Sprawozdanie

Bazy danych: SQL

Numer indeksu:

Spis treści

[Baza danych 4](#_Toc35507884)

[Zapytanie 1 4](#_Toc35507885)

[Semantyka Zapytania 4](#_Toc35507886)

[SQL 4](#_Toc35507887)

[Drzewo operatorów 4](#_Toc35507888)

[Zapytanie 2 5](#_Toc35507889)

[Semantyka Zapytania 5](#_Toc35507890)

[SQL 5](#_Toc35507891)

[Drzewo operatorów 5](#_Toc35507892)

[Zapytanie 3 6](#_Toc35507893)

[Semantyka Zapytania 6](#_Toc35507894)

[SQL 6](#_Toc35507895)

[Drzewo operatorów 6](#_Toc35507896)

[Zapytanie 4 7](#_Toc35507897)

[Semantyka Zapytania 7](#_Toc35507898)

[SQL 7](#_Toc35507899)

[Drzewo operatorów 7](#_Toc35507900)

[Zapytanie 5 8](#_Toc35507901)

[Semantyka Zapytania 8](#_Toc35507902)

[SQL 8](#_Toc35507903)

[Drzewo operatorów 8](#_Toc35507904)

[Zapytanie 6 9](#_Toc35507905)

[Semantyka Zapytania 9](#_Toc35507906)

[SQL 9](#_Toc35507907)

[Drzewo operatorów 9](#_Toc35507908)

[Zapytanie 7 10](#_Toc35507909)

[Semantyka Zapytania 10](#_Toc35507910)

[SQL 10](#_Toc35507911)

[Drzewo operatorów 10](#_Toc35507912)

[Zapytanie 8 11](#_Toc35507913)

[Semantyka Zapytania 11](#_Toc35507914)

[SQL 11](#_Toc35507915)

[Zapytanie 9 12](#_Toc35507916)

[Semantyka Zapytania 12](#_Toc35507917)

[SQL 12](#_Toc35507918)

[Drzewo operatorów 12](#_Toc35507919)

[Zapytanie 10 13](#_Toc35507920)

[Semantyka Zapytania 13](#_Toc35507921)

[SQL 13](#_Toc35507922)

[Drzewo operatorów 13](#_Toc35507923)

# Baza danych

Zapytania należy wykonać na silniku: Sqlite. Binarkę dostępna [tutaj](https://github.com/mpenarprz/BazyDanychI4/blob/master/Laboratorium/tools/sqlite.zip). Plik bazy danych chinook.db dostępny [tutaj](https://github.com/mpenarprz/BazyDanychI4/blob/master/Laboratorium/tools/chinook.db). Plik ze schematem ERD dostępny [tutaj](https://github.com/mpenarprz/BazyDanychI4/blob/master/Laboratorium/tools/sqlite-sample-database-diagram-color.pdf).

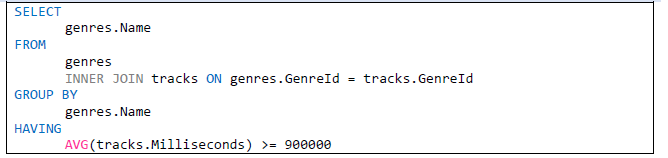
# Zapytanie 1

## Semantyka Zapytania

Zwrócić gatunki których średni czas utworu to **co najmniej** 15 minut.

## SQL

Modelowo napisany SQL:



### Błąd 1

Często popełnianym błędem jest wykorzystanie w SELECT kolumn które nie występują w sekcji GROUP BY (tak nie wolno wg. ANSI SQL) – SQLite wprawdzie dopuszcza taką możliwość, ale nie wiedziałem, żeby ktokolwiek poczynił komentarz odnośnie tego niestandardowego zachowania. Przykładowo:

SELECT genres.Name

FROM

genres

INNER JOIN tracks ON genres.GenreId = tracks.GenreId

GROUP BY genres.GenreId

HAVING AVG(tracks.Milliseconds) >= 900000

Jest błędnym SQL-em: grupowanie jest po **GenreId**, z kolei na wyjściu użytkownik chce kolumnę **Name** – problem w tym, że Name to teraz kolekcja wartości. Dlaczego więc SQLite dopuszcza takiego zachowania? Bo robi niejawnie coś w stylu:

SELECT MAX(genres.Name)

FROM

genres

INNER JOIN tracks ON genres.GenreId = tracks.GenreId

GROUP BY genres.GenreId

HAVING AVG(tracks.Milliseconds) >= 900000

Tak naprawdę funkcja którą SQLite wykorzystuje do zwrócenia wartości genres.Name to FIRST(), ale z tego co pamiętam to jest dostępna tylko wewnętrznie w C++. Równie dobrze można poprosić o tracks.Name – które też nie ma sensu: dostajemy 1 prawie-losową-nazwę utworu wg. danego gatunku.

Pro-tip: na ogół jeśli zachodzi potrzeba obróbki stringów wg. jakiś grup to używamy funkcji GROUP\_CONCAT (konkatenacja)

## Drzewo operatorów

Poprawne najmniejsze drzewo wygląda tak:



# Zapytanie 2

## Semantyka Zapytania

Zwrócić kompozytorów, długości utworów (wyrażone w minutach) oraz ich nazwy – dla których kompozytorem jest ich artysta wykonujący (tabela artist) oraz tytuł zaczyna się na literę N. Zbiór posortować po wg. tytułu.

## SQL

Poprawnie:

SELECT

t.Composer,

t.Milliseconds/1000/60 AS 'Minutes',

t.Name

FROM

tracks t

INNER JOIN albums al ON al.AlbumId=t.AlbumId

INNER JOIN artists ar ON al.ArtistId=ar.ArtistId AND t.Composer=ar.Name

WHERE

t.Name LIKE 'N%'

ORDER BY

t.Name

### Błąd 1

Błędy wynikały najczęściej ze złych złączeń np.:

FROM

tracks t

INNER JOIN artists a ON t.Composer = a.name

Co pomija kwestię kompozytora albumu: nie jest sztuką dodać 2 utworzy z dwóch różnych albumów powiedzmy…. Zoo Station w wykonaniu NIN oraz Hurt w wykonaniu NIN. Kompozytorem pierwszego jest Bono (chyba) drugiego Reznor…. A jednak oba śpiewa Reznor…. Dlatego to złączenie nie będzie poprawne, bo fakt że NIN znalazło się w wykonawcach wpływa na wykonanie tego złączenia przez BD.

OK – teraz mindfcuk - ktoś może tutaj narzekać, że zapytanie dot. istnienia kompozytorów którzy wykonują swoje utwory wymusza by w tabeli tracks w kolumnie Composers znalazł się ten Reznor przy Hurt oraz Reznor jako artysta.

Ale bardzo często artyści piszą piosenki między sobą i przed taką sytuacją nas to złączenie nie zabezpiecza.

### Błąd 2

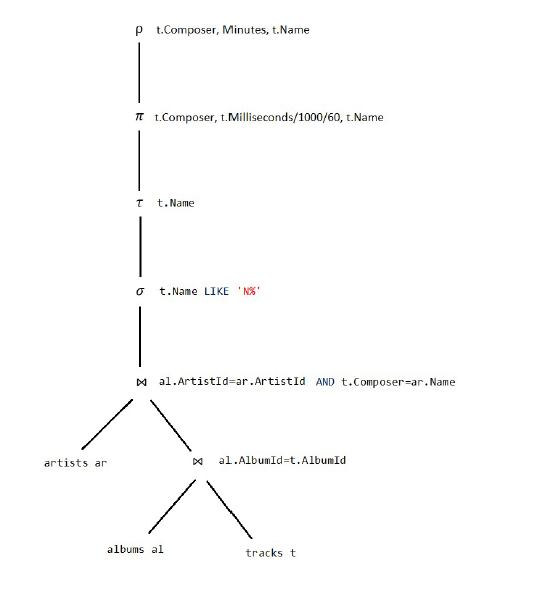
Taki warunek w WHERE też budzi wątpliwości:

SUBSTR(t.Name,1,1) = 'N'

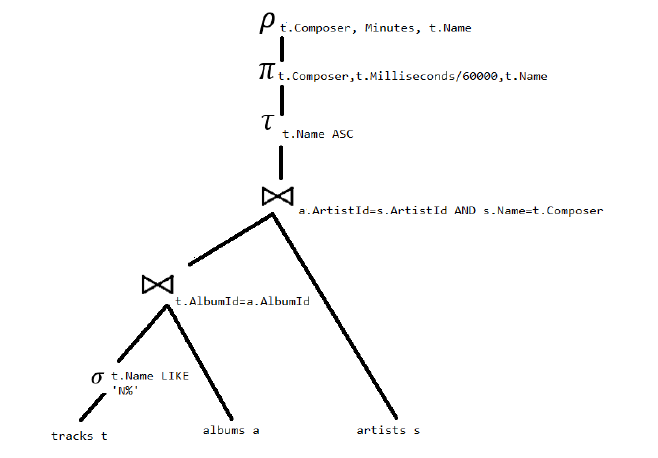
Z powodów o których powiemy sobie na liście …. Którejśtam.

## Drzewo operatorów

OK drzewo:



Jak ktoś przerzucał warunek przed złączenie to dorzucałem w bonusie 0,25:



# Zapytanie 3

## Semantyka Zapytania

Zgłupiałem przy ocenie tego zadania i przyznaje bez bicia, że jest punktowy random. Semantycznie równoważne zapytanie to:

SELECT DISTINCT

t.Composer

FROM

tracks t

INNER JOIN albums a ON t.AlbumId = a.AlbumId

WHERE

NOT EXISTS (SELECT 1 FROM artists s WHERE a.ArtistId = s.ArtistId AND t.Composer = s.name)

AND t.Composer IS NOT NULL

ORDER BY

t.Composer

Zgłupiałem dlatego, że nie wiem jak po polsku określić warunek w EXISTS tj. sformułowanie z pierwszego lepszego sprawozdania: „Zwrócić kompozytorów takich utworów, których **sami nie są wykonawcami** ale mają kompozytorów. Zbiór posortować wg. kompozytora.” – ma pewną nieścisłość: wykonawca nie musi istnieć.

Rzeczywiste znaczenie napisałbym:

*Zwrócić posortowany zbiór unikatowych kompozytorów tych utworów które należą do albumu, ale których artysta wykonujący nie istnieje bądź nie jest taki sam jak kompozytor utworu.*

Nie wiem czy to się dało zgrabnie ująć, dopisuje wszystkim 0,25 i reklamacji nie przyjmuję.

## SQL

SELECT DISTINCT

t.Composer

FROM

tracks t

INNER JOIN albums a ON t.AlbumId = a.AlbumId

LEFT JOIN artists s ON a.ArtistId = s.ArtistId AND t.Composer = s.name

WHERE

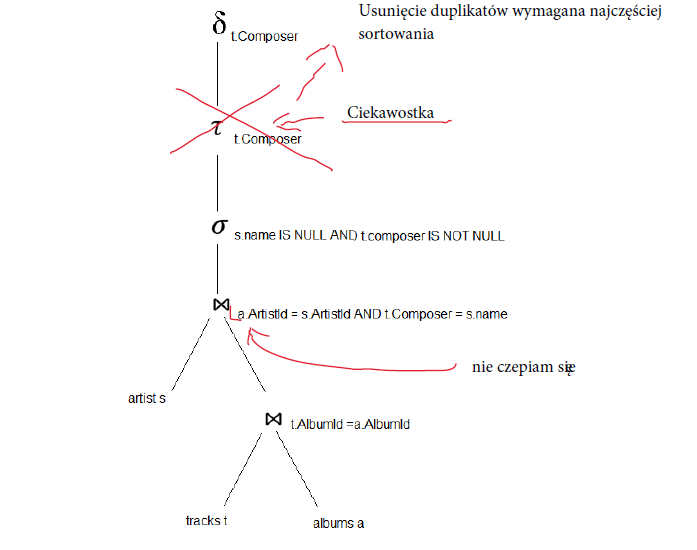
s.name IS NULL

AND t.Composer IS NOT NULL

ORDER BY

t.Composer

## Drzewo operatorów



Czuję się tu zobligowany do małego komentarza: *na tym etapie rozważań nie obcinam punktów za rozwijanie drzew w lewo/prawo.*

# Zapytanie 4

## Semantyka Zapytania

Najbardziej podobało mi się to:

*Zwróć wszystkie pary gatunków i różnicę w ~~ilości~~ liczbie utworów między nimi, dla których ta różnica będzie dodatnia.*

Ja bym napisał:

*Dla każdego gatunku policz o ile utworów więcej znajduje się w każdym innym gatunku. Zwróć te pary gatunków oraz różnicę w liczbie utworów dla których w pierwszym analizowanym gatunku znajduje się ich więcej.*

….

Pojawiło się np.: *Zwróć alfabetycznie nazwy gatunków wraz z nazwami gatunków które mają większą liczbę utworów, oraz różnice ilość tych utworów.*

Skąd wiadomo, że alfabetycznie? :P (Akurat będą)

Podobał mi się też ten autyzm:

*Zwrocić zestawienia nazw dwoch gatunkow (iloczyn kartezjański: drugi gatunek* ***X*** *pierwszy gatunek) i rożnice*

*między całkowitymi ilościami utworow przynależących do poszczegolnych gatunkow z danych par (wartość*

*rożnicy = ilość dla pierwszego gatunku – ilość dla drugiego drugiego). Wyświetlać tylko pary, w ktorych pierwszy*

*gatunek ma większą ilość przyporządkowanych mu utworow niż drugi. Kolumny nazw gatunkow wyświetlać*

*w kolejności: pierwszy gatunek, drugi gatunek.*

Najbardziej nie podobało mi się to stwierdzenie:

*Zwróć nazwy gatunków dla których liczba utworów jest większa od następnego gatunku w tabeli.*

Bo wchodzi na śliski grunt alokacji danych…. „Następny gatunek” sugeruje porównanie z tylko 1 wierszem i jakąś formą „kolejności”

## SQL

WITH Temp(Name, Number) AS (

SELECT

g.name,

COUNT(\*)

FROM

tracks t

INNER JOIN genres g ON t.GenreId = g.GenreId

GROUP BY

g.name

)

SELECT

t2.name,

t1.name,

t2.Number - t1.Number

FROM

TEMP t1,

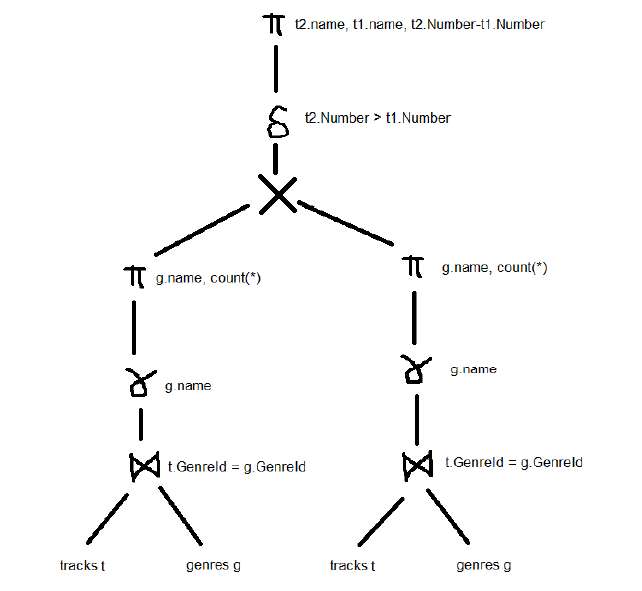
TEMP t2

WHERE

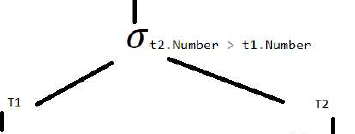
t2.Number > t1.Number;

## Drzewo operatorów

Kanoniczne drzewo:



Dużo było złych bzdur takich jak:



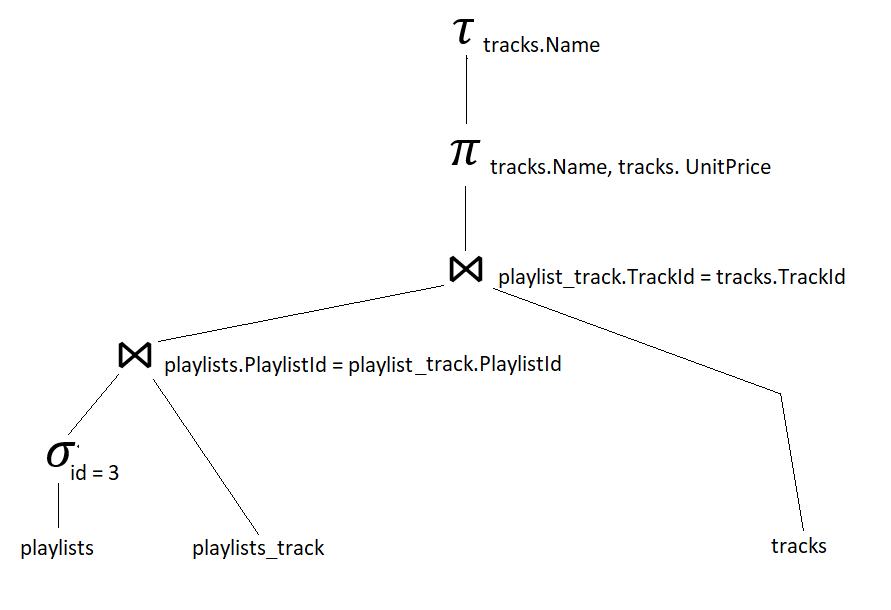
Dwa wejścia dla selekcji tylko dla zapytań skorelowanych dopuszczam. Dla złączeń niestety nie.

# Zapytanie 5

## Semantyka Zapytania

## SQL

## Drzewo operatorów

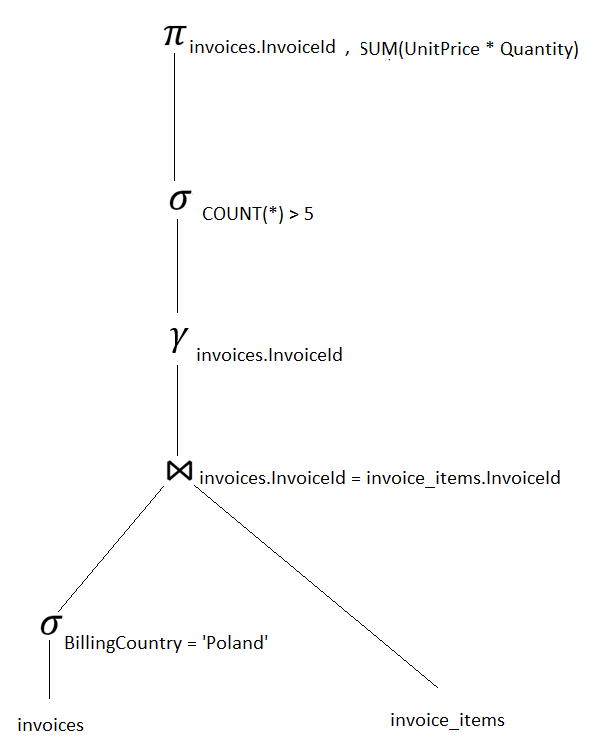


# Zapytanie 6

## Semantyka Zapytania

## SQL

## Drzewo operatorów



# Zapytanie 7

## Semantyka Zapytania

Zwrócić dane klienta który dokonał największą liczbę zakupów (paragonów, nie pozycji paragonów)

## SQL

## Drzewo operatorów

# Zapytanie 8

## Semantyka Zapytania

Zwrócić nazwy oraz całkowitą długość trwania (w minutach) 3 najdłuższych albumów zawierających przynajmniej jeden utwór z gatunku Pop i 3 najdłuższych albumów zawierających przynajmniej jeden utwór z gatunku Jazz (powinno być 6 rekordów, ale nie sprawdzałem).

## SQL

# Zapytanie 9

## Semantyka Zapytania

*Wybrać dowolne zapytanie odwołujące się do 3 tabel. Opisać tu co powinno robić:*

## SQL

## Drzewo operatorów

# Zapytanie 10

## Semantyka Zapytania

Wybrać listę pierwszy-lepszych (wg. identyfikatora) utworów których zbiór wynikowy zawiera blisko 1024 MB danych.

## SQL

## Drzewo operatorów