (1 + '1')/2; -- 1
```

```
SELECT (1 + CAST('1' AS UNSIGNED))/2; -- 1.0000
```

```
SELECT CONCAT('MySQL CAST ',CAST(1000 AS CHAR));
-- 'MySQL CAST 1000,
```

```
SELECT CAST(GREATEST(10, '20', 30) AS SIGNED), -- 30
CAST(LEAST('10', 20, 30) AS SIGNED); -- 10
```

INTERSECT w MS SQL

- Polecenie INTERSECT w języku T-SQL łączy w sobie wyniki dwóch instrukcji SQL i zwraca tylko dane, które są obecne w obu instrukcjach SQL.
- INTERSECT może być traktowany jako operator AND (wartość jest wybrana tylko wtedy, gdy występuje w obu instrukcjach), podczas gdy UNION i UNION ALL mogą być traktowane jako operatora OR (wartość jest wybrana, jeśli występuje w pierwszej lub drugiej instrukcji).
- Składnia

```
[SQL Statement 1]
INTERSECT
[SQL Statement 2];
```

- Kolumny w zapytaniu SELECT, podobnie jak w UNION, muszą być tego samego typu oraz ilość kolumn musi się zgadzać

INTERSECT w MySQL

- W MySQL nie ma instrukcji INSERT.
- Aby otrzymać podobna funkcjonalność można użyć składni:

```
SELECT DISTINCT value
FROM table_a
INNER JOIN table_b
USING (value);
```

- Przykład:

```
SELECT DISTINCT imie
FROM osoby o2
INNER JOIN osoby o1
USING (imie);
```

MINUS w MS SQL

- Instrukcja MINUS działa na dwóch instrukcjach SQL.
- Pobiera wszystkie wyniki z pierwszej instrukcji SQL, a następnie odejmuje te, które są obecne w drugiej instrukcji SQL, aby uzyskać końcowy zestaw wyników.
- Jeśli druga instrukcja SQL zawiera wyniki nie występujące w pierwszej instrukcji SQL, takie wyniki są ignorowane.
- Składnia:

```
[SQL Statement 1]
INTERSECT
[SQL Statement 2];
```

CASE

- CASE jest używany do dostarczania od SQL typu logiki if-then-else.
- Istnieją dwa rodzaje:
  - pierwszy to prosta wyrażenie CASE, w której porównujemy wyrażenie do wartości statycznych.
  - drugi to wyszukiwane wyrażenie CASE, w którym porównujemy wyrażenie z jednym lub większą liczbą warunków logicznych.

CASE – proste wyrażenia

- ```
  SELECT CASE ("column_name")
    WHEN "value1" THEN "result1"
    WHEN "value2" THEN "result2"
    ...
    [ELSE "resultN"]
    END
  FROM "table_name";
  ```
- **Przykład:**

  ```
  SELECT Store_Name, CASE Store_Name
    WHEN 'Los Angeles' THEN Sales * 2
    WHEN 'San Diego' THEN Sales * 1.5
    ELSE Sales
    END
  "New Sales",
  Txn_Date
  FROM Store_Information;
  ```
  ```

CASE – wyrażenia wyszukiwane

- ```
  SELECT CASE
    WHEN "condition1" THEN "result1"
    WHEN "condition2" THEN "result2"
    ...
    [ELSE "resultN"]
    END
  FROM "table_name";
  ```
- **Przykład**

  ```
  SELECT Store_Name, Txn_Date, CASE
    WHEN Sales >= 1000 THEN 'Good Day'
    WHEN Sales >= 500 THEN 'OK Day'
    ELSE 'Bad Day'
    END
  "Sales Status"
  FROM Store_Information;
  ```
  ```

DECODE

- DECODE jest funkcją Oracle i służy do dostarczania do SQL logiki typu if-then-else. NIE jest dostępna w MySQL lub SQL Server.
- Składnia:

```
SELECT DECODE ( "column_name", "search_value_1",
"result_1",
["search_value_n", "result_n"],
{"default_result"} );
```

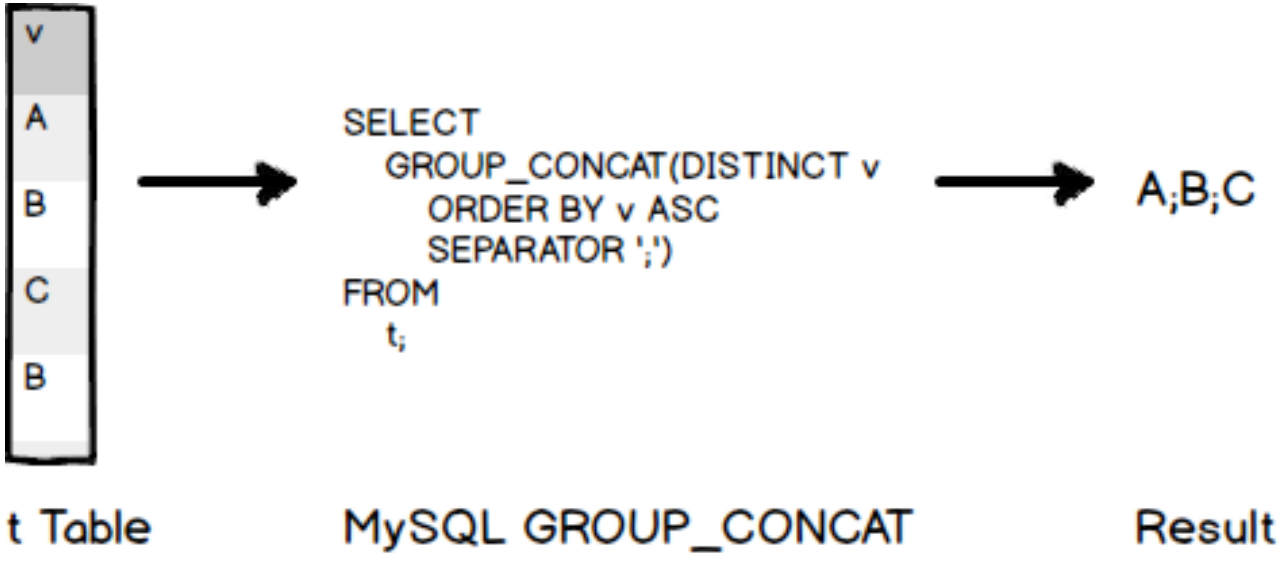
- Przykład:

```
SELECT DECODE (Store_Name,
   'Los Angeles', 'LA',
   'San Francisco', 'SF',
   'San Diego', 'SD',
   'Others') Area, Sales, Txn_Date
FROM Store_Information;
```

MySQL GROUP_CONCAT

- Funkcja MySQL GROUP_CONCAT łączy ciągi znaków z grupy (kolumny) w jeden ciąg znaków przy wykorzystaniu różnych opcji
- Składnia:

```
GROUP_CONCAT(DISTINCT expression
  ORDER BY expression
  SEPARATOR sep);
```



GROUP_CONCAT - szczegóły

- SEPARATOR określa wartość literalną wstawioną między wartościami w grupie. Jeśli separator nie będzie określony, funkcja GROUP_CONCAT używa przecinków (,) jako domyślnego separatora.
- Funkcja GROUP_CONCAT ignoruje wartości NULL. Zwraca wartość NULL jeśli nie znaleziono pasującego wiersza lub wszystkie argumenty są wartościami NULL.
- Funkcja GROUP_CONCAT zwraca ciąg napisów binarny lub nie-binarny, który zależy od argumentów. Domyślnie maksymalna długość ciągu zwrotnego wynosi 1024. Jeśli potrzebujemy więcej, możemy rozszerzyć maksymalną długość, ustawiając zmienną systemową group_concat_max_len na poziomie SESSION lub GLOBAL.

GROUP_CONCAT - przykład

```
SELECT
  imie, nazwisko,
  GROUP_CONCAT(
    DISTINCT id_firmy
    ORDER BY id_firmy
    SEPARATOR ';') as etaty_w_firmach
FROM
  osoby o
  LEFT JOIN etaty e ON o.id_osoby=e.id_osoby
GROUP BY
  o.id_osoby;
```

imie	nazwisko	etaty_w_firmach
Maciej	Stodolski	FLP;HP;UKSW
Jacek	Korytkowski	UKSW
Mis	Nieznany	NULL
Krol	Neptun	FLP
Juz	Niepracujacy	HP

MySQL IFNULL

- Funkcja IFNULL MySQL jest jedną z funkcji kontroli przepływu MySQL, która akceptuje dwa argumenty i zwraca pierwszy argument, jeśli nie jest to wartość NULL. W przeciwnym razie funkcja IFNULL zwróci drugi argument.
- Te dwa argumenty mogą być wartościami literalnymi lub wyrażeniami.
- Składnia:
  `IFNULL(expression_1,expression_2);`
- Funkcja IFNULL zwraca wyrażenie_1, jeśli wyrażenie_1 nie jest NULL; w przeciwnym razie zwraca wyrażenie_2.
- Funkcja IFNULL zwraca ciąg znaków lub liczbę w oparciu o kontekst, w którym jest używany.

IFNULL - przykłady

```
SELECT IFNULL(1,0);
-- zwraca 1
```

```
SELECT IFNULL('',1);
-- zwraca ''
```

```
SELECT IFNULL(NULL,'IFNULL');
-- zwraca 'IFNULL'
```

```
SELECT
  e.id_etatu,
  IFNULL(e.do,'aktualnie niezakończony')
     AS koniec_etatu
FROM
  etaty e;
```

id_etatu	koniec_etatu
1	1998-01-01 00:00:00
2	2000-01-01 00:00:00
3	aktualnie niezakończony
4	aktualnie niezakończony
5	aktualnie niezakończony
6	aktualnie niezakończony
7	2004-10-22 00:00:00
8	2002-10-21 00:00:00
9	2004-10-22 00:00:00
10	aktualnie niezakończony

MySQL NULLIF

- Funkcja NULLIF jest jedną z funkcji kontroli przepływu, która akceptuje 2 argumenty.
- Funkcja NULLIF zwraca NULL, jeśli pierwszy argument jest równy drugiemu argumentowi, w przeciwnym razie zwraca pierwszy argument.
- Składnia:
  `NULLIF(expression_1,expression_2);`
- Funkcja NULLIF zwraca NULL, jeśli wyrażenie 1 = wyrażenie 2 jest prawdziwe, w przeciwnym wypadku zwraca wyrażenie 1
- Zauważmy, że funkcja NULLIF jest podobna do następującej wyrażenia, która używa wyrażenia CASE:

```
CASE WHEN expression_1 = expression_2
   THEN NULL
ELSE
   expression_1
END;
```

NULLIF - przykłady

- ```
  SELECT NULLIF(1,1);
  -- zwraca NULL
  ```
- ```
  SELECT NULLIF(1,2);
  -- zwraca 1
  ```
- ```
  SELECT NULLIF('NULLIF','NULLIF');
  -- zwraca NULL
  ```
  ```

MySQL IF

- Funkcja MySQL IF jest jedną z funkcji sterowania przepływem MySQL, która zwraca wartość w oparciu o warunek.
- Funkcja IF jest czasami nazywana IF ELSE lub IF THEN ELSE.
- Składnia:
  `IF(expr,if_true_expr,if_false_expr)`
- Jeśli wyrażenie expr ma wartość TRUE, tj. expr nie jest NULL i expr nie wynosi 0, to funkcja IF zwraca wartość if_true_expr, w przeciwnym przypadku zwraca if_false_expr.
- Funkcja IF zwraca liczby lub ciąg znaków, w zależności od tego, jak jest użyta (dla jakich argumnetów).

IF - przykłady

- ```
  SELECT IF(1 = 2,'true','false');
  -- false
  ```
- ```
  SELECT IF(1 = 1,'true','false');
  -- true
  ```
- ```
  SELECT
    SUM(IF(stanowisko IN ('Asystent', 'Adiunkt', 'Profesor'), 1, 0))
      AS etaty_szkolnictwo,
    SUM(IF(stanowisko IN ('Kierownik', 'Dyrektor', 'Prezes'), 1, 0))
      AS etaty_biurowe
  FROM
    etaty;
  ```
  ```

Ranking

- Niektóre systemy bazodanowe (jak MS SQL Server czy Oracle) mają specjalna funkcję RANK
- Jak obliczyć ranking w SZBD, które tego nie mają?
- Ogólna idea jest następująca:
  - najpierw robimy samosprzeżenie (self-join)
  - Następnie sotujemy wynik w odpowiednim porządku
  - Na koniec zliczamy ilość rekordów poprzedzających (i aktualny) dla każdego rekordu wynikowego

Ranking - przykład

Name	Sales
John	10
Jennifer	15
Stella	20

Self-join

Name1	Sales1	Name2	Sales2
John	10	John	10
John	10	Jennifer	15
John	10	Stella	20
Jennifer	15	John	10
Jennifer	15	Jennifer	15
Jennifer	15	Stella	20
Stella	20	John	10
Stella	20	Jennifer	15
Stella	20	Stella	20

porządkowanie

Name1	Sales1	Name2	Sales2
Stella	20	Stella	20
Jennifer	15	Stella	20
Jennifer	15	Jennifer	15
John	10	Stella	20
John	10	Jennifer	15
John	10	John	10

zliczanie

Name	Sales	RANK
Stella	20	1
Jennifer	15	2
John	10	3

Ranking – zapytanie SQL

```
SELECT a1.Name, a1.Sales,
COUNT (a2.Sales) Sales_Rank
FROM Total_Sales a1, Total_Sales a2
WHERE a1.Sales < a2.Sales
OR (a1.Sales=a2.Sales AND a1.Name = a2.Name)
GROUP BY a1.Name, a1.Sales
ORDER BY a1.Sales DESC, a1.Name DESC;
```

Mediana

- Aby uzyskać medianę, musimy wykonać następujące czynności:
  - Sortuj wiersze w kolejności i znajdź rangę każdego wiersza.
  - Ustal, co to jest "środkowy" ranking. Na przykład, jeśli jest 9 wierszy, średnim rangą będzie 5.
  - Zdobądź wartość dla rzędu średniej.

Mediana – zapytanie SQL

```
SELECT Sales Median FROM
(
 SELECT a1.Name, a1.Sales, COUNT(a1.Sales) Rank
 FROM Total_Sales a1, Total_Sales a2
 WHERE a1.Sales < a2.Sales
 OR (a1.Sales=a2.Sales AND a1.Name <= a2.Name)
 GROUP by a1.Name, a1.Sales
 ORDER by a1.Sales desc
) a3
WHERE Rank = (
             SELECT (COUNT(*)+1) DIV 2
             FROM Total_Sales
             );
```

Name	Sales	RANK
Stella	20	1
Jennifer	15	2
John	10	3

Sumy częściowe

- Wyświetlanie sum częściowych jest częstym żądaniem dla tabel w firmach i nie ma prostego sposobu na to w SQL.
- Pomysł jak wyświetlić sumy częściowe za pomocą SQL jest podobny do rankingu: najpierw zrób samosprzężenie, a następnie uporządkuj wyniki
- Znalezienie bieżącej sumy wymaga zsumowania wartości rekordów, które są wymienione przed (i zawierające) odpowiedni rekord.
- Przykład:

```
SELECT a1.Name, a1.Sales, SUM(a2.Sales) Running_Total
FROM Total_Sales a1, Total_Sales a2
WHERE a1.Sales <= a2.sales
OR (a1.Sales=a2.Sales and a1.Name = a2.Name)
GROUP BY a1.Name, a1.Sales
ORDER BY a1.Sales DESC, a1.Name DESC;
```

Name	Sales	Running _Total
Stella	20	20
Jennifer	15	35
John	10	45

Procent całości

- Aby obliczyć procent całości możemy wykorzystać poprzednie wyniki z rankingu i sum częściowych plus małe podzapytanie
- Jak to zrobić?

```
SELECT a1.Name, a1.Sales,
a1.Sales/(SELECT SUM(Sales) FROM Total_Sales) Pct_To_Total
FROM Total_Sales a1, Total_Sales a2
WHERE a1.Sales <= a2.sales
OR (a1.Sales=a2.Sales and a1.Name = a2.Name)
GROUP BY a1.Name, a1.Sales
ORDER BY a1.Sales DESC, a1.Name DESC;
```

Name	Sales	Pct_To_Total
Stella	20	0.4444
Jennifer	15	0.3333
John	10	0.2222

Skumulowany procent do całości

- Jest to przerobienie kodu procent do całości.
- Jak to zrobić?

```
SELECT
  a1.Name, a1.Sales,
  SUM(a2.Sales)/(SELECT SUM(Sales) FROM Total_Sales) Pct_To_Total
FROM
  Total_Sales a1, Total_Sales a2
WHERE
  a1.Sales <= a2.sales
  OR (a1.Sales=a2.Sales and a1.Name = a2.Name)
GROUP BY a1.Name, a1.Sales
ORDER BY a1.Sales DESC, a1.Name DESC;
```

Name	Sales	Pct_To_Total
Stella	20	0.4444
Jennifer	15	0.7778
John	10	1.0000

Odchylenie standardowe MySQL

- Odchylenie standardowe jest miarą rozłożenia wartości w zbiorze danych. Odchylenie standardowe pokazuje, jak duża różnica istnieje od przeciętnej (średniej).
- Niski odchylenie standardowe wskazuje, że wartości w zestawie danych są bliskie średniej. I wysoki odchylenie standardowe wskazuje, że wartości zestawu danych są rozłożone na szeroki zakres wartości.
- Jeśli w obliczeniach uwzględniono wszystkie wartości w zestawie danych, to odchylenie standardowe nazywa się odchyleniem standardowym populacji. Jeśli jednak do obliczenia zostanie wzięty podzbiór wartości lub próbki, to odchylenie standardowe nazywa się próbką odchylenia standardowego.

- Aby obliczyć odchylenie standardowe populacji, należy użyć jednej z następujących funkcji:
  - STD (wyrażenie) - zwraca wartość odchylenia standardowego populacji wyrażenia. Funkcja STD zwraca NULL, jeśli nie ma pasującego wiersza.
  - STDDEV (wyrażenie) - odpowiada funkcji STD. Jest on zgodny z bazą danych Oracle.
  - STDEV_POP (wyrażenie) - jest odpowiednikiem funkcji STD.
- MySQL zawiera również pewne funkcje wariancji populacji i obliczania wariancji próbki:
  - VAR_POP (wyrażenie) - oblicza standardową wariancję populacji z wyrażenia.
  - VARIANCE (wyrażenie) - jest odpowiednikiem funkcji VAR_POP
  - VAR_SAMP (wyrażenie) - oblicza wariancję próbki.

Pytania?