Teste QA Técnico

Analista de Qualidade e Testes responsável: Cristian Fior Cavalheiro

SQL

1. JOINs: Explique a diferença entre INNER JOIN, LEFT JOIN e RIGHT JOIN. Dê um exemplo prático.

Tendo por base as tabelas:

clientes

id	nome_cliente	id_profissao
1	Pafuncio	3
2	Fulano	1
3	Sicrano	9

profissoes

id	nome_profissao
1	QA
2	Dev
3	DBA

INNER JOIN: a consulta retorna SOMENTE os dados que se relacionam mutuamente entre duas tabelas.

Exemplo: SELECT * FROM clientes INNER JOIN profissoes ON clientes.id_profissao = profissoes.id;

A tabela resultante da consulta ficaria:

id	nome_cliente	id_profissao	id	nome_profissao
1	Pafuncio	3	3	DBA
2	Fulano	1	1	QA

Perceba que somente os dados estritamente iguais serão devolvidos por esta consulta. O registro do cliente de id = 3 não é exibido pois seu id_profissao não possui equivalente na tabela profissoes.

LEFT JOIN: a consulta retorna todos os dados existentes na tabela à esquerda, a primeira tabela mencionada na consulta, mesmo quando os dados correspondentes na tabela à direita não existam, e nesse caso retorna valor NULL.

Exemplo: SELECT * FROM clientes LEFT JOIN profissoes ON clientes.id_profissao = profissoes.id;
A tabela resultante da consulta ficaria:

id	nome_cliente	id_profissao	id	nome_profissao
1	Pafuncio	3	3	DBA
2	Fulano	1	1	QA
3	Sicrano	9	NULL	NULL

Perceba que a tabela CLIENTES está à esquerda do LEFT JOIN e portanto a consulta respeitará essa tabela. Os dados da tabela profissoes para o terceiro registro da tabela trazem os valores nulo, pois o id_profissao utilizado no registro do cliente de id = 3 não está cadastrado na tabela profissoes.

RIGHT JOIN: é o exato oposto da LEFT, a consulta retorna todos os dados existentes na tabela à direita, a segunda tabela mencionada, mesmo quando os dados correspondentes na tabela à esquerda não existam, e nesse caso retorna valor NULL.
Exemplo: SELECT * FROM clientes RIGHT JOIN profissoes ON clientes.id_profissao = profissoes.id;
A tabela resultante da consulta ficaria:

id	nome_cliente	id_profissao	id	nome_profissao
1	Pafuncio	3	3	DBA
2	Fulano	1	1	QA
NULL	NULL	NULL	2	Dev

Perceba que a tabela PROFISSOES está à direita do RIGHT JOIN e portanto a consulta respeitará essa tabela. Os dados da tabela clientes para o terceiro registro da tabela trazem os valores nulo, pois nenhum dos registros utiliza o id da profissao = 2.

2. NOLOCK: O que faz o hint WITH (NOLOCK)? Cite uma vantagem e um risco de usá-lo.
Permite a leitura de dados sem esperar por bloqueios de outras transações, por isso "NOLOCK". Um exemplo disso é quando alguém está rodando um UPDATE de grande dimensão e ao mesmo momento você precisa consultar dados daquela tabela sendo atualizada, com esse hint é possível realizar a consulta sem ter que esperar que a atualização de dados seja finalizada.
Na própria explicação encontra-se uma vantagem e uma desvantagem; a vantagem de poder ler dados ainda não comitados, além da performance pois ignora qualquer tipo de bloqueio imposto nas tabelas, e a desvantagem é que a consulta pode vir com dados sujos, como é o caso do cenário onde a consulta ser feita antes do update finalizar traria dados inconsistentes com o pós update e poderiam conflitar eventualmente em alguma auditoria, por exemplo.

3. Linked Server: O que é e como consultar uma tabela em outro servidor usando um linked server chamado SrvERP?
É um objeto de conexão existente no SQL Server que permite acessar dados que se encontram em outros bancos ou em outros SGBDS, com isso é possível realizar comandos DQL e DML.
Utilizando do exemplo das tabelas clientes e profissoes, posto que a tabela profissoes se encontra no linked server SrvERP, poderíamos ter consultas como: SELECT * FROM SrvERP.Clientes.dbo.profissoes;
ou fazer interações entre ambas tabelas com consultas como: SELECT * FROM clientes INNER JOIN SrvERP.Clientes.dbo.profissoes ON clientes.id_profissao = SrvERP.Clientes.dbo.profissoes.id

4. Funções de agregação: Qual a diferença entre COUNT(*), COUNT(coluna), SUM() e AVG()? Dê um exemplo.

- A função COUNT(*) retorna com o valor do total de registros com valores ou nulos, existentes na tabela informada. Por exemplo a consulta: SELECT COUNT(*) FROM clientes; retornaria um COUNT igual a 3, pois possui 3 registros cadastrados nessa tabela.
- Já a função COUNT(coluna) por sua vez não faz a somatória com registros nulos. Então caso na tabela existisse um quarto registro com nome nulo, no COUNT(nome_cliente) retornaria somente o valor 3, enquanto que no COUNT(*) retornaria 4.
- A função SUM() realiza a soma de todos os valores da coluna informada. Na consulta: SELECT SUM(id) FROM clientes; retornaria o valor 6, pois temos os IDs 1, 2 e 3.
- A função AVG() faz o calculo da média da coluna, somando os valores e dividindo pela quantidade de registros. Por exemplo, a consulta: SELECT AVG(id) FROM clientes; retornaria o valor 2, pois temos os IDs 1, 2 e 3 que somam 6, dividindo pelo total de registros que é 3 temos o resultado 2.

5. Filtragem e ordenação: Escreva um comando SQL que traga os 10 clientes mais recentes de São Paulo, ordenados do mais novo para o mais antigo.
SELECT * FROM clientes WHERE cidade = "São Paulo" ORDER BY id DESC limit = 10;
Em SQL Server: SELECT TOP 10 * FROM clientes WHERE cidade = "São Paulo" ORDER BY id DESC;
Utilizando o parametro DESC da função ORDER BY criamos uma tabela decrescente, do 10° ao 1° id da consulta, já que o registro de maior número foi incluído mais recentemente

6. Performance: Cite duas boas práticas para melhorar o desempenho de queries e uma forma de identificar gargalos.
Em comandos DQL sempre especificar as colunas desejadas, dessa forma o tempo de consulta diminui pois o banco tem as instruções exatas do que trazer e não perde tempo trazendo dados indesejados ou inúteis. Outra boa prática é a não utilização de subconsultas repetitivas, especialmente quando JOINS podem resolver o problema.
Uma boa forma para identificar gargalos é a analise do plano de execução pois nele consta todo o esquema de arquitetura da consulta, mostrando passo a passo como ela foi executada.

7. Bugs e comunicação: Você encontrou um bug em produção que impede o envio de mensagens e o dev diz 'na minha máquina funciona'. O que faz?
Realizo a validação mais de uma vez e gero evidências com imagens e vídeos, se possível realizo também um par com o desenvolvedor para mostrar ao vivo que o problema está realmente acontecendo e como reproduzir. Feito o evidenciamento, abro o card de bug e altero sua severidade e prioridade de correção, se tratando de uma função essencial que afeta diretamente o usuário e o impede de utilizar o sistema, categorizo como severidade crítica e prioridade altíssima para ser corrigida e lançada como hotfix.

API

Seguindo a documentação da API https://reqres.in/, responda as seguintes questões: Api Key: reqres-free-v1

1. Crie o registro do seguinte usuário: charles.morris@reqres.in. Apresente o endpoint utilizado e o retorno da chamada.
Foi realizada uma requisição no método POST para o endpoint: *https://reqres.in/api/register* passando os dados em formato JSON no corpo da requisição: {“email”: “charles.morris@reqres.in”, “password”: “Senha123”}
Como resposta dessa requisição obtive o status code *200 OK* e o JSON: {“id”: “5”, “token”: “QpwL5tke4Pnpja7X5”}

2. Liste o email, primeiro nome, último nome e link do avatar de 12 usuários existentes nesta API. Apresente o endpoint utilizado e o retorno da chamada.
Foi realizada uma requisição no método GET para o endpoint: *https://reqres.in/api/users?per_page=12*, onde *per_page=12* é o filtro para trazer somente os 12 primeiros usuários existentes.
Como resposta dessa requisição obtive o status code *200 OK* e o JSON:


```
{
   "page": 1,
   "per_page": 12,
   "total": 12,
   "total_pages": 1,
   "data": [
      {
         "id": 1,
         "email": "george.bluth@reqres.in",
         "first_name": "George",
         "last_name": "Bluth",
         "avatar": "https://reqres.in/img/faces/1-image.jpg"
      },
      {
         "id": 2,
         "email": "janet.weaver@reqres.in",
```

```
    "first_name": "Janet",
    "last_name": "Weaver",
    "avatar": "https://reqres.in/img/faces/2-image.jpg"
},
{
    "id": 3,
    "email": "emma.wong@reqres.in",
    "first_name": "Emma",
    "last_name": "Wong",
    "avatar": "https://reqres.in/img/faces/3-image.jpg"
},
{
    "id": 4,
    "email": "eve.holt@reqres.in",
    "first_name": "Eve",
    "last_name": "Holt",
    "avatar": "https://reqres.in/img/faces/4-image.jpg"
},
{
    "id": 5,
    "email": "charles.morris@reqres.in",
    "first_name": "Charles",
    "last_name": "Morris",
    "avatar": "https://reqres.in/img/faces/5-image.jpg"
},
{
    "id": 6,
    "email": "tracey.ramos@reqres.in",
    "first_name": "Tracey",
    "last_name": "Ramos",
    "avatar": "https://reqres.in/img/faces/6-image.jpg"
},
{
    "id": 7,
    "email": "michael.lawson@reqres.in",
    "first_name": "Michael",
    "last_name": "Lawson",
    "avatar": "https://reqres.in/img/faces/7-image.jpg"
},
{
    "id": 8,
    "email": "lindsay.ferguson@reqres.in",
    "first_name": "Lindsay",
    "last_name": "Ferguson",
```

```
            "avatar": "https://reqres.in/img/faces/8-image.jpg"
        },
        {
            "id": 9,
            "email": "tobias.funke@reqres.in",
            "first_name": "Tobias",
            "last_name": "Funke",
            "avatar": "https://reqres.in/img/faces/9-image.jpg"
        },
        {
            "id": 10,
            "email": "byron.fields@reqres.in",
            "first_name": "Byron",
            "last_name": "Fields",
            "avatar": "https://reqres.in/img/faces/10-image.jpg"
        },
        {
            "id": 11,
            "email": "george.edwards@reqres.in",
            "first_name": "George",
            "last_name": "Edwards",
            "avatar": "https://reqres.in/img/faces/11-image.jpg"
        },
        {
            "id": 12,
            "email": "rachel.howell@reqres.in",
            "first_name": "Rachel",
            "last_name": "Howell",
            "avatar": "https://reqres.in/img/faces/12-image.jpg"
        }
    ],
    "support": {
        "url":
"https://contentcaddy.io?utm_source=reqres&utm_medium=json&utm_campaign=referral",
        "text": "Tired of writing endless social media content? Let Content Caddy generate it for
you."
    },
    "_meta": {
        "powered_by": "🚀 ReqRes - Deploy backends in 30 seconds",
        "upgrade_url": "https://app.reqres.in/upgrade",
        "docs_url": "https://reqres.in",
        "template_gallery": "https://app.reqres.in/templates",
        "message": "This API is powered by ReqRes. Deploy your own backend in 30 seconds!",
        "features": [
```

```
            "30 Second Backend Templates",
            "Custom API Endpoints",
            "Data Persistence",
            "Real-time Analytics"
        ],
        "upgrade_cta": "Upgrade to Pro for unlimited requests, custom endpoints, and data
persistence"
    }
}
```


3. Remova o usuário de ID = 11. Apresente o endpoint utilizado e o retorno da chamada.
Foi realizada uma requisição no método DELETE para o endpoint *https://reqres.in/api/users/11*, onde o parametro *11* é o id do usuário a ser removido.
Como resposta dessa requisição obtive o status code *204 No Content* sem JSON pois se trata de uma deleção.

4. Atualize o nome do usuário de ID = 10 para 'BrasilCard'. Apresente o endpoint utilizado e o retorno da chamada.
Foi realizada uma requisição no método PATCH para o endpoint *https://reqres.in/api/users/10*, passando os dados em formato JSON no corpo da requisição: {“username”:”BrasilCard”}.
O parametro *10* é o id do usuário a ser alterado. O método utilizado foi PATCH ao invés de PUT, pois só houve necessidade de alterar o campo username.
Como resposta dessa requisição obtive o status code *200 OK* com o JSON: { "username": "BrasilCard", "updatedAt": "2025-11-09T15:07:41.038Z" }