Jak zdać PiBiDi i nie być skibidi

Autor: PuckMoment - podolog z powołania

Informacje przekazane przed egzaminem

Na kolosie będzie 6 zadań, każde za 1 punkt. Typy tych zadań są następujące:

1. Operacja na tekście
2. Operacja na dacie
3. Złączenie zewnętrzne
4. Agregacja danych
5. Podzapytanie nieskorelowane (z wieloma rezultatami)
6. Podzapytanie skorelowane

Obowiązujące na egzaminie funkcje są następujące:

- Funkcje agregacyjne
  - AVG
  - COUNT
  - MAX
  - MIN
  - SUM
- Funkcje daty i czasu
  - DATEADD
  - DATEDIFF
  - DATENAME
  - DATEPART
  - DAY
  - GETDATE
  - MONTH
  - YEAR
- Funkcje dla tekstów
  - CHARINDEX
  - CONCAT
  - LEFT
  - RIGHT
  - LEN
  - SUBSTRING

Niedozwolone kostrukcje:

- STRING_AGG
- TOP 1 WITH TIES
- WITH (Common Table Expressions)

Czysta teoria

1. Założenia ogólne

Każde zapytanie SELECT może składać się z następujących elementów

- SELECT [kolumny które wybieramy] (5)
- FROM [z jakiej tabeli wybieramy] (1)
- WHERE [jakiś warunek, dotyczący pojedynczych rekordów] (2)
- GROUP BY [kolumny na podstawie których grupujemy dane] (3)
- HAVING [jakiś warunek, dotyczący całych grup] (4)
- ORDER BY [kolumna po której chcemy sortować] (6)

Przy czym SELECT i FROM są obowiązkowe. Liczby w nawiasach po prawej stronie reprezentują kolejność w której zapytanie jest przetwarzane

2. Operacje na tekstach

Wszystkie wymienione wyżej funkcje tekstowe prawdopodobnie na kolosie będą nam potrzebne przy określaniu warunku w WHERE, ale jak trzeba będzie użyć w SELECT to sobie poradzicie na pewno również. Co robią metody po kolei:

- CHARINDEX(a, tekst, start)
  Wyszukuje tekstu **a** w przekazanym tekście i zwraca jego pozycje. Parametr start jest opcjonalny i określa gdzie zacząć przeszukiwanie. Trzeba pamiętać, że duże i małe litery nie mają znaczenia oraz że parametr start oraz sam zwrócony wynik jest liczony od 1 a nie od 0. Jeśli nie znajdzie tekstu to zwraca 0. Przykład:

```
SELECT CHARINDEX('US', 'Big Justice') AS MatchPosition;
```

  Wynik: 6

- CONCAT(string_1, string_2, …., string_n)
  Łączy kilka stringów w jeden.

```
SELECT CONCAT('SQL', ' ', 'is', ' ', 'sigma!');
```

  Wynik: SQL is sigma!

- LEFT(tekst, N) oraz RIGHT(tekst, N)
  Zwracają N znaków od lewej oraz prawej strony z tekstu.

```
SELECT LEFT('Hawk tuah', 4) AS ExtractString;
SELECT RIGHT('Hawk tuah', 4) AS ExtractString;
```

  Wyniki:
  Hawk
  tuah
- LEN(tekst)
  Zwraca ilość znaków w tekście.
  Tak proste, że obejdzie się bez przykładu
- SUBSTRING(tekst, i, N)
  Zwraca N znaków z tekstu zaczynając od znaku na miejscu i. Numerowanie znaków zaczyna się od 1

```
SELECT SUBSTRING('Jestes Skibidi', 8, 5) AS ExtractString;
```

  Wynik:
  Skibi

Poza tymi komendami warto sobie ogarnąć jak działa komenda LIKE. Widziałem zadanie gdzie trzeba wypisać jakieś dane tylko jeśli zawierają więcej niż 2 słowa.

```
SELECT cosie
FROM pierdolety
WHERE cosie LIKE '% %';
```

To oznacza, że jak będzie jakakolwiek spacja w środku to warunek zostanie spełniony.

Znak „%” oznacza, jakikolwiek ciąg znaków (może być pusty).

Znak „_” oznacza, dowolny jeden znak (ale musi występować).

Wszystko inne to po prostu znaki, które muszą się zgadzać, czyli jak zapiszemy `_la` to pasującymi tekstami będą Ola, Ala, Ula, Dla itp.

`% i kropka` oznacza, że tekst musi się kończyć na „ i kropka”.

3. Operacje z datami

Tu może będę tłumaczył ale chuj wie

- DATEADD
- DATEDIFF
- DATENAME
- DATEPART
- DAY
- GETDATE
- MONTH
- YEAR

4. Łączenie tabel

W SQL są 4 rodzaje JOIN'ów:

- (INNER) JOIN: Returns records that have matching values in both tables
- LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table
- RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
- FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table

JOIN'y pozwalają nam połączyć wartości z dwóch tabeli przez co możemy je analizować w obrębie jednego zapytania. W przypadku LEFT, RIGHT i FULL JOIN'a wartości które nie posiadają odpowiadających im wierszy w innych tabelach, to komórki te zostaną uzupełnione NULL'ami, z kolei w INNER JOIN w zapytaniu zostaną tylko wartości, które mają dopasowanie.

Konstrukcja każdego JOIN'a wygląda tak samo.

[RODZAJ] JOIN [jaką tabele dodajemy] ON [wartość z jednej tabeli] = [wartość z drugiej tabeli]

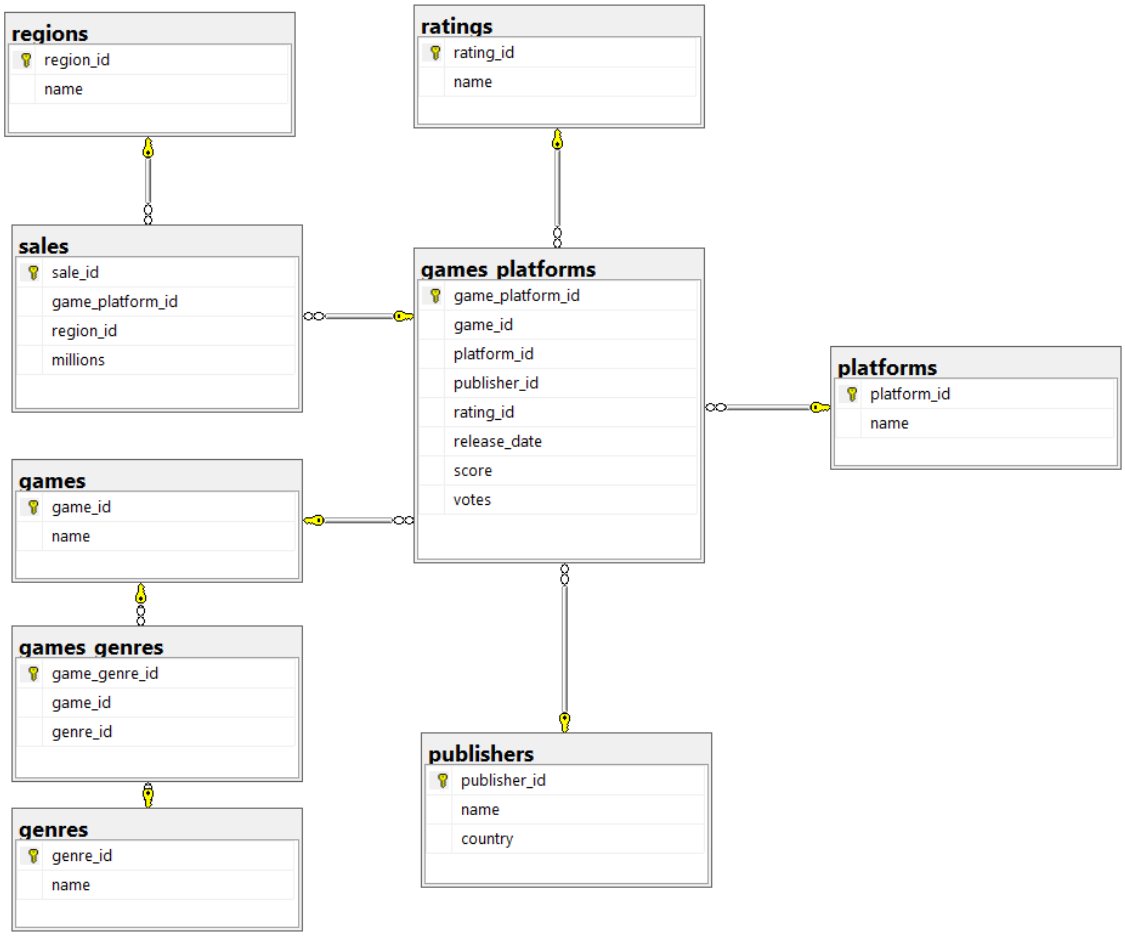
Najczęściej wartości na podstawie, których łączymy to klucze główne i obce. Na kolosie raczej nie będzie sytuacji w której te dane nie będą kluczami.

5. Agregacja danych
6. Podzapytania skorelowane
7. Podzapytania nieskorelowane

Przerobione pytania

Zestaw pytań numer 1 (games)

Dana baza:



1. Wyświetl daty wydania gier, których tytuł i nazwa wydawcy mają tę samą długość. Wyniki posortuj według tytułu w porządku malejącym.

```
SELECT release_date
FROM games_platforms
JOIN publishers ON (games_platforms.publisher_id = publishers.publisher_id)
JOIN games ON (games_platforms.game_id = games.game_id)
WHERE LEN(games.name) = LEN(publishers.name)
ORDER BY games.name DESC;
```

2. Wyświetl nazwy kategorii wiekowych gier, które miały premierę w ciągu ostatnich 60 miesięcy. Zastosuj alias „kategoria”.

```
SELECT ratings.name AS kategoria
FROM games_platforms
JOIN ratings ON (ratings.rating_id = games_platforms.rating_id)
WHERE games_platforms.release_date >= DATEADD(MONTH, -60, GETDATE());
```

3. Wyświetl nazwy platform, na które nie zostały wydane żadne gry.

```
SELECT name
FROM platforms
LEFT JOIN games_platforms ON platforms.platform_id =
games_platforms.platform_id
WHERE games_platforms.game_platform_id IS NULL;
```

4. Dla każdego kraju pochodzenia wydawców gier wyświetl jego nazwę oraz liczbę sprzedanych w milionach sztuk najlepiej sprzedającej się gry na pojedynczą platformę. Uwzględnij tylko te kraje, dla których wartość jest równa co najwyżej 5.

```
SELECT country, MAX(s.millions) AS sales
FROM publishers p
JOIN games_platforms gp ON gp.publisher_id = p.publisher_id
JOIN sales s ON s.game_platform_id = gp.game_platform_id
GROUP BY country
HAVING MAX(s.millions) <= 5;
```

5. Wyświetl tytuły gier, które zostały wydane na najmniejszej liczbie platform.

```
SELECT g.name
FROM games g
JOIN games_platforms gp ON gp.game_id = g.game_id
GROUP BY g.game_id, g.name
HAVING COUNT(gp.game_platform_id) = (
    SELECT MIN(platform_count)
    FROM (
        SELECT g.game_id, COUNT(gp.platform_id) AS platform_count
        FROM games g
        JOIN games_platforms gp ON g.game_id = gp.game_id
        GROUP BY g.game_id
    ) AS platform_counts
)
```

6. Dla każdej kategorii wiekowej wyświetl jej nazwę oraz nazwy gier, które zostały wydane najwcześniej.

```
SELECT r.name, g.name
FROM ratings r
JOIN games_platforms gp ON gp.rating_id = r.rating_id
JOIN games g ON gp.game_id = g.game_id
WHERE gp.release_date = (
    SELECT MIN(release_date)
    FROM games_platforms gp_pod
    WHERE gp_pod.rating_id = r.rating_id
    AND gp.game_id = g.game_id
)
```

Zestaw pytań numer 2 (books) (dokładnie to miałem, zapytania ocenione na maks punktów)

1. Wyświetl numery egzemplarzy wypożyczonych przez czytelników, których pierwsza i ostatnia litera imienia są identyczne. Wyniki posortuj według daty wypożyczenia w porządku malejącym.

```
SELECT c.copy_id
FROM copies c
JOIN orders o ON o.copy_id = c.copy_id
JOIN readers r ON r.reader_id = o.reader_id
WHERE LEFT(r.first_name, 1) = RIGHT(r.first_name, 1)
ORDER BY o.order_date DESC;
```

2. Wyświetl z pominięciem duplikatów imiona i nazwiska autorów książek, które zostały wydaje ponad 30 lat temu. Zastosuj aliasy „imię” i „nazwisko”.

```
SELECT DISTINCT a.first_name AS 'imię', a.last_name AS 'nazwisko'
FROM authors a
JOIN books_authors ba ON ba.author_id = a.author_id
JOIN books b ON b.book_id = ba.book_id
WHERE b.publication_date < DATEADD(yyyy, -30, GETDATE());
```

3. Wyświetl nazwy gatunków, do których nie należą żadne książki.

```
SELECT g.name
FROM genres g
LEFT JOIN books_genres bg ON bg.genre_id = g.genre_id
WHERE bg.book_id IS NULL;
```

4. Dla każdego autora wyświetli jego imię i nazwisko oraz najlepszą ocenę jego dowolnej książki. Uwzględnij tylko tych autorów, dla których wartość ta jest równa co najmniej 4.3.

```sql
SELECT a.first_name, a.last_name, MAX(b.rating) AS 'najwyższa ocena'
FROM authors a
JOIN books_authors ba ON ba.author_id = a.author_id
JOIN books b ON b.book_id = ba.book_id
GROUP BY a.author_id, a.first_name, a.last_name
HAVING MAX(b.rating) >= 4.3;
```

5. Wyświetl nazwy wydawnictw, które wydały najmniejszą liczbę książek.

```sql
SELECT p.name
FROM publishers p
JOIN books b ON p.publisher_id = b.publisher_id
GROUP BY p.publisher_id, p.name
HAVING COUNT(b.book_id) = (
    SELECT MIN(ksiazki)
    FROM (
        SELECT COUNT(b.book_id) as ksiazki
        FROM publishers p
        JOIN books b ON p.publisher_id = b.publisher_id
        GROUP BY p.publisher_id, p.name
    ) AS subquery
);
```

6. Dla każdego gatunku wyświetl jego nazwę oraz tytuły książek, które mają najwięcej stron.

```sql
SELECT g.name, b.title, b.pages
FROM genres g
JOIN books_genres bg ON bg.genre_id = g.genre_id
JOIN books b ON b.book_id = bg.book_id
WHERE b.pages = (
    SELECT MAX(b1.pages)
    FROM books b1
    JOIN books_genres bg1 ON b1.book_id = bg1.book_id
    GROUP BY bg1.genre_id
    HAVING bg1.genre_id = g.genre_id
)
```

Zestaw pytań numer 3 (tracks)

```sql
-- ZAD 1
SELECT t.bytes
FROM tracks t
JOIN genres g ON g.genre_id = t.genre_id
WHERE g.name LIKE '% %'
ORDER BY t.milliseconds DESC;
go

-- ZAD 2
SELECT DISTINCT c.first_name AS 'imi', c.last_name AS 'nazwisko'
FROM customers c
JOIN orders o ON o.customer_id = c.customer_id
WHERE DATEPART(mm, o.order_date) = 3
OR  DATEPART(mm, o.order_date) = 9;
go

-- ZAD 3
SELECT t.title
FROM tracks t
LEFT JOIN playlists_tracks pt ON pt.track_id = t.track_id
WHERE pt.playlist_id IS NULL;

-- ZAD 4
SELECT ar.name, AVG(t.milliseconds) as 'sredni czas utworow'
FROM artists ar
JOIN albums al ON al.artist_id = ar.artist_id
JOIN tracks t ON al.album_id = t.album_id
GROUP BY ar.artist_id, ar.name
HAVING AVG(t.milliseconds) < 250000;
go

-- ZAD 5
SELECT c.first_name, c.last_name
FROM customers c
JOIN orders o ON o.customer_id = c.customer_id
GROUP BY c.customer_id, c.first_name, c.last_name
HAVING COUNT(o.order_id) = (
    SELECT MAX(zamowienia)
    FROM (
        SELECT COUNT(o.order_id) AS zamowienia
        FROM customers c
        JOIN orders o ON o.customer_id = c.customer_id
        GROUP BY c.customer_id
    ) AS azsasas
)
```

```sql
-- ZAD 6
SELECT o.order_date, t.title
FROM orders o
JOIN orders_tracks ot ON o.order_id = ot.order_id
JOIN tracks t ON t.track_id = ot.track_id
WHERE t.bytes = (
    SELECT MIN(bytes)
    FROM tracks t1
    JOIN orders_tracks ot1 ON t1.track_id = ot1.track_id
    WHERE ot1.order_id = ot.order_id
);
go
```

Zestaw pytań nr 4 (movies)

```sql
-- ZAD 1
SELECT ma.role
FROM movies_actors ma
JOIN actors a ON ma.actor_id = a.actor_id
WHERE LEN(a.first_name) = LEN(a.last_name)
ORDER BY a.gender;
go

-- ZAD 2
SELECT DISTINCT r.name AS 'recenzent'
FROM reviewers r
JOIN reviews re ON re.reviewer_id = r.reviewer_id
JOIN movies m ON m.movie_id = re.movie_id
WHERE DATEPART(mm, m.release_date) = 7
OR DATEPART(mm, m.release_date) = 12;

-- ZAD 3
SELECT a.first_name, a.last_name
FROM actors a
LEFT JOIN movies_actors ma ON a.actor_id = ma.actor_id
WHERE ma.movie_id IS NULL;

-- ZAD 4
SELECT g.name, MIN(m.minutes) AS dlugosc
FROM genres g
JOIN movies_genres mg ON mg.genre_id = g.genre_id
JOIN movies m ON mg.movie_id = m.movie_id
GROUP BY g.genre_id, g.name
HAVING MIN(m.minutes) > 120;
```

```sql
-- ZAD 5
SELECT d.first_name, d.last_name
FROM directors d
JOIN movies_directors md ON d.director_id = md.director_id
GROUP BY d.director_id, d.first_name, d.last_name
HAVING COUNT(md.movie_id) = (
    SELECT MIN(skibidiliczenie)
    FROM (
        SELECT COUNT(md.movie_id) AS skibidiliczenie
        FROM directors d
        JOIN movies_directors md ON d.director_id = md.director_id
        GROUP BY d.director_id, d.first_name, d.last_name
    ) AS skibidizapytanie
);
go

-- ZAD 6
SELECT a.first_name, a.last_name, m.title
FROM actors a
JOIN movies_actors ma ON a.actor_id = ma.actor_id
JOIN movies m ON m.movie_id = ma.movie_id
WHERE m.minutes = (
    SELECT MAX(m1.minutes)
    FROM movies m1
    JOIN movies_actors ma1 ON ma1.movie_id = m1.movie_id
    WHERE ma1.actor_id = a.actor_id
)
```