Katedra
Informatyki i Automatyki
Politechniki Rzeszowskiej

Bazy Danych

1. SQL

Opracował: Maciej Penar

Spis treści

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/

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.:

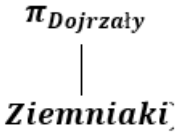
Ziemniaki		
Dojrzały	**Rozmiar**	**Waga**
True	Duży	180
True	Średni	120
True	Średni	160
False	Mały	50

Zastanówmy się nad znaczeniem operatorów algebry relacji – otóż wyznaczają one pewien podzbiór relacji nad którą operują. I tak wyrażenie $\pi(Ziemniaki)_{Dojrzały}$ wyznacza podzbiór relacji Ziemniaki zawierający jedynie atrybut Dojrzały. Instancja (wystąpienie) relacji: $\pi(Ziemniaki)_{Dojrzały}$ to:

$\boldsymbol{\pi(Ziemniaki)_{Dojrzały}}$
Dojrzały
True
True
True
False

Zapis w formie: $\pi(Ziemniaki)_{Dojrzały}$ nazywamy liniowym

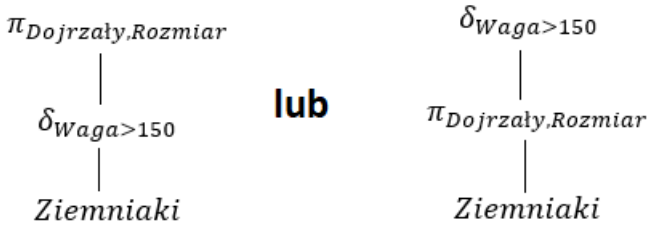
Nas będzie interesował zapis w formie drzewa:



Weźmy bardziej skomplikowane zapytanie np.

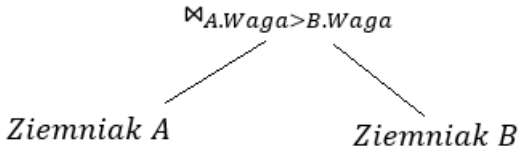
$\pi(\delta(Ziemniaki)_{Waga>150})_{Dojrzały,Rozmiar}$ które wyznacza podzbiór relacji Ziemniaki zawierający jedynie atrybut Dojrzały oraz Rozmiar, w którym ziemniaki ważą 150g.

To zapis w formie drzewa przyjąłby postać:



Niektóre operatory są dwuargumentowe np. ∩,∪, ⋈ co powoduje rozgałęzianie się drzewa. Weźmy ultra trudne zapytanie np.⋈ $(Ziemniak\ A, Ziemniak\ B)_{A.Waga>B.Waga}$ 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

Do Algebry Relacji i jej związku z wykonywaniem zapytań wrócimy na liście 4.

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 *	Opakowanie zapytania. W klauzuli FROM można użyć zapytania.

FROM **([SQL]) ALIAS**	
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**,**	Utworzenie grup po wartościach atrybutu ATT oraz wyliczenie agregacji typu COUNT.

COUNT(*) **FROM** MY_TABLE **GROUP BY** ATT **HAVING** **AVG(**TTT**)** > 10	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.
WITH [nazwa X] AS (**[dowolny SQL]** **)** **[dowolny SQL, w tym odwołujący się do `nazwa X`]**	Common Table Expression (CTE)

3. SQL

JAK PISAĆ SQL-E?

Klucz do sukcesu w pisaniu SQL-a (i jego ocenianiu) to piękne FORMATOWANIE ZAPYTAŃ.

Ogólnie przyjęty przeze mnie sposób formatowania jest następujący: wyobrażamy sobie kilka kolumn do których stosujemy kilka reguł:

- W pierwszej kolumnie umieszczamy **tylko** główne słowa kluczowe / grupy: SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY
- W każdej kolejnej nieparzystej kolumnie umieszczamy zazwyczaj główne słowa kluczowe / grupy: SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY – ale wyjątkiem jest łamanie warunków w JOIN-ach (przykład 2)
- W parzystach zamieszczamy wszystko inne łamiąc wiersze wyrażeniami: INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN, AND, OR (Przykład 2) – choć nie musimy łamać wierszy łącząc warunki w JOINACH (Przykład 4)
- Wyjątkiem gdy zapytanie możemy wpisać całości in-line jest przypadek gdy jest podzapytaniem z 1 kolumną w SELECT i 1 tabelą we FROM
- Wyjątkiem gdy słowo kluczowe i jego argumenty możemy wpisać w jednej linijce jest przypadek gdy słowo kluczowe i argument stanowią łącznie 2 wyrazy (z pominięciem przemianowania) (Przykład 3)

Przykładowe zapytania sformatowane w ten sposób:

Przykład 1:

```
SELECT
	p.Id AS [Id],
	p.IdJednostkaSprawozdawcza AS [UnitId],
	p.Imie AS [Name],
	p.Nazwisko AS [Surname],
	RTRIM(LTRIM(p.Imie + ' ' + p.Nazwisko)) AS [DisplayName],
	p.Aktywny AS [Active],
	p.DataModyfikacji AS [SyncDate]
FROM
	dbo.Pracownik p
```

Przykład 2:

```
SELECT
        f.[Id] AS [Id]
        ,t.[Id] AS [LessonTimeId]
        ,te.[Id] AS [TeacherId]
        ,cat.[Id] AS [CategoryId]
        ,cat.[Active] AS [CategoryActive]
FROM
        [dbo].[UP_UczenFrekwencja] f
        INNER JOIN [dbo].[UP_V_Mobile_LessonTime] t ON t.UnitId = f.IdJednostkaSprawozdawcza
                AND t.Id = f.IdPoraLekcji
        INNER JOIN [dbo].[UP_V_Mobile_Employee] te ON te.UnitId = f.IdJednostkaSprawozdawcza
                AND te.Id = f.IdPracownikModyfikujacy
                AND f.IdTypWpisuFrekwencji = cat.Id
```

Przykład 3:

```
SELECT f.[Id] AS [Id]
FROM [dbo].[UP_UczenFrekwencja] f
```

Przykład 4:

```
SELECT
        Id,
        IdLogin
FROM
        [dbo].[Uczen]
WHERE
        IdLogin IS NOT NULL
UNION ALL
SELECT
        U.Id,
        O.IdLogin
FROM
        [dbo].[Uczen] U
        INNER JOIN [dbo].[Opiekun]  O ON U.IdOpiekun1 = O.Id OR U.IdOpiekun2 = O.Id
WHERE
        O.IdLogin IS NOT NULL
        AND O.Id > 0
```

NO DOBRZE PANIE MAGISTRZE, W CZYM MA MI TO POMÓC?

Może nie jest to ewidentne na początku – ale SQL ma dużo śmieci. Najczęściej błędne działanie SQL-a wynika z klauzuli **WHERE.** Na ogół fragment FROM nie zawiera błędów – jego postać wynika z kluczy obcych w BD. Dostając od kogoś zapytanie takie jak z Przykładu 4 moje (i liczę na to, że w przyszłości Wasze) oczy widzą coś w tym stylu:

Przykład 4:

```
SELECT
        U.Id,
        O.IdLogin
FROM
        [dbo].[Uczen] U
        INNER JOIN [dbo].[Opiekun]  O ON U.IdOpiekun1 = O.Id OR U.IdOpiekun2 = O.Id
WHERE
        O.IdLogin IS NOT NULL
        AND O.Id > 0
```

ROZGRZEWKA

W tej sekcji zamieszczam zapytania na rozgrzewkę – te z chęcią skonsultuję:

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

(10 PKT) ZADANIE

Z repozytorium pobrać szablon sprawozdania – uzupełnić – odesłać.

Uwagi:

- Termin do godziny 8:00 dnia 2020-04-06
- W tym roku nie pobłażam plagiatom – jeśli zauważę reużyty diagram to pracę od razu oceniam w całości na 0 pkt – zadanie możecie mi utrudnić poprzez konsekwentne formatowanie zapytań
- Na ocenę nie będzie tylko wpływać logiczna poprawność zapytania, ale też sposób jego formatowania