isaac
Incrivelmente simples

Resolução Desafio Técnico

Candidato: Lucas Ferreira da Silva

Parte 1 - SQL

Recomendamos utilizar o SQLFiddle, PostgreSQL como engine de SQL. Pense em usar CTEs e WindowFunctions para lhe auxiliar. Assuma um banco PostgreSQL, timezone do server é UTC.

1. Dada a seguinte tabela com os campos:

Updated_at: timestamp
Id: int64 Name: string
Age: int64

updated_at	id	name	age
2021-03-25 12:59:30 UTC	4	Joao Silva	28
2021-03-25 12:59:15 UTC	4	João Silva	28
2021-03-24 19:50:30 UTC	4	João Sil	28
2021-03-24 19:46:30 UTC	4	João Silva	28
2021-03-24 19:45:30 UTC	1	Marcos Santana	25
2021-03-25 19:44:30 UTC	1	Marcos Santana Santos	25
2021-03-24 19:44:30 UTC	3	Airton	60
2021-03-24 19:44:30 UTC	5	Aurélio	35
2021-03-24 19:44:30 UTC	6	Carushow	27
2021-03-24 19:44:30 UTC	7	Perlita	29
2021-03-28 00:00:30 UTC	7	Perlita	30
2021-03-24 19:44:30 UTC	2	Leonardo	26
2021-03-24 15:44:20 UTC	4	João Silva	28
2021-03-24 14:20:30 UTC	4	João Silva	28
2021-03-24 13:44:30 UTC	4	João Silva	28
2021-03-24 12:25:30 UTC	4	João Silva	28
2021-03-24 10:44:30 UTC	4	João Silva	22

Retorne somente as linhas mais atualizadas para cada ID, ignorando qualquer eventual mudança nocaminho. (Entregue o código SQL e um print screen do resultado)

Resposta:

Inicialmente, vamos criar o SCHEMA no SQL Fiddle conforme informações fornecidas. Após, vamos realizar a query conforme solicitação.

Código para criar o SCHEMA:

```
CREATE TABLE tabela_registros (
            updated_at TIMESTAMP NOT NULL,
            id INT NOT NULL,
            name VARCHAR(100) NOT NULL,
            age INT NOT NULL
);

INSERT INTO tabela_registros VALUES
('2021-03-25 12:59:30', 4, 'Joao Silva', 28),
('2021-03-25 12:59:15', 4, 'Joao Silva', 28),
('2021-03-24 19:50:30', 4, 'João Sil', 28),
('2021-03-24 19:46:30', 4, 'João Silva', 28),
('2021-03-24 19:45:30', 1, 'Marcos Santana', 25),
('2021-03-25 19:44:30', 1, 'Marcos Santana Santos', 25),
('2021-03-24 19:44:30', 3, 'Airton', 60),
('2021-03-24 19:44:30', 5, 'Aurélio', 35),
('2021-03-24 19:44:30', 6, 'Carushow', 27),
('2021-03-24 19:44:30', 7, 'Perlita', 29),
('2021-03-28 00:00:30', 7, 'Perlita', 30),
('2021-03-24 19:44:30', 2, 'Leonardo', 26),
('2021-03-24 15:44:20', 4, 'João Silva', 28),
('2021-03-24 14:20:30', 4, 'João Silva', 28),
('2021-03-24 13:44:30', 4, 'João Silva', 28),
('2021-03-24 12:25:30', 4, 'João Silva', 28),
('2021-03-24 10:44:30', 4, 'João Silva', 22);
```

Query

Desejamos exibir id, name, age e o valor mais atual (MAX) de updated_at, da tabela_registros, agrupando a exibição pelo id. Assim:

```
SELECT MAX(updated_at), id, name, age
FROM tabela_registros
GROUP BY id;
```

isaac
Incrivelmente simples

Resultado da consulta

MAX(updated_at)	id	name	age
2021-03-25T19:44:30Z	1	Marcos Santana	25
2021-03-24T19:44:30Z	2	Leonardo	26
2021-03-24T19:44:30Z	3	Airton	60
2021-03-25T12:59:30Z	4	Joao Silva	28
2021-03-24T19:44:30Z	5	Aurélio	35
2021-03-24T19:44:30Z	6	Carushow	27
2021-03-28T00:00:30Z	7	Perlita	29

2. Dada as duas tabelas abaixo (vendas_cardapio e itens_cardapio), responda às seguintes perguntas (resposta e query utilizada):

Tabela: vendas_cardapio

data	id_item	preco	qt_vendidas
09/06/2020	111	6.5	32
10/06/2020	200	7	22
10/06/2020	340	5.25	15
11/06/2020	111	8	50
11/06/2020	340	5.5	9
10/06/2020	111	6.5	20
11/06/2020	200	7	10

Tabela: itens_cardapio

id_item	nome	gluten	media_preco
111	Café expresso	nao	6.5
112	Cookie de Laranja	sim	8.49
200	Suco de Abacaxi	nao	7
340	Pão de queijo	sim	5

Resposta

Inicialmente, vamos criar o SCHEMA no SQL Fiddle conforme informações fornecidas. Após, vamos realizar as queries conforme solicitação.

Código para criar o SCHEMA:

```
CREATE TABLE vendas_cardapio (
            data DATE NOT NULL,
            id_item INT NOT NULL,
            preco FLOAT NOT NULL,
            qt_vendidas INT NOT NULL
);

CREATE TABLE itens_cardapio (
            id_item INT NOT NULL,
            nome VARCHAR(30) NOT NULL,
            gluten VARCHAR(3) NOT NULL,
            media_preco FLOAT NOT NULL
);
```

```
INSERT INTO vendas_cardapio VALUES
('2020-06-09', 111, 6.5, 32),
('2020-06-10', 200, 7, 22),
('2020-06-10', 340, 5.25, 15),
('2020-06-11', 111, 8, 50),
('2020-06-11', 340, 5.5, 9),
('2020-06-10', 111, 6.5, 20),
('2020-06-11', 200, 7, 10);

INSERT INTO itens_cardapio VALUES
(111, 'Café expresso', 'nao', 6.5),
(112, 'Cookie de Laranja', 'sim', 8.49),
(200, 'Suco de Abacaxi', 'nao', 7),
(340, 'Pão de queijo', 'sim', 5);
```

a. Qual foi o nome do produto que mais teve quantidades vendidas durante o período todo (dia 09 ao 11)?

Resposta: Desejamos retornar o nome do produto que mais vendeu. Iremos precisar fazer uma junção interna entre as tabelas vendas_cardapio e itens_cardapio, usando a coluna id_item comum a ambas para a junção. Nessa tabela resultante, teremos os valores do nome e qt_vendidas, e podemos fazer a seleção considerando o nome e valor da soma (SUM) das qt_vendidas. Após, iremos exibir os dados considerando um agrupamento pelo nome, exibindo somente o primeiro item, que é o mais vendido. O item mais vendido foi o Café expresso, com 102 unidades vendidas no total.

Query para retornar nome e qtd_total

```
SELECT itens_cardapio.nome, SUM(vendas_cardapio.qt_vendidas) AS qtd_total
FROM itens_cardapio
INNER JOIN vendas_cardapio ON vendas_cardapio.id_item = itens_cardapio.id_item
GROUP BY itens_cardapio.nome
LIMIT 1;
```

Resultado da consulta

nome	qtd_total
Café expresso	102

Query para retornar somente nome

```
SELECT itens_cardapio.nome
FROM itens_cardapio
```

```
INNER JOIN vendas_cardapio ON vendas_cardapio.id_item = itens_cardapio.id_item
GROUP BY itens_cardapio.nome
LIMIT 1;
```

Resultado da consulta

nome
Café expresso

b. Qual foi o nome do produto que não teve alteração de preço durante os reajustes? Esse produto tem glúten?

Resposta: O item que não teve reajuste corresponde ao item que tem o preco = media_preco e que também não teve alteração no preço em nenhum momento, pois podemos ter itens em que algum momento o preco foi igual a media_preco, porém, reajustou depois. Neste caso, vamos fazer uma junção das tabelas itens_cardapio e vendas_cardapio, considerando a coluna item comum a ambas. Nesta tabela resultante, podemos fazer um filtro testando se o maior valor do preco de um item (CEILING) é diferente da média, ou seja, se para este item em algum momento ele sofreu alteração. Isto irá nos retornar os itens únicos em que o preco = media_preco e que nunca tiveram alteração de valor. No caso, foi o Suco de Abacaxi, que não tem glúten.

Query

```
SELECT distinct itens_cardapio.nome, itens_cardapio.gluten
FROM itens_cardapio
RIGHT JOIN vendas_cardapio ON vendas_cardapio.id_item = itens_cardapio.id_item
AND ceiling(vendas_cardapio.preco) = itens_cardapio.media_preco
WHERE vendas_cardapio.preco = itens_cardapio.media_preco;
```

Resultado da consulta

nome	gluten
Suco de Abacaxi	nao

c. Houve algum produto que não vendeu?

Resposta: Iremos juntar as tabelas vendas_cardapio e itens_cardapio fazendo uma junção pelo lado direito, dessa forma, os itens da tabela Itens_cardapio que não tem correspondente em vendas_cardapio – e,

portanto, não tiveram vendas – irão ser preenchidos com NULL. Após, é só filtrar pelos itens que são nulos (IS NULL) para exibir o que não vendeu. No caso, seria o Cookie de Laranja.

Query

```
SELECT itens_cardapio.nome
FROM vendas_cardapio
RIGHT JOIN itens_cardapio ON vendas_cardapio.id_item = itens_cardapio.id_item
WHERE vendas_cardapio.qt_vendidas IS NULL;
```

Resultado da consulta

nome
Cookie de Laranja

d. Qual foi o valor total de vendas de cada dia?

Resposta: Neste caso, queremos exibir uma tabela com um filtro considerando o agrupamento da soma dos produtos entre qt_vendidas por preco.

Query

```
SELECT vendas_cardapio.data, SUM(vendas_cardapio.qt_vendidas * vendas_cardapio.preco) AS 'Valor Total Dia'
FROM vendas_cardapio
GROUP BY vendas_cardapio.data;
```

Resultado da consulta

data	Valor Total Dia
2020-06-09	208
2020-06-10	362.75
2020-06-11	519.5

Parte 2 - Programação

1. Inverter os valores após os valores inteiros (casas decimais)

   Exemplo:

   Input 234.567

   Output esperado 432.765

Resposta

Segue rotina comentada que executa a operação solicitada. No programa abaixo, podemos enviar qualquer número float / int ou qualquer string com números que será feita a operação de inversão com a lógica proposta. Ao final foram elaborados alguns testes automatizados para checar a operação.

```
def inverter_posicoes(numero):
    """
    Esta função inverte os valores de número inteiro no espaço delimitado pelos separadores de
    unidades do sistema decimal. Por exemplo, se for enviado o número:
        4, a função retorna 4
        34, a função retorna 43
        234, a função retorna 432
        4.567, a função retorna 4.765
        34.567, a função retorna 43.765
        234.567, a função retorna 432.765
        123.456.789, a função retorna 321.654.987

    Estrutura da função:
        1 - Trata o número recebido, de modo que strings ou inteiros possam ser enviados,
        mas somente é feita inversão na parte inteira do número;
        2 - Quebra o número em partes para efetuar inversão;
        3 - Efetua a inversão conforme lógica acima;

    Lógica da função:
        i   - Ler um número ABC.DEF.GHI qualquer
        ii  - Transformar este número em uma lista de strings ['A','B','C','D','E','F','G','H','I']
        iii - Inverter a lista ['I','H','G','F','E','D','C','B','A']
        iv  - Converter em subset de 3 em 3 [['I','H','G'],['F','E','D'],['C','B','A']]
        v   - Inverter novamente [['C','B','A'], ['F','E','D'], ['I','H','G']]
        vi  - Converter em lista simples ['C','B','A', 'F','E','D', 'I','H','G']
        vii - Gerar o número de saída CBA.FED.IHG

    Input: numero (int) -> número inteiro em que se deseja realizar a operação
    Output: numero invertido (str) -> string com o número invertido com pontuação
    """

    # 1 - Tratar o número, adicionando separadores de unidades para impressão
    # e garantir que possa ser enviado float, int ou str
    numero_formatado = formatar_numero(numero)

    # 2 - Quebrar o número em uma lista separada de 3 em 3, pois temos sempre
    #três casas decimais no sistema de numeração decimal
    numero_quebrado = quebrar_numero(numero_formatado)
```

```
    # 3 - Inverter a lista separada em 3 em 3 e gerar número
    numero_invertido = inverter_lista(numero_quebrado)

    return numero_invertido


def formatar_numero(numero):
    """ Esta função recebe um int, float ou str e retorna um inteiro
    contendo pontos como separador de unidades. Por exemplo, se enviarmos
    '1234', 1234.00, '1234.00' ou 1234 a função retorna 1.234

    Input: numero (int,str) numero que irá receber formatação
    Return: int formatado conforme descrito acima
    """
    if type(numero) == int:
        #números inteiros permanecem como estão
        numero_formato_int = numero
    elif type(numero) == str or type(numero) == float:
        #esta conversão garante que strings de floats serão convertidas em , '123456.00' vira 123456
        numero_formato_int = int(float(numero))
    else:
        #se tiver um formato diferente, retorna um erro
        return ValueError("Operação não permitida neste formato de dado.")
    #retorna o número_formato_int 123456789 em 123.456.789
    return format(numero_formato_int,',d').replace(",",".")

def quebrar_numero(numero):
    """Esta função quebra um número fornecido em uma lista de strings onde cada
    componente é um item da string.

    Input:  numero (str) a ser transformado em uma lista de strings e invertido
    Return: lista_em_trios, uma lista de strings com ordem inversa ao número fornecido
    """
    #transforma 123456 em ['1', '2', '3', '4', '5', '6']
    lista_numero = [i for i in numero]
    #transforma ['1', '2', '3', '4', '5', '6'] em ['6', '5', '4', '3', '2', '1']
    lista_numero_invertida = list(reversed(lista_numero))
    lista_em_trios = list(separar_em_trios(lista_numero_invertida, 3))

    return lista_em_trios

def separar_em_trios(lista, n):
    """Esta função recebe uma lista e quebra a mesma em subsets de tamanho n
    Inputs: lista (list) a ser quebrada em subsets
            n     (int) tamanho do subset
    Return: um gerador, deve ser convertido em list para ser usado
    """
    #remove os pontos de separação que possam haver na string
    while '.' in lista: lista.remove('.')
    # transforma ['6', '5', '4', '3', '2', '1'] em [['6', '5', '4'], ['3', '2', '1']]
    for i in range(0, len(lista), n):
        yield (lista[i:i + n])

def inverter_lista(lista):
    """Esta função recebe uma lista de listas do número a ser invertido
```

```
    Input:  lista (list) no formato [['6', '5', '4'], ['3', '2', '1']]
    Return: numero_invertido (str) '654.321'
    """
    #converte [['6', '5', '4'], ['3', '2', '1']] em [['3', '2', '1'] ,['6', '5', '4']]
    lista_invertida = list(reversed(lista))
    #cria uma string vazia para contarmos o tamanho da nossa lista
    numero_invertido = ""
    #tarnsforma lista de listas em lista simples
    #[['3', '2', '1'] ,['6', '5', '4']] vira ['3', '2', '1' ,'6', '5', '4']
    lista_simples = sum(lista_invertida, [])
    #percorre a lista simples já na ordem correta da saída e monta uma string numero_invertido
    for i in lista_simples:
        numero_invertido += str(i)
    #formata o número invertido para ficar com as separações e facilitar leitura
    numero_invertido = formatar_numero(numero_invertido)
    return numero_invertido

def testar():
    """Esta função executa vários testes automatizados para confirmar funcionamento
    da função
    """
    assert inverter_posicoes(234567) == '432.765'
    assert inverter_posicoes(1) == '1'
    assert inverter_posicoes(12.0) == '21'
    assert inverter_posicoes(123.00) == '321'
    assert inverter_posicoes('1234') == '1.432'
    assert inverter_posicoes(12345) == '21.543'
    assert inverter_posicoes('123456.00') == '321.654'
    assert inverter_posicoes('1234567') == '1.432.765'
    assert inverter_posicoes(12345678.00) == '21.543.876'
    assert inverter_posicoes('123456789.0') == '321.654.987'
    assert inverter_posicoes(1234567890) == '1.432.765.098'
    print('Testes ok!')

def mostrar_testes():
    """Esta função retorna a impressão dos números enviados a função e a saída da função,
    para melhor visualização da operação da função
    """
    for i in [234567, 1, 12.0, 123.00, '1234', 12345, 123456, '1234567', 12345678.00, 123456789,
1234567890]:
        print(f'Número inserido:  {formatar_numero(i)}')
        print(f"Número invertido: {inverter_posicoes(i)}")
        print('----------------------------------------')

testar()
mostrar_testes()
```

Saída da função:

```
Testes ok!
Número inserido:  234.567
Número invertido: 432.765
----------------------------------------
Número inserido:  1
```

```
Número invertido: 1
----------------------------------------
Número inserido:  12
Número invertido: 21
----------------------------------------
Número inserido:  123
Número invertido: 321
----------------------------------------
Número inserido:  1.234
Número invertido: 1.432
----------------------------------------
Número inserido:  12.345
Número invertido: 21.543
----------------------------------------
Número inserido:  123.456
Número invertido: 321.654
----------------------------------------
Número inserido:  1.234.567
Número invertido: 1.432.765
----------------------------------------
Número inserido:  12.345.678
Número invertido: 21.543.876
----------------------------------------
Número inserido:  123.456.789
Número invertido: 321.654.987
----------------------------------------
Número inserido:  1.234.567.890
Número invertido: 1.432.765.098
----------------------------------------

Process finished with exit code 0
```

2. Imagine dois arquivos. O primeiro contém 5 milhões de linhas de ids, o segundo arquivo contém 4 milhões de linhas de ids. Gere um arquivo resultante (da forma mais performática que você conhecer) da diferença entre o arquivo A e o arquivo B.

Nota: a leitura do arquivo não é o mais importante. A lógica, a estrutura de dados e explicações do porque terão pesos maiores.

Exemplo:

Arquivo a:
KKKKKKKKKKKKKKKKKK
MM45ds4dssd5sd65dcrewd
BBBBBBBBBBBBBBBBBB
Hk45ds4dssZZsd65dcrewd

Arquivo b:
BBBBBBBBBBBBBBBBBB
Hk45ds4dsXXXsd65dcrewd
KKKKKKKKKKKKKKKKKK
Hk45ds4dssd5sdTT8crew

Output esperado do seu programa:
MM45ds4dssd5sd65dcrewd
Hk45ds4dssZZsd65dcrewd

OBS: Lembre de nos enviar o seu código

Resposta

Segue rotina que executa a operação proposta. Neste caso, matematicamente, temos dois conjuntos e desejamos realizar a operação de diferença entre eles, retornando o que existe no primeiro conjunto, mas não existe no segundo conjunto. O Python dispõe do tipo set para estas manipulações, porém, este tipo de lista não é ordenada. Como não foi dito se a informação de ordem de registro dos ids é relevante, optou-se por um método que mantém no arquivo de saída a ordem de aparição dos ids conforme ordem do arquivo a. Ao final também foi criado um teste com 2 arquivos com ids geradas para o teste.

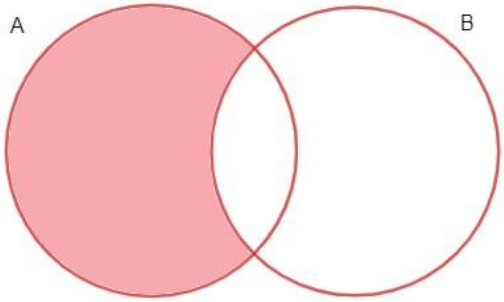


Figura 1: Operação de diferença de conjuntos A - B

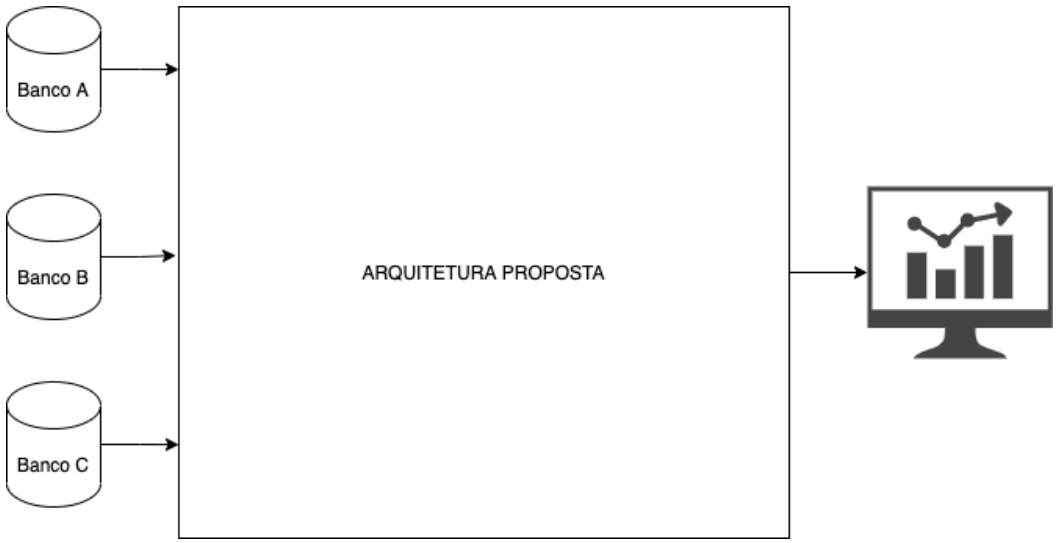
```
import time

def diferenciar_arquivos(dir_arquivo_a, dir_arquivo_b, dir_arquivo_resultado):
    """Este método recebe o caminho do diretório de 2 arquivos .csv, dir_arquivo_a
    e dir_arquivo_b, e o diretório do arquivo de saída dir_resultado. O método
    efetua a leitura dos dois arquivos e compara os conteúdos, retornando no
    arquivo de saída a diferença entre os dados do arquivo_a e o arquivo_b.

    Inputs:             dir_arquivo_a -> local onde está o arquivo a
                        dir_arquivo_a -> local onde está o arquivo b
                dir_arquivo_resultado -> local onde será salvo o arquivo resultado
    Outputs:              um arquivo -> arquivo_resultado.csv
    """

    # Parte 1: Leitura dos dois arquivos e armazenamento dos dados em uma lista

    lista_a = []
    arquivo_a = open(dir_arquivo_a, "r")
    lista_a = arquivo_a.readlines()
    arquivo_a.close()

    lista_b = []
    arquivo_b = open(dir_arquivo_b, "r")
    lista_b = arquivo_b.readlines()
    arquivo_b.close()

    # Parte 2: Converte a lista 2 em um conjunto (set), a fim de usar
    # um loop para checar se os dados da lista_a estão dentro deste set

    lista_b_set = set(lista_b)

    # Usando List Comprehensions vamos criar uma lista com os valores da lista_a
    # que não estejam contidos no set lista_b_set
```

```
    lista_resultado = [x for x in lista_a if x not in lista_b_set]

    # Também poderíamos transformar as duas listas em sets e fazer set(a) - set(b)
    # porém, neste caso iríamos perder a informação da ordem de aparição na lista_a
    # pois os sets são listas desordenadas

    # Parte 3: Criar o arquivo resultado e adicionar os dados da lista_resultado

    tamanho_lista_resultado = range(len(lista_resultado))

    arquivo_resultado = open(dir_arquivo_resultado, "w+")
    for linha in tamanho_lista_resultado:
        arquivo_resultado.write(lista_resultado[linha])
    arquivo_resultado.close()

""" Vamos testar com 2 arquivos de exemplo:

            arquivo_a.csv tem 1.048.576 ids
            arquivo_b.csv tem 800.000 ids
As ids do arquivo_b.csv são as mesmas 800.000 primeiras ids do arquivo_a.csv.
Dessa forma, o arquivo de saída terá 1.048.576 - 800.000 = 248.576 ids.
Vamos contar o tempo de execução para ter uma ideia da performance do método.
Iremos executar o método 5 vezes em um loop para simular uma quantidade de ids
acima de 5 milhões, conforme enunciado do desafio.
"""
dir_arquivo_a = "arquivo_a.csv"
dir_arquivo_b = "arquivo_b.csv"
dir_arquivo_resultado = "arquivo_resultado.csv"

start_time = time.time()
for k in range(5):
    diferenciar_arquivos(dir_arquivo_a, dir_arquivo_b, dir_arquivo_resultado)
print(f'{time.time() - start_time} segundos de execução para este método.')
```

Parte 3 - Arquitetura (não necessário, é um diferencial)

Queremos que você fale um pouco sobre a organização da arquitetura necessária para atender uma necessidade de negócio. Vamos falar de um caso de uso e gostaríamos que propusesse uma solução envolvendo as ferramentas com que tem familiaridade. Sinta-se à vontade para explicar de maneira descritiva, com desenhos ou qualquer forma que preferir, o importante mesmo é explicar quais são suas ideias.
Suponha que você tem três bancos de dados transacionais e tradicionais que serão sua fonte de dados. Eles possuem o mesmo schema, mas são de organizações diferentes.
Seu trabalho é desenhar uma arquitetura que permita a união contínua e atualizada dessas bases em uma nova fonte de dados. Essa fonte de dados única será a entrada para dashboards da área de negócio.
Sinta-se à vontade para usar qualquer ferramenta, cloud provider, ou qualquer coisa que você julgar necessária.

O desenho abaixo ilustra o problema:



A entrega dessa etapa é o desenho da arquitetura com a explicação dos itens utilizados no pipeline.

OBS: qualquer solução é bem-vinda se estiver corretamente embasada e explicada.

Resposta:

Como os bancos de dados tem o mesmo schema, uma opção seria mover dados de cada um dos três bancos para um banco de dados central, a fim de disponibilizar os dados para as aplicações. Porém, esta solução seria difícil de escalar e requer muitos processos manuais de integração local, uma vez que é preciso mover os dados constantemente para se ter uma base unificada atualizada.

Uma opção melhor para este problema seria a integração dos dados dos 3 bancos de dados por meio de um serviço de nuvem, com a criação de um repositório centralizado, ou data warehouse. No data warehouse, os dados históricos dos bancos de dados ficam disponíveis e podem ser utilizados para criação de relatórios e análise de dados.

Para tanto, os dados dos bancos de dados seriam periodicamente enviados para a nuvem. Para serem enviados, os dados passam pelo processo de Extração, Transformação e Carregamento (ETL) para serem formatados e reorganizados. Após, são carregados no data warehouse e podem ser acessados por outras interfaces que utilizam estes dados, como ferramentas de BI e analytics. Há diversas ferramentas disponíveis para implementar esta arquitetura, como por exemplo:

- Azure Synapse Analytics, solução completa da Azure que integra desde a ingestão de dados pelos bancos de dado SQL até a visualização com Power BI;
- Amazon Redshift, solução para criação de data warehouse da Amazon com integração com S3;

- Google BigQuery, solução para criação de data warehouse do Google com suporte para consultas SQL e ferramentas para aprendizado de máquina, com uso do Data Fusion para criar o pipeline de dados;

Esta abordagem oferece diversos benefícios:

- Suporta uma grande quantidade de dados, pois é focado em big data;
- Permite escalabilidade e adição de novos bancos de dados para integração;
- Aumenta a segurança, pois os dados ficam em servidores remotos;
- Aumenta a disponibilidade dos dados, pois todas as áreas que necessitem analisar os dados têm acesso, o que melhora o processo de tomada de decisão;

Exemplo de arquitetura utilizando serviços Google Cloud