**Backend Test**

**BACA INI DULU:**

1. Gunakan Bahasa Inggris atau Bahasa Indonesia dalam semua jawaban Anda.
2. Ada 5 pertanyaan tanpa urutan kesulitan tertentu.
3. Anda dapat meninggalkan pertanyaan yang ditandai dengan "Untuk diskusi nanti".
4. Anda harus dapat menjelaskan semua jawaban Anda selama wawancara berikut. Ini juga berlaku jika Anda menyalin jawaban Anda dari referensi eksternal apa pun.
5. Unggah karya Anda di gitlab atau github.

* Jawab pertanyaan dalam file markdown (.md), dan untuk pertanyaan pemrograman, jawab dalam file terpisah menggunakan python (.py).

# Pertanyaan #1 (pemeliharaan)

Saat memecahkan masalah bug di aplikasi lama, Anda menemukan bahwa metode "multiply" di bawah ini adalah sumber masalahnya.

**CODE**:

///

/// multiplies x times y

///

**def multiply(x, y):**

**total = 0**

**while x > 0:**

**total += y**

**x -= 1**

**return total**

QUESTIONS:

1. Apa yang salah dengan kode di atas?

Jawab:

Tidak memanggil kembali function multiply(2,3) sebagai contoh..

1. Jika Anda merasa salah, harap perbaiki kode di atas tanpa menggunakan operator "\*" atau "/" atau panggilan Absolut?

Jawab:

Tidak ada yang salah. Hanya penambahan pemanggilan function seperti :

multiply(parameter 1, parameter 2)

1. Sebagai bagian dari proses pengembangan kami, kami menguji semua metode pada tingkat kode. Nilai input mana yang akan Anda gunakan untuk melakukan pengujian?

Jawab:

Nilai input dua parameter yang bersifat numerik.

1. Untuk diskusi nanti: Apa lagi yang membuat Anda khawatir saat Anda memperbaiki masalah ini?

* Belum ada kekhawatiran

# Question #2 (SQL)

Given these two tables:

Table: USA\_CUSTOMERS (USA)

|  |  |
| --- | --- |
| ID | NAME |
| 1 | Thomas |
| 3 | Cindy |

Table: EU\_CUSTOMERS (EU)

|  |  |
| --- | --- |
| ID | NAME |
| 2 | Francois |
| 1 | Thomas |

What would be the output of the following select statements?

|  |  |  |
| --- | --- | --- |
| Select USA.NAME, EU.NAME From USA, EU Where USA.ID = EU.ID | Select USA.NAME, EU.NAME From USA left join EU on (USA.ID = EU.ID) | Select USA.NAME, EU.NAME From USA, EU |
| NAME NAME(1)  Thomas Thomas | NAME NAME(1)  Thomas Thomas  Cindy | NAME NAME(1)  Cindy Thomas  Thomas Thomas  Cindy Francois  Thomas Francois |

**Untuk diskusi nanti**: kami menggunakan tabel tersebut untuk melacak pelanggan Eropa dan Amerika kami. Tolong berikan kritik untuk desain tabel itu (apakah itu bagus? Bagaimana bisa lebih baik?).

Jawab:

Table bagus tergandung data mau diambil berdasarkan permintaan kasus,

Sedangkan untuk lebih baik itu tergantung kasus juga, seperti :

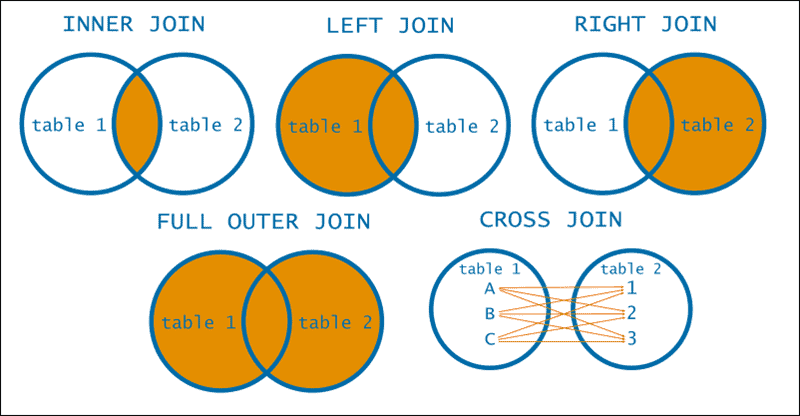
1. INNER JOIN – Hasil mengembalikan data yang cocok dari kedua tabel.

2. LEFT OUTER JOIN – Hasil berasal dari tabel kiri dan data yang cocok dari tabel kanan.

3. RIGHT OUTER JOIN – Hasil berasal dari tabel kanan dan data yang cocok dari tabel kiri.

4. FULL OUTER JOIN – Hasil berasal dari kedua tabel ketika ada data yang cocok.

5. CROSS JOIN – Hasil adalah kombinasi dari setiap baris dari tabel yang digabungkan.



Sintaks untuk INNER JOIN adalah:

SELECT table1.column1, table1.column2, table2.column1, ...

FROM table1

INNER JOIN table2

ON table1.matching\_column = table2.matching\_column;

Sintaks untuk LEFT OUTER JOIN adalah:

SELECT table1.column1, table1.column2, table2.column1, ...

FROM table1

LEFT JOIN table2

ON table1.matching\_column = table2.matching\_column;

Sintaks untuk RIGHT OUTER JOIN adalah:

SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

RIGHT JOIN table2

ON table1.matching\_column = table2.matching\_column;

Sintaks FULL OUTER JOIN adalah:

SELECT \* FROM table1

LEFT JOIN table2 ON table1.matching\_column = table2.matching\_column

UNION ALL

SELECT \* FROM table1

RIGHT JOIN table2 ON table1.matching\_column = table2.matching\_column

Sintaks untuk CROSS JOIN adalah:

SELECT table1.column1, table1.column2, table2.column1, ...

FROM table1

CROSS JOIN table2;

# Question #3 (algorithm)

Kelas berikut menjelaskan node pada pohon biner. Sebuah node dapat memiliki anak kiri, atau anak kanan, atau keduanya, atau tidak ada anak sama sekali.

**CODE**:

**class Node:**

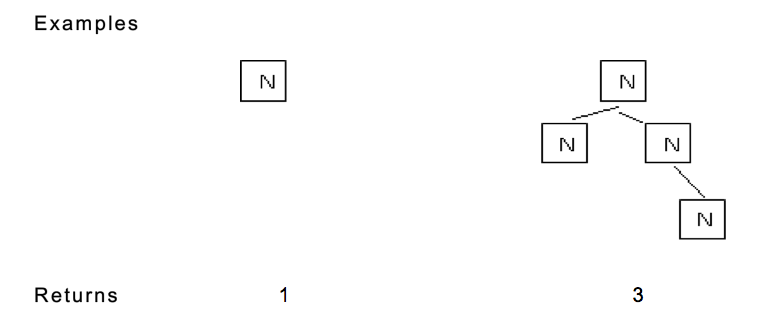
**def \_\_init\_\_(self):**

**self.right = None**

**self.left = None**

**QUESTION:**

Tulis konten metode di bawah ini yang menghitung jumlah maksimum level dalam pohon tertentu. Harap perhatikan bahwa ini TIDAK menghitung jumlah TOTAL node, tetapi menghitung DEPTH.

****

Jawab:

class Node:

def \_\_init\_\_(self, data):

self.data = data

self.right = None

self.left = None

def max\_depth(root):

if root is None:

return 0

left\_depth = max\_depth(root.left)

right\_depth = max\_depth(root.right)

return max(left\_depth, right\_depth) + 1

# Contoh penggunaan

# Membangun pohon biner

root = Node(10)

root.left = Node(5)

root.right = Node(15)

root.left.left = Node(3)

root.left.right = Node(8)

root.right.right = Node(20)

# Memanggil metode untuk menghitung kedalaman maksimum

depth = max\_depth(root)

print("Kedalaman maksimum pohon: ", depth)

Hasilnya adalah:

Kedalaman maksimum pohon: 3

# Question #4 (performance)

Metode berikut akan menemukan persimpangan (duplikat) dari dua set yang diberikan.

**CODE**:

**def intersect(bagA, bagB):**

**result = []**

**for o in bagA:**

**if o in bagB:**

**result.append(o)**

**return result**

**QUESTIONS:**

Apa yang akan menjadi efek pada kinerja dalam dua kasus ini:

|  |  |  |
| --- | --- | --- |
|  | **bagA** | **bagB** |
| Case 1 | Has LARGE number of elements | Has SMALL number of elements |
| Case 2 | Has SMALL number of elements | Has LARGE number of elements |

Jawab: fungsi diatas adalah mencari nilai yang sama antara bagA dan bagB

Contoh:

bagA=[1,2,3,4,5]

bagB=[2,7,3,9,1]

intersect(bagA, bagB)

**maka hasilnya: [1, 2, 3]**

jika untuk menjawab case 1 contoh data seperti ini:

bagA=[11,12,13,14,15]

bagB=[1,2,3,4,5]

intersect(bagA, bagB)

**maka hasilnya adalah []**

sedangkan untuk menjawab case 2 contoh data seperti berikut:

bagA=[1,2,3,4,5]

bagB=[11,12,13,14,15]

intersect(bagA, bagB)

**maka hasilnya adalah []**

Apakah Anda punya rekomendasi untuk meningkatkan kinerja? Jangan ragu untuk mengubah metode di atas.

Jawab: Ada

bagA={1,2,3,4,5}

bagB={2,7,3,9,1}

print(bagA.intersection(bagB))

**Maka Hasilnya : {1, 2, 3}**

# Question #5 (algorithm)

Bilangan prima didefinisikan sebagai bilangan bulat positif lebih besar dari 1 yang hanya dapat dibagi oleh 1 dan dirinya sendiri. Ketika angka dibagi dengan bilangan bulat positif lainnya, itu akan memiliki sisa.

Misalnya:

7 adalah bilangan prima karena dapat dibagi dengan 1 dan 7 saja.

9 BUKAN bilangan prima karena dapat dibagi dengan 3.

PERTANYAAN:

Silakan tulis metode untuk mencetak SEMUA bilangan prima antara 2 dan 100.

def is\_prima (x):

if x < 2:

return False

for i in range(2, x):

if x % i == 0:

return False

return True

def cari\_bilangan\_prima (awal, akhir):

list\_bilangan\_prima = []

for x in range(awal, akhir + 1):

if is\_prima(x):

list\_bilangan\_prima.append(x)

return list\_bilangan\_prima

print(cari\_bilangan\_prima(2, 100))

**Maka Hasilnya : [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]**

**END OF DOCUMENT**