Administração de Bases de Dados

Engenharia Informática – Universidade do Minho

Trabalho Prático – 2024/2025

Os grupos de trabalho devem ser constituídos por 4 ou 5 elementos, todos inscritos na Unidade Curricular. O resultado do trabalho é um relatório escrito e código/scripts/resultados criados por cada grupo. O relatório deve omitir considerações genéricas sobre as ferramentas utilizadas, focando a apresentação e justificação dos objetivos atingidos. O desenvolvimento e entrega do trabalho é realizado em repositórios *git* disponibilizados pela equipa docente. O relatório deve identificar também o nome e número de todos os elementos do grupo. A data limite é 2025/05/13.

1 Contexto

O trabalho prático consiste na configuração, otimização, e avaliação de um *benchmark* disponível em cada repositório, baseado num pequeno excerto do dataset da loja online *Steam* e contendo operações transacionais e analíticas.

A componente transacional simula operações de *backend* da *Steam*, considerando os seguintes procedimentos:

- **AddPlaytime** – incrementar a estatística de tempo jogado a um par utilizador-jogo.
- **ReviewGame** – adicionar uma nova crítica a um jogo.
- **BuyGame** – adicionar um jogo à biblioteca de um utilizador.
- **NewFriendship** – criar uma amizade entre dois utilizadores.
- **NewGame** – registar um novo jogo.
- **NewUser** – registar um novo utilizador.
- **GameInfo** – consultar a informação de um jogo.
- **GameReviews** – consultar o score e recomendações recentes de um jogo.
- **UserInfo** – consultar a informação de um utilizador.
- **RecentGamesPerTag** – consultar os jogos mais recentes com uma dada *tag*.
- **SearchGames** – consultar jogos usando pesquisa exata ou por similaridade.

As cinco interrogações analíticas simulam operações estatísticas e de *data warehousing* sobre o dataset, definindo os seguintes argumentos:

- **Q1** – *n/a*.
- **Q2** – identificador numérico de utilizador (valor predefinido = 3331100).
- **Q3** – prefixo de estúdio/empresa (valor predefinido = Ubisoft).

- **Q4** – data inferior (valor predefinido = 2020-01-01) e tamanho do intervalo (valor predefinido = 12 hours)
- **Q5** – identificador numérico de jogo (valor predefinido = 730).

Apesar dos argumentos estarem definidos de forma estática, deve-se considerar que podem ser modificados para **valores arbitrários**. Para além disso, é pretendido que as interrogações obtenham resultados sobre os **dados mais recentes**, sendo que não é viável a sua completa materialização.

2 Objetivos

A configuração de referência é uma máquina virtual na *Google Cloud* do tipo N2, 8 vCPUs, 16 GB RAM, disco Permanente SSD 500 GB[1], e sistema operativo Linux Ubuntu 24.04 LTS x86/64. A carga transacional deve considerar 16 clientes, enquanto que a carga analítica deve considerar apenas 1. Usando esta configuração, devem:

1. Otimizar o desempenho da carga transacional.
2. Otimizar o desempenho das interrogações analíticas.

A realização destes objetivos deve considerar **redundância** (e.g., índices, materialização), **paralelismo**, **parâmetros de configuração**, e até **código SQL/Java**.

Importante: Para a componente transacional, devem primeiro otimizar através da redundância e/ou código e só depois considerarem a otimização através dos parâmetros de configuração, de modo a obterem diferenças mais significativas neste último ponto.

3 Código e Dados

O código que implementa as operações descritas, bem como scripts de auxílio para carregar os dados, estão disponíveis em cada repositório. O dataset com os dados, em formato CSV, é disponibilizado em `https://storage.googleapis.com/abd25/data.zip`. Não coloquem o dataset no repositório!

4 Notas

- Como medidas de desempenho da carga transacional deve considerar-se principalmente o **débito máximo** atingível. Nas operações analíticas deve considerar-se o **tempo de resposta** médio de várias execuções.
- Poderão modificar o código SQL/Java e/ou o esquema da base de dados. Devem explicar no relatório em que medida essas alterações preservam o funcionamento da aplicação original.
- Devem considerar que cada *workload* corre separadamente, não sendo necessário executar a carga transacional e analítica de forma concorrente. Contudo, devem verificar que impacto a criação de redundância / alteração do esquema num *workload* tem no outro.
- Em cada um dos objetivos, poderão não ter tempo para considerar todas as otimizações possíveis nas suas várias combinações. Devem focar-se nas que consideram mais prometedoras e que mais vos interessam, justificando no relatório essas opções.

[1] Ao escolher o disco poderá ser necessário selecionar "Request quota adjustment" na altura da criação da máquina.

- No caso de não obterem melhorias de desempenho, devem explicar porque é que a configuração de referência já era ótima. Por outro lado, a simples apresentação de uma melhoria de desempenho, não justificada, não é muito interessante.
- A utilização de ferramentas de monitorização e diagnóstico do PostgreSQL e do sistema operativo (e.g., pgbadger, psutil, iostat) valoriza o trabalho.
- Devem considerar uma instalação nativa para o PostgreSQL (i.e., sem Docker).
- A **automatização** da instalação, execução, e obtenção de dados do *benchmark* permitirá obter resultados em maior quantidade e uma análise mais profunda, valorizando o trabalho.
- Devem também procurar estratégias para poupar recursos na *Google Cloud*, por exemplo, armazenando os dados em *Cloud Storage* e reutilizando-os[2], em vez de re-executar o `load.py`, e/ou explorando o uso de *Machine Images*. Considerem as opções mais baratas na *Google Cloud* (i.e., regiões nos EUA, instâncias *spot*, ...) e de **não deixar máquinas virtuais ativas a consumir recursos desnecessariamente!**

[2] Ver `pg_dump` (https://www.postgresql.org/docs/17/app-pgdump.html) e `pg_restore` (https://www.postgresql.org/docs/17/app-pgrestore.html).