Bazy Danych

1.Algebra Relacji & SQL

Opracował: Maciej Penar

Spis treści

[1. Zanim zaczniemy 3](#_Toc508211720)

[Drzewo operatorów algebry relacji 3](#_Toc508211721)

[Ściąga sql 5](#_Toc508211722)

[2. (5 pkt) Algebra relacji – część bardziej ćwiczeniowa 8](#_Toc508211723)

[3. (7 pkt) SQL 8](#_Toc508211724)

[4. Kartkówka 9](#_Toc508211725)

# 1. Zanim zaczniemy

Zrelaksować się i przyswoić sobie teorię dot. Algebry relacji.

Materiały:

* Google: <https://www.google.pl/search?q=algebra+relacji&oq=algebra+relacji>
* Podstawowy kurs systemów baz danych, rozdział 2 oraz 5.2, J. Ullman, J. Widom

Oprogramowanie:

* SQLite: <https://www.sqlite.org/index.html>
* SQLite (link 2):

<https://github.com/mpenarprz/BazyDanychI4/tree/master/Laboratorium/tools>

* GUI do SQLite: <http://sqlitebrowser.org/>
* MS Access ?

## Drzewo operatorów algebry relacji

Jak komuś nie chce się otwierać książki „Podstawowy kurs systemów baz danych” to zamieszczam krótkie info o co chodzi z zapytaniami w „algebrze relacji” w **formie drzewa operatorów**. Trzeba znać operatory żeby zrozumieć o co tu chodzi.

Załóżmy relację np. ***Ziemniaki****(Dojrzały, Rozmiar, Waga)*

Niech atrybut **Dojrzały** opisuje czy ziemniak należący do relacji jest dojrzały lub nie (true/false). Z kolei atrybut **Rozmiar** niech ma zdefiniowaną dziedzinę {„Mały”, „Średni”, „Duży”} i opisuje jakościowo naszego ziemniaka. Atrybut **Waga** opisuje ilościowo ziemniaka. Prawidłowe wartości są większe od 0 (Waga >=0 ) – przyjmijmy że to waga gramach.

Załóżmy że instancja relacji Ziemniaki to np.:

|  |  |  |
| --- | --- | --- |
| **Ziemniaki** | | |
| **Dojrzały** | **Rozmiar** | **Waga** |
| True | Duży | 180 |
| True | Średni | 120 |
| True | Średni | 160 |
| False | Mały | 50 |

Zastanówmy się nad znaczeniem operatorów algebry relacji – otóż wyznaczają one pewien **podzbiór** relacji nad którą operują.

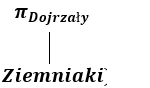
I tak wyrażenie wyznacza podzbiór relacji Ziemniaki zawierający jedynie atrybut *Dojrzały.*

Instancja relacji: to:

|  |
| --- |
|  |
| **Dojrzały** |
| True |
| True |
| True |
| False |

Zapis w formie: nazywamy liniowym

Nas interesuje zapis w formie drzewa:

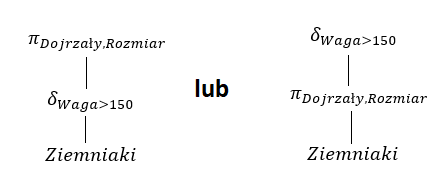
****

I czytamy go od góry do dołu, podążając lewą (na ogół) ścieżką. Czytamy: „Wykonujemy projekcję na atrybucie **Dojrzały** z relacji **Ziemniaki**”

Weźmy bardziej skomplikowane zapytanie np.

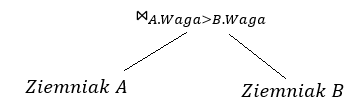
które wyznacza podzbiór relacji Ziemniaki zawierający jedynie atrybut *Dojrzały* oraz *Rozmiar,* w którym ziemniaki ważą 150g.

To zapis w formie drzewa przyjąłby postać:



Niektóre operatory są dwuargumentowe np. co powoduje rozgałęzianie się drzewa. Weźmy ultra trudne zapytanie np. które wybiera wszystkie pary ziemniaków których waga pierwszego jest większa od wagi drugiego.

Drzewo wygląda tak:



Ciekawostka: relacja wynikowa:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ziemniaki** | | | | | |
| **A.Dojrzały** | **A.Rozmiar** | **A.Waga** | **B.Dojrzały** | **B.Rozmiar** | **B.Waga** |
| True | Duży | 180 | True | Średni | 160 |
| True | Duży | 180 | True | Średni | 120 |
| True | Duży | 180 | False | Mały | 50 |
| True | Średni | 160 | True | Średni | 120 |
| True | Średni | 160 | False | Mały | 50 |
| True | Średni | 120 | False | Mały | 50 |

## Ściąga sql

Ściąga DQL w SQL – w miarę uniwersalna. Wytłuszczoną czcionką zaznaczono słowa kluczowe.

|  |  |
| --- | --- |
| Przykład | Co oznacza |
| **SELECT**  \*  **FROM**  MY\_TABLE | Pobiera wszystko z tabeli MY\_TABLE |
| **SELECT**  \*  **FROM**  MY\_TABLE  **ORDER BY**  ATT | Pobiera wszystko z tabeli MY\_TABLE, sortuje po atrybucie ATT rosnąco |
| **ORDER BY**  ATT **ASC**,  ATT2 **DESC** | Sortowanie po kilku atrybutach. Specyfikacja sortowania rosnąco ASC, malejąco DESC. |
| **SELECT TOP** 10  \*  **FROM**  MY\_TABLE | Wybranie pierwszych 10 rekordów. Wynik niedeterministyczny. To chyba że użyte z ORDER BY. |
| **SELECT**  \*  **FROM**  MY\_TABLE  **LIMIT** 10 | Wybranie pierwszych 10 rekordów. Wynik niedeterministyczny. To chyba że użyte z ORDER BY. |
| **SELECT DISTINCT**  \*  **FROM**  MY\_TABLE | Pobiera wszystkie unikatowe rekordy z tabeli MY\_TABLE |
| **SELECT**  MY\_ATTRIBUTE AS A,  MY\_ATTRIBUTE2  **FROM**  MY\_TABLE | Pobiera atrybuty MY\_ATTRIBUTE, który zostaje przemianowany na A, oraz atrybut MY\_ATTRIBUTE2 z tabeli MY\_TABLE |
| **SELECT**  **\***  **FROM**  MY\_TABLE AS TTT  **INNER JOIN** YOUR\_TABLE AS KKK **ON** TTT.ATT = KKK.ATT | Pobiera wszystko ze złączenia pomiędzy tabelą MY\_TABLE oraz YOUR\_TABLE. Obu tabelom nadano aliasy (odpowiednio TTT/KKK). Złączenie jest po warunku równościowym na atrybucie ATT |
| **INNER JOIN**  **LEFT OUTER JOIN**  **RIGHT OUTER JOIN**  **FULL OUTER JOIN**  **CROSS JOIN** | Rodzaje złączeń w SQL |
| **SELECT**  **\***  **FROM**  MY\_TABLE,  MY\_TABLE2,  MY\_TABLE3 | Iloczyn kartezjański (CROSS JOIN) table MY\_TABLE, MY\_TABLE2, MY\_TABLE3 |
| **SELECT**  **\***  **FROM**  **([SQL]) ALIAS** | Opakowanie zapytania. W klauzuli FROM można użyć zapytania. |
| **SELECT**  **\***  **FROM**  MY\_TABLE  **WHERE**  A > 0 | Pobiera wszystkie atrybuty z odfiltrowanej tabeli MY\_TABLE. Filtrowanie zachodzi na warunku A > 0. |
| **WHERE**  **[warunek]**  **AND [warunek]** | Łączenie warunków w klauzuli where – logiczne AND |
| **WHERE**  **[warunek]**  **OR [warunek]** | Łączenie warunków w klauzuli where – logiczne OR |
| **NOT [warunek]** | Negacja warunku |
| **WHERE**  **ATT IN (1,2,3,10)** | Sprawdzenie czy atrybut ATT posiada wartość ze zbioru {1,2,3,10} |
| **WHERE**  **ATT IN ([SQL])** | Sprawdzenie czy atrybut ATT posiada wartość ze zbioru – dynamicznie wyliczony zbiór |
| **WHERE**  **EXISTS ([SQL])** | Sprawdzenie niepustości dynamicznie wyliczonego zbioru |
| **WHERE**  MY\_TEXT\_ATTRIBUTE **LIKE [wzorzec]** | Sprawdzenie czy wartość atrybutu MY\_TEXT\_ATTRIBUTE pasuje do wzorca |
| ? (czasem \_) – dowolny znak (regexp: ‘.’)  % - dowolny ciąg znaków (regexp: ‘.’\*) | Specjalny znaki we wzorcach |
| **SELECT**  ATT**,**  **COUNT(\*)**  **FROM**  MY\_TABLE  **GROUP BY**  ATT | Utworzenie grup po wartościach atrybutu ATT oraz wyliczenie agregacji typu COUNT. |
| **SELECT**  ATT**,**  **COUNT(\*)**  **FROM**  MY\_TABLE  **WHERE**  A > 0  **GROUP BY**  ATT | Utworzenie grup po wartościach atrybutu ATT oraz wyliczenie agregacji typu COUNT. Do agregacji wliczane są **tylko** rekordy spełniające warunek A>0 |
| **COUNT**  **SUM**  **MIN**  **MAX**  **AVG** | Rodzaje funkcji agregujących w SQL – podstawowe |
| **COUNT(\*)** | Wyjątkowa agregacja – ile jest wartości |
| **AVG(WIEK)** | Średnia wartość atrybutu WIEK |
| **COUNT(DISTINCT WIEK)** | Wyjątkowa agregacja – ile różnych wartości znajduje się w grupie |
| **SELECT**  ATT**,**  **COUNT(\*)**  **FROM**  MY\_TABLE  **GROUP BY**  ATT  **HAVING**  **AVG(**TTT**)** > 10 | Utworzenie grup po wartościach atrybutu ATT oraz wyliczenie agregacji typu COUNT. Odfiltrowanie tych **grup** dla których agregacja AVG(TTT) osiąga wartość większą niż 10. |
| **[SQL]**  **UNION**  **[SQL]** | Suma wyników dwóch zapytań SQL.  Jako zbiór. |
| **[SQL]**  **UNION ALL**  **[SQL]** | Suma wyników dwóch zapytań SQL.  Jako multizbiór. |
| **UNION**  **UNION ALL**  **MINUS (EXCEPT)**  **MINUS (EXCEPT) ALL**  **INTERSECT** | Możliwe operacje na zbiorach w SQL. |

# 2. (5 pkt) Algebra relacji – część bardziej ćwiczeniowa

1. (0.5 pkt) Co to za operatory:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

1. (0.5 pkt) Z czego składa się schemat relacji
2. Dana jest relacja ***Osoba****(Imię, Nazwisko, Wiek, PESEL, Kolor Oczu, Włosy, Płeć)* oraz ***Zwierzę****(PESEL Właściciela, Gatunek, Nazwa, Wiek)*, napisać zapytania algebry relacji w formie drzewa operatorów:
   1. (1 pkt) Wybrać imię i nazwisko osób których wiek jest większy niż 30 lat i kolor oczu jest niebieski
   2. (1 pkt) Wybrać imię i nazwisko kobiet które posiadają zwierzę z gatunku Kot
   3. (2 pkt) Ile jest zwierząt w relacji, w podziale na płeć właściciela - którzy posiadają długie włosy - oraz gatunek zwierzęcia

Chodzi o relację wynikową:

|  |  |  |
| --- | --- | --- |
| Płeć | Gatunek | Liczba osobników |

# 3. (7 pkt) SQL

Otworzyć w SQLite bazę danych chinook.db (dostępne na repo) i napisać zapytania w formie wyrażeń SQL.

Zwrócić uwagę na **FORMATOWANIE ZAPYTAŃ.**

1. (0 pkt) Wyświetlić zawartość tabeli Customers (tzw. dump tabeli)
2. (0.5 pkt) Wyświetlić pierwsze alfabetycznie tytuły pierwszych 5 rekordów z tabeli albums
3. (0.5 pkt) Znaleźć kompozytora utworu (‘tracks’) o nazwie ‘No Futuro’
4. (0.5 pkt) Ile jest albumów?
5. (0.5 pkt) Znaleźć nazwy utworów oraz czasy trwania (w minutach) utworów które zajmują więcej niż 900000000 bajtów
6. (0.5 pkt) Wyświetlić albumy artysty ‘Van Halen’
7. (0.5 pkt)Wyświetlić pierwsze alfabetycznie tytuły pierwszych 5 rekordów z tabeli albums kończący się ‘(Remastered)’
8. (0.5 pkt) Wyświetlić alfabetycznie nazwy albumów które posiadają utwory z gatunku ‘Rock’ oraz ‘Metal
9. (0.5 pkt) Ile jest utworów bez kompozytora?
10. (0.5 pkt) Ile jest kompozytorów (nie artystów)?
11. (0.5 pkt) Policzyć zestawienie ile utworów ma album. Na zestawieniu są wszystkie albumy?
12. (0.5 pkt) Policzyć ile jest utworów których autorem jest autor albumu do którego należą te utwory
13. (0.5 pkt) Wyświetlić nazwę albumu oraz tytuł najdłuższego utworu tego albumu
14. (0.5 pkt) Wyświetlić 10 rekordów. Po 5 najdłuższych płyt w gatunkach Pop oraz Electronica/Dance
15. (0.5 pkt) Wyświetlić wszystkie pary utworów z albumu ‘Chemical Wedding’ dla których pierwszy utwór z pary jest krótszy od drugiego utworu z pary

# 4. Kartkówka

1. Definicja Bazy Danych
2. Co to jest Relacyjna Baza Danych
3. Co to jest Relacja
4. Co to jest Związek (związek ≠ relacja)
5. Co to jest transakcja?
6. **Rozwinięcie skrótu ACID**
7. Proste zapytania w formie:
   1. Wyrażeń SQL SELECT
   2. Algebry relacji
8. Co robi dany operator algebry relacji?