

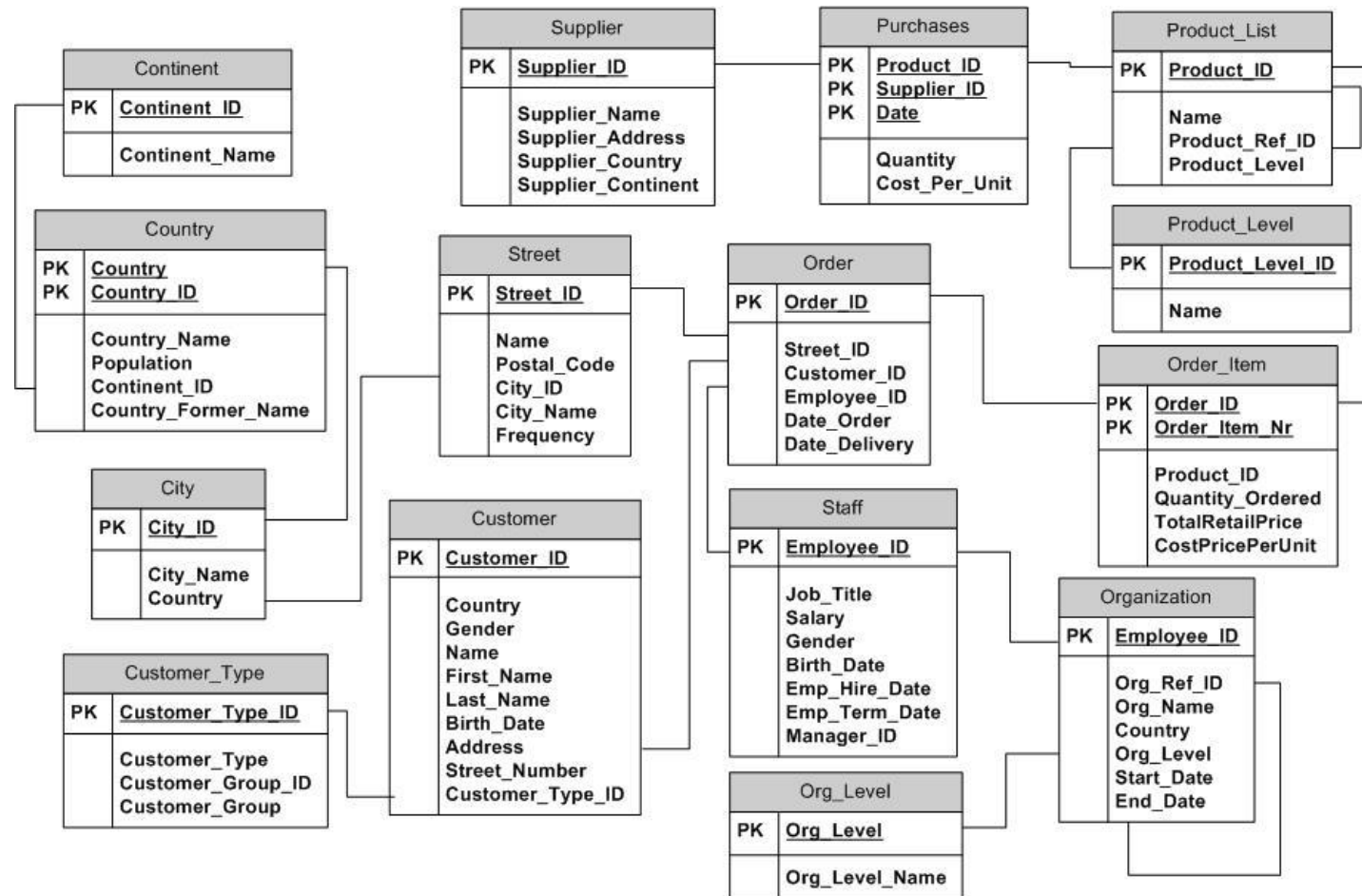


# ORIENTAÇÃO A OBJETOS 2

*Bancos relacionais, objeto-relacionais  
e não relacionais*

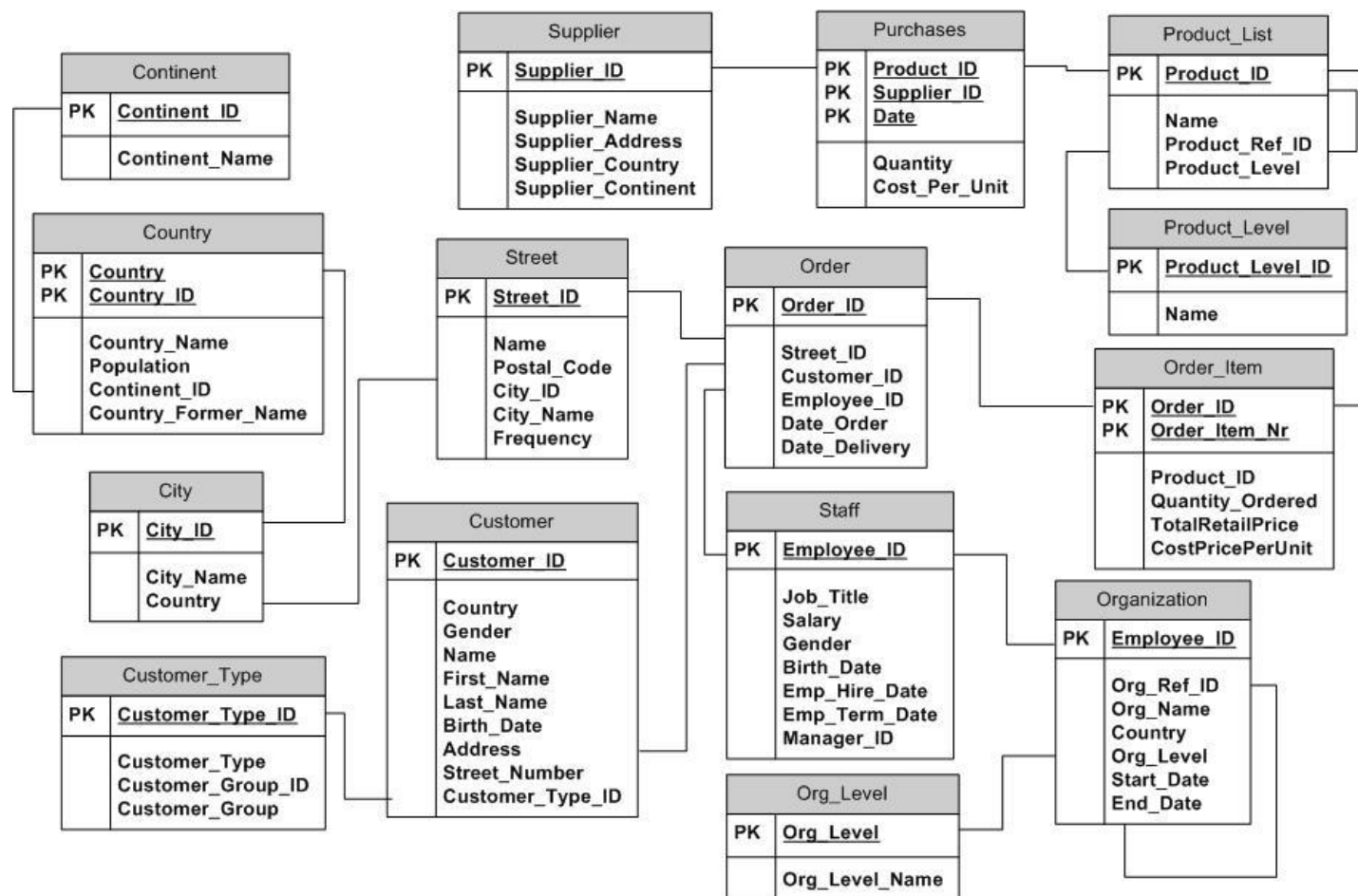
# BANCO DE DADOS RELACIONAL

Que diagrama é esse?



# BANCO DE DADOS RELACIONAL

Que diagrama é esse?



Um diagrama relacional!



# BANCO DE DADOS RELACIONAL

- Definido por E.F. Codd em 1970, uma base de dados **relacional** é uma base de dados digital construída em cima do modelo relacional de informação.
  - Os dados são guardados em tabelas contendo linhas ou tuplas (que representam uma entidade/instância) e colunas (que representam uma característica, atributo, feature, propriedade, etc, das entidades/instâncias).
- As relações são estabelecidas por meio de chaves primárias e chaves estrangeiras.
  - O que são **chaves primárias e estrangeiras**?



# BANCO DE DADOS RELACIONAL

- O diagrama anterior é um diagrama **relacional**.
- Nesse sentido, o que são bancos de dados relacionais?



# BANCO DE DADOS RELACIONAL

- O diagrama anterior é um diagrama **relacional**.
- Nesse sentido, o que são bancos de dados relacionais?
  - São bancos que se estruturam **de forma relacional**, como na imagem do diagrama.
  - São bancos que **normalmente são controlados pela linguagem SQL**, que é uma linguagem relacional.

# EXEMPLO DE BANCO DE DADOS RELACIONAL

The screenshot displays the phpMyAdmin web interface. On the left, a sidebar shows the database hierarchy: 'custo246' > 'custo246\_pres848' > 'ps\_access'. The main panel shows the 'ps\_access' table structure and data. The table has 7 columns: 'id\_profile', 'id\_tab', 'view', 'add', 'edit', and 'delete'. The data is sorted by 'id\_profile' and shows 7 rows.

Server: localhost » Database: custo246\_pres848 » Table: ps\_access

Showing rows 0 - 24 (416 total, Query took 0.0029 sec)

`SELECT * FROM `ps_access``

1 Show all > >> Number of rows: 25

Sort by key: None

+ Options

			id_profile	id_tab	view	add	edit	delete
<input type="checkbox"/>	Edit	Copy	Delete	1	0	1	1	1
<input type="checkbox"/>	Edit	Copy	Delete	1	1	1	1	1
<input type="checkbox"/>	Edit	Copy	Delete	1	2	0	0	0
<input type="checkbox"/>	Edit	Copy	Delete	1	3	0	0	0
<input type="checkbox"/>	Edit	Copy	Delete	1	4	0	0	0
<input type="checkbox"/>	Edit	Copy	Delete	1	5	1	1	1
<input type="checkbox"/>	Edit	Copy	Delete	1	6	0	0	0

[https://secure147.inmotionhosting.com:2083/cpsess1093019108/3rdparty/phpMyAdmin/sql.php?server=1&db=custo246\\_pres848&table=ps\\_access&pos=0&token=ce253f91c7c4317acb484d658304f6b2](https://secure147.inmotionhosting.com:2083/cpsess1093019108/3rdparty/phpMyAdmin/sql.php?server=1&db=custo246_pres848&table=ps_access&pos=0&token=ce253f91c7c4317acb484d658304f6b2)



# BANCO DE DADOS RELACIONAL


- O que são banco de dados/ferramentas **objeto-relacional** e **orientado a objetos**?



# BANCO DE DADOS OBJETO-RELACIONAL

- Como o nome já diz, existe uma **conversão** de objeto para uma linguagem relacional ou diretamente para uma base relacional.

```
public class Pessoa{  
    int id;  
    String nome;  
    int idade;  
}
```



Nome da Coluna	Tipo de dado
id	int
nome	varchar (30)
Idade	int



# BANCO DE DADOS OBJETO-RELACIONAL

- Como o nome já diz, existe uma **conversão** de objeto para uma linguagem relacional ou diretamente para uma base relacional.
  - Em outras palavras, existe um mecanismo que converte o conteúdo do seu objeto para uma linguagem relacional (SQL, por exemplo), e persiste os dados do objeto diretamente no banco de dados.
- No geral:
  - A estrutura da sua classe vira SQL (ou algo tipo).
  - As propriedades dos objetos também viram SQL.



# BANCO DE DADOS NÃO RELACIONAL (NoSQL)

- Começou como “no SQL” e depois se transformou em “Not Only SQL”.
- A terminologia “**Not Only SQL**” surgiu para enfatizar que esses tipos de banco de dados **podem** suportar linguagens como SQL, porém a sua estrutura interna de armazenamento de dados **não é relacional**.

# BANCO DE DADOS NÃO RELACIONAL (NoSQL)

- Exemplo:
  - Bancos de dados de **arquivos** (ligar uma chave ou identificador a um arquivo).
  - Banco de dados de imagens
    - Imagens possuem características que normalmente não se conformam com as **características relacionais** dos bancos. Exemplos:
      - Brilho
      - Tamanho
      - Cor predominante
      - Contraste
      - Etc
  - Banco de dados de videos (mesmo sentido do exemplo anterior). Vídeos são conjuntos de imagens sequenciais.

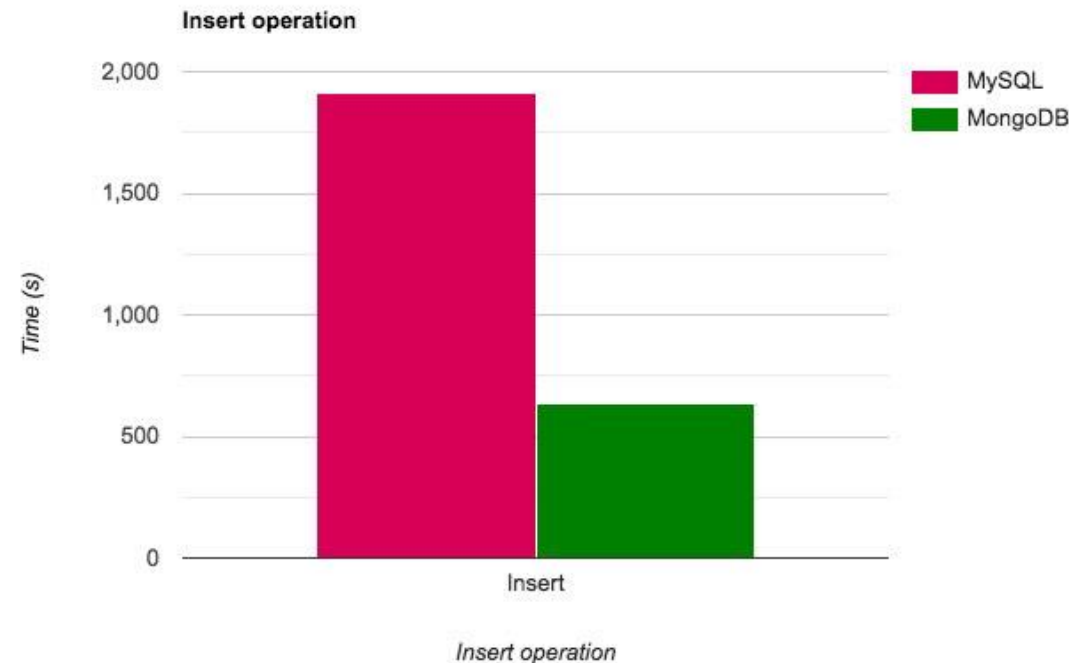


# POS VS CONS

- Quais seriam as vantagens e desvantagens de bancos relacionais x não relacionais?

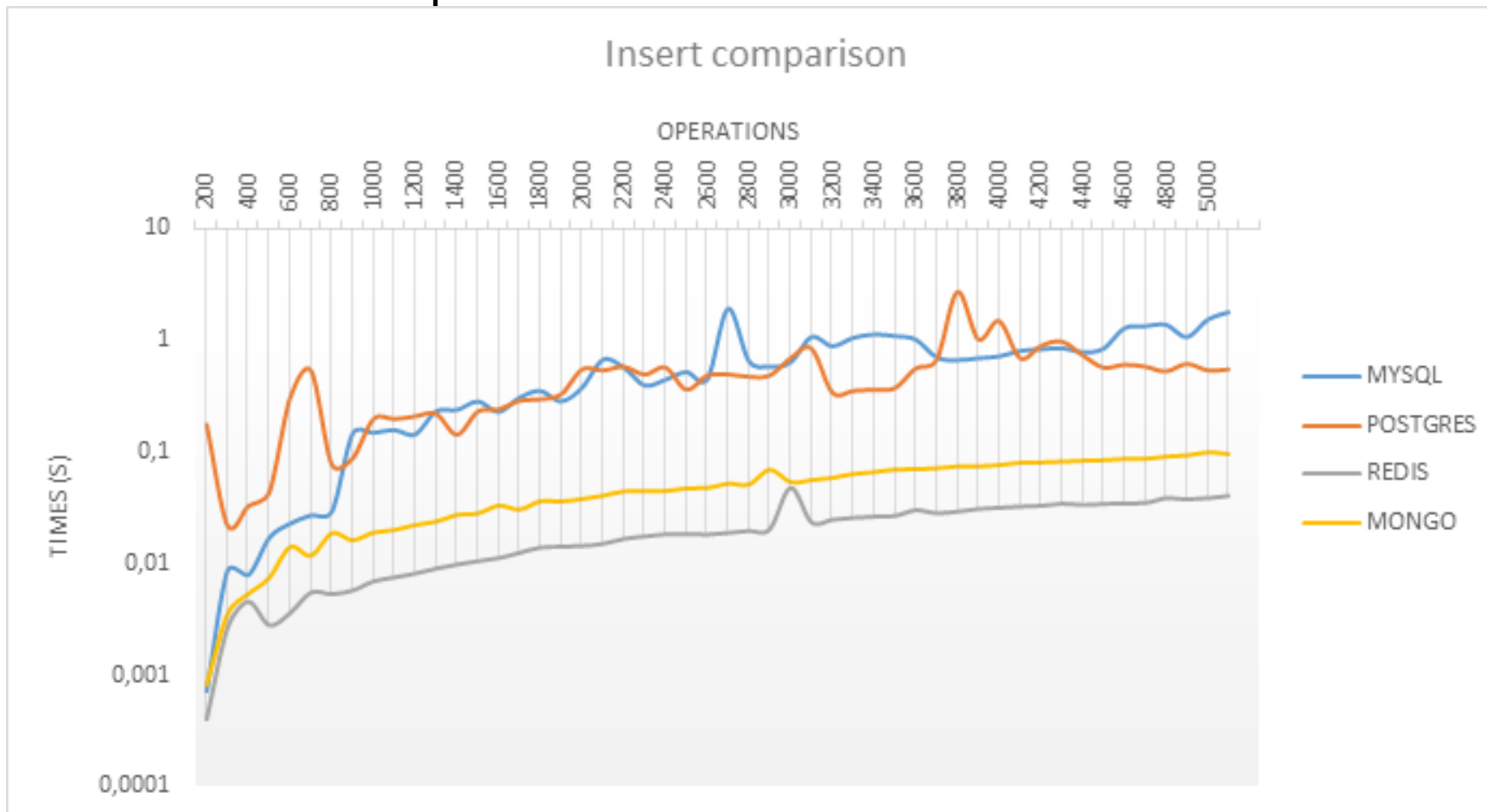
# MONGODB

- MongoDB é um banco de dados de propósito geral baseado em documento. Esse documento é uma **estrutura de dados composta de campos e valores em par** (como um HashMap/HashTable).

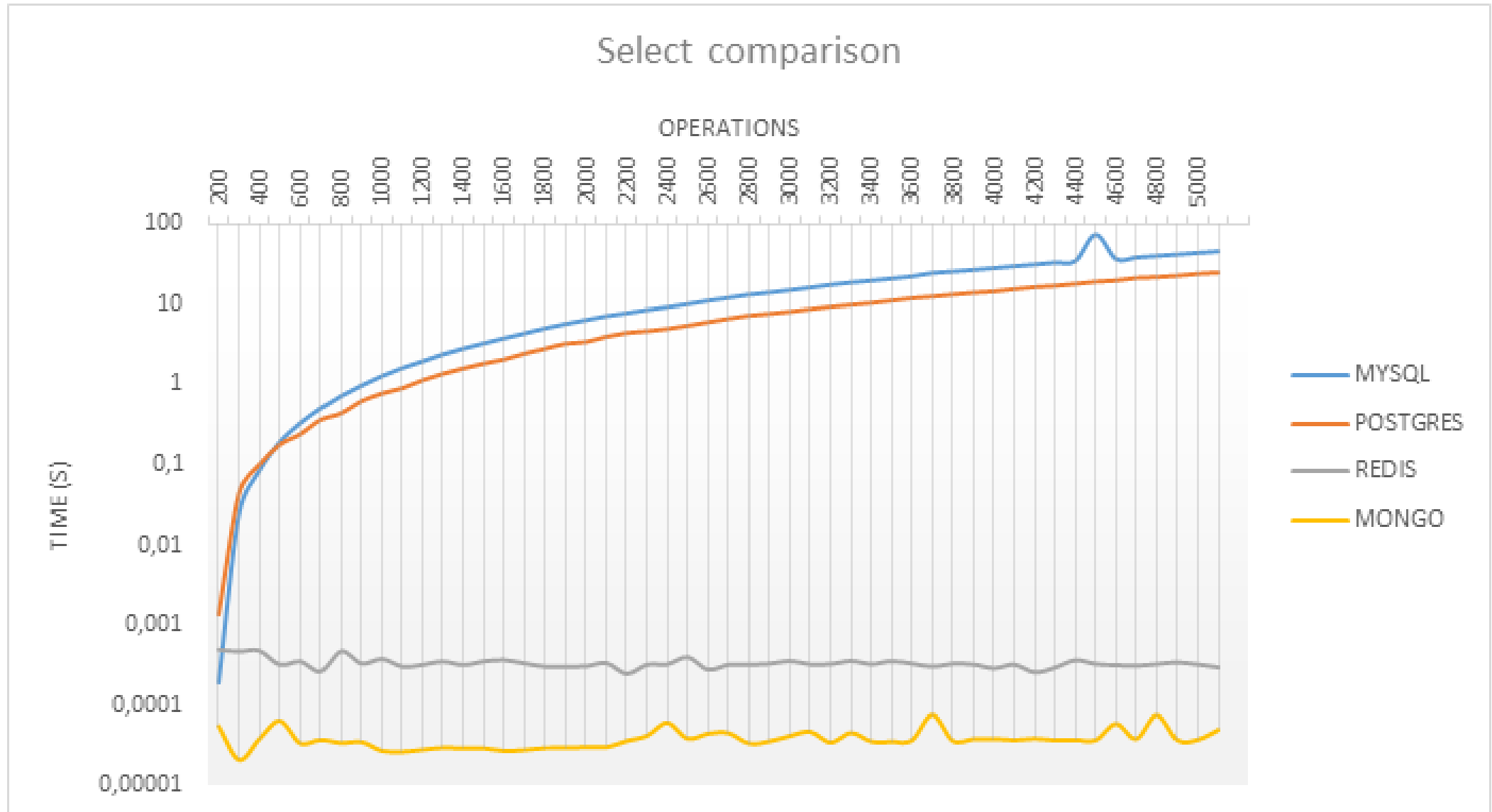


# CRUD: REDIS e MONGO - INSERT

- Redis é um banco de dados open-source NoSQL. Usado no GitHub, Pinterest e Snapchat.

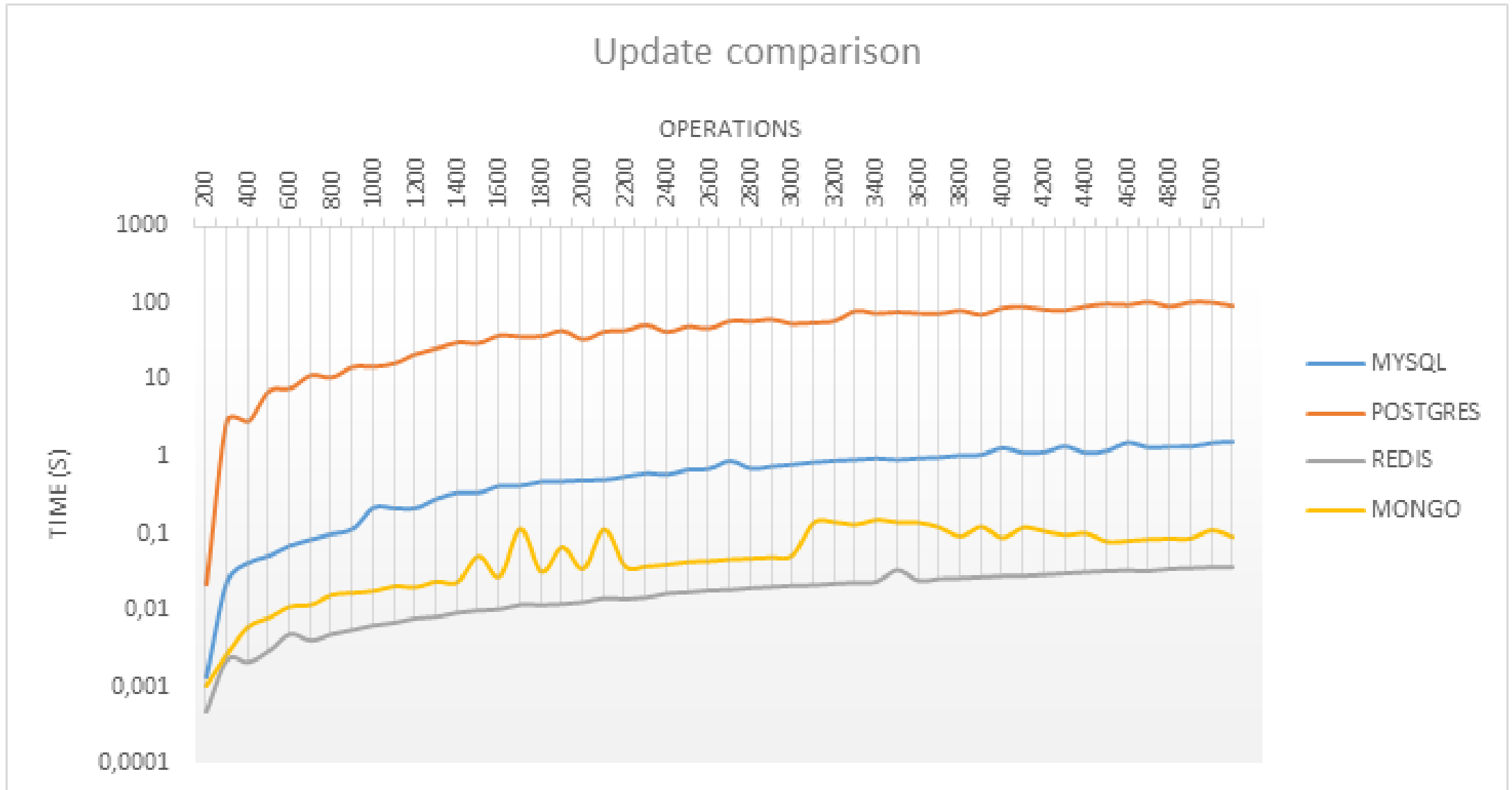


# CRUD: REDIS e MONGO - SELECT

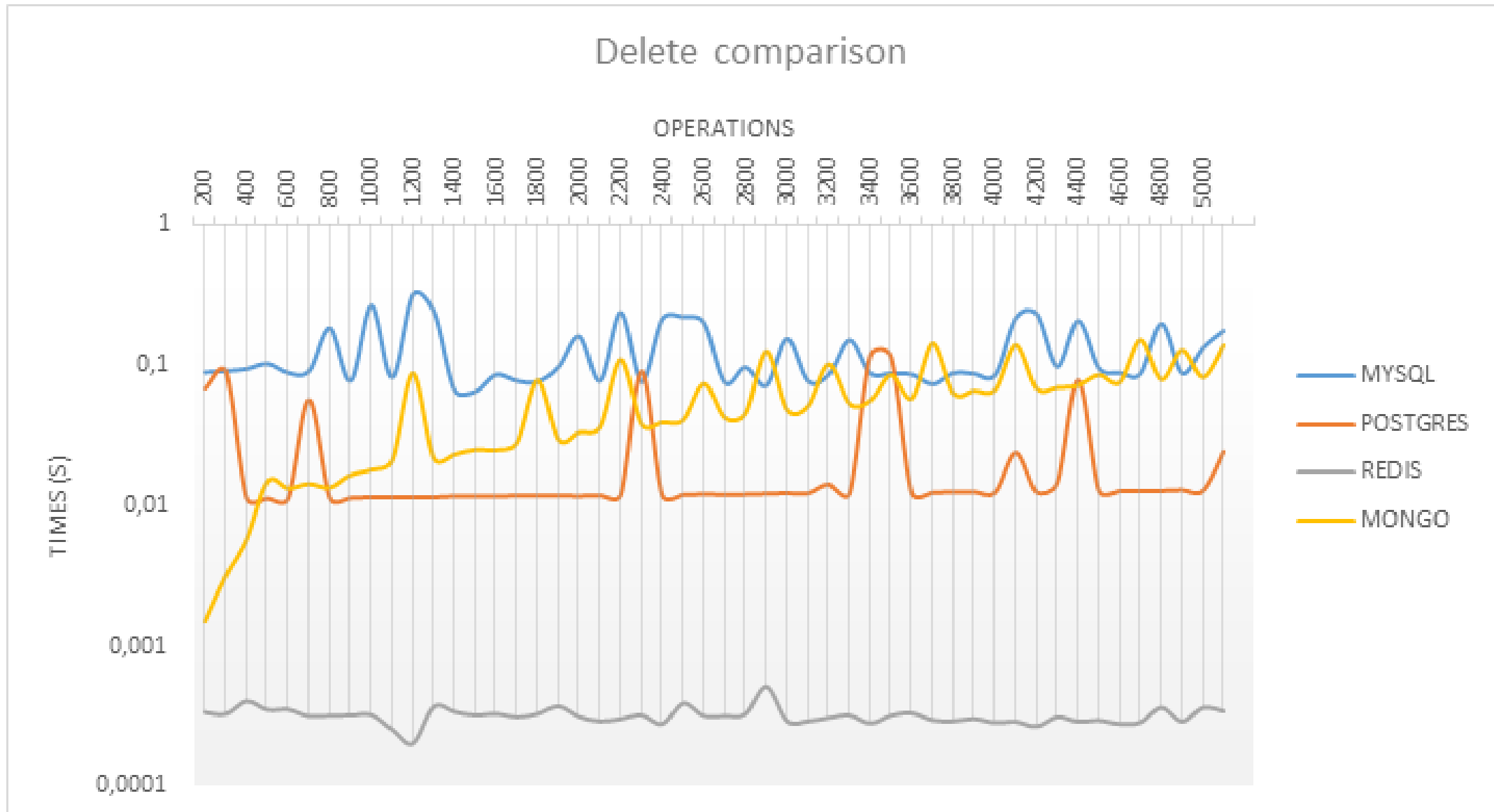




# CRUD: REDIS e MONGO - UPDATE



# CRUD: REDIS e MONGO - DELETE



# COMPARAÇÕES

- Redis teve uma performance melhor nas comparações anteriores porque ele trabalha como um HashMap, assim, quando você utiliza operações de SELECT para achar uma informação (**arquivo**) é esperado que essa abordagem ganhe. Nos bancos relacionais, você utilizaria um **canhão pra matar uma mosca**.
- NoSQL é mais rápido para **armazenar arquivos inteiros**. Contudo, operações como **JOIN** ou análises de atributos envolvendo diferentes bases de dados **ainda vão ser mais rápidas nos bancos relacionais**.



# BENEFÍCIOS DOS RELACIONAIS

- Conseguir lidar com “complexas” queries com operações diversas, exemplo:

```
'SELECT activities.id, activities.moduleid, activities.title, activities.creationtime, activities.deadline, activities.repeatamount, modules.name
FROM activities
JOIN modules ON activities.moduleid = modules.id
WHERE activities.userid = ?
ORDER BY modules.name ASC, activities.creationtime DESC'
```

# BENEFÍCIOS DOS RELACIONAIS

- Planejado para **lidar** com **tabelas e operações de conjunto**.
  - Nunca é interessante trabalhar com tabelas com poucas colunas ou poucas linhas, nesse caso, **talvez sua base não seja relacional** na essência.
- No mesmo sentido da opção anterior, é **mais eficiente** para operação de **conjuntos** no geral. Por quê?
- Normalmente possui um ambiente de administração do banco (PHPmyAdmin do **MySQL**, PgAdmin do **Postgree**, etc).
  - Fácil de realizar backup e manutenção (**exportar e importar**).
  - Fácil de editar ou modificar algo nos dados com **comandos SQL internos**, sem precisar de programação extra.
  - **Não necessita de um conhecimento grande** de programação.



# BENEFÍCIOS DOS RELACIONAIS

- Maior **suporte**. Isto é, os bancos não relacionais normalmente são open source e não fornecem qualquer tipo de suporte.
- Mais maduro, portanto **menos propenso** a erros (pelo tempo de amadurecimento).

# BENEFÍCIOS DOS NÃO-RELACIONAIS

- Consegue armazenar imagens e arquivos  **muito grandes** .
- Muito maior **escalabilidade** (o que é?). Pense em termos de muitos computadores com bases de dados, por que os não-relacionais tem maior escalabilidade?
  - É muito **custoso e, na grande maioria dos casos, impossível** realizar as operações propostas e disponíveis na linguagem SQL em um ambiente de servidores (com múltiplos computadores rodando bancos).
    - Como se roda um **JOIN** do SQL em dados que estão em diversas máquinas?

# BENEFÍCIOS DOS NÃO-RELACIONAIS

- Consegue armazenar imagens e arquivos  **muito grandes** .
- Muito maior **escalabilidade (o que é?)**. Pense em termos de muitos computadores com bases de dados, por que os não-relacionais tem maior escalabilidade?
  - É muito **custoso e, na grande maioria dos casos, impossível** realizar as operações propostas e disponíveis na linguagem SQL em um ambiente de servidores (com múltiplos computadores rodando bancos).
    - Como se roda um **JOIN** do SQL em dados que estão em diversas máquinas?
- Por ser escalável mais facilmente, **gera menos custos** em termos de manutenção de grandes sistemas. Seria impossível, por exemplo, pra empresas como Google, Facebook, etc, utilizar bases relacionais. Seus sistemas são gigantemente distribuídos.



# ONDE ESTÁ A DIFERENÇA

- Resumindo bastante, a principal diferença de eficiência entre as abordagens está no **tempo de busca** da informação e nas **operações realizadas**.
- Como a **informação** é buscada?
- Arquivos como **video, imagens**, etc, se beneficiam de uma busca relacional?

# ONDE ESTÁ A DIFERENÇA

- Resumindo bastante, a principal diferença de eficiência entre as abordagens está no **tempo de busca** da informação e nas **operações realizadas**.
- Como a **informação** é buscada?
  - Buscas de em tempo de execução/complexidade de **log** são as mais eficientes. O que isso quer dizer? O que isso tem a ver com ordenação?
  - <https://gist.github.com/psayre23/c30a821239f4818b0709>
- Arquivos como **video, imagens**, etc, se beneficiam de uma busca relacional?
  - Esses arquivos **não se beneficiam de uma forma tão direta** como uma base que armazena dados de usuários, por exemplo.



# PERGUNTAS

- Qual modelo seria mais interessante para a parte de vídeos do Youtube?



# PERGUNTAS

- Qual modelo seria mais interessante para a parte de vídeos do Youtube?
  - O modelo **não relacional**. Os videos são arquivos muito grandes e que não tem certas características específicas que podem ser extraídas sem pré-processamento. Videos, por si só, também não se encaixam no padrão de orientação a objeto.



# PERGUNTAS

- Qual modelo seria mais interessante para a parte de vídeos do Youtube?
  - O modelo **não relacional**. Os videos são arquivos muito grandes e que não tem certas características específicas que podem ser extraídas sem pré-processamento. Videos, por si só, também não se encaixam no padrão de orientação a objeto.
- Qual modelo seria mais interessante para uma rede social como o Facebook?



# PERGUNTAS

- Qual modelo seria mais interessante para a parte de vídeos do Youtube?
  - O modelo **não relacional**. Os videos são arquivos muito grandes e que não tem certas características específicas que podem ser extraídas sem pré-processamento. Videos, por si só, também não se encaixam no padrão de orientação a objeto.
- Qual modelo seria mais interessante para uma rede social como o Facebook?
  - **Depende**. Inicialmente o facebook era feito em PHP e MySQL. Contudo, sistemas muito grandes normalmente tem muitas subpartes que podem ser implementadas de diferentes formas.



# PERGUNTAS

- Qual modelo seria mais interessante pra um jogo tático online?
- E para um jogo de tiro online em tempo real, tem alguma diferença?

# EXEMPLO COM MYSQL



- <https://github.com/Oyatsumi/NarutoBrowserMmorpg>





# FORMA CORRETA

- Na verdade, a forma correta de se trabalhar com jogos é utilizando uma **cache**.
- É necessário **minimizar o tempo de acesso ao banco o máximo possível**, então você carrega as informações dos jogadores na memória **RAM**, e evita o acessar novamente.
  - Em um RPG, por exemplo, o ideal é **salvar no banco de tempos em tempos e de forma gradual** para não impactar o desempenho.
- É possível trabalhar com todos os tipos de bancos desde que o restante da arquitetura esteja bem trabalhada.
  - Como você dividirá os servidores para lidar com muitos jogadores?
  - Os jogadores possuem muitas características pessoais como num jogo de RPG ou não?



# OUTRAS DIFERENÇAS (RELACIONAL VS OO)

- Herança
  - **Presente** no modelo O.O.
  - **Inexistente** no modelo relacional.
- **Identidade** dos objetos
  - Java
    - Operador ==
    - Método *equals()*
  - Banco de dados relacional
    - Chave **primária**

# OUTRAS DIFERENÇAS (RELACIONAL VS OO)

- O modelo de OO possui associações unidirecionais e bidirecionais.
  - Junção de tabelas não possuem o **conceito de “direção”**.
- Associações em OO podem ser do tipo **muitos-para-muitos** (many-to-many).
- Associações entre tabelas **só podem ser uma-para-muitas** (one-to-many) e uma-para-uma (one-to-one).
  - Precisa **criar uma tabela de relacionamentos** para associação **many-to-many**.



# JAVA PERSISTENCE API (JPA)

- **Especificação JSR 317:** Java Persistence 2.0
- Especificação elaborada pelo Java Community Process para **persistência em Java**.
- Baseou-se em diversas **soluções existentes** (grande parte baseou-se nas funcionalidades presentes **no framework Hibernate**).
- Frameworks existentes **passaram a implementar a especificação**.
- Se trata de um conjunto de regras pra **mapear objetos em tabelas relacionais**.



# JAVA PERSISTENCE API (JPA)

- Permite a **persistência de objetos** em tabelas de uma base de dados relacional.
- Tenta deixar a relação entre os objetos e tabelas ser **automática e transparente**.
- Utiliza **metadados** para descrever o relacionamento entre os objetos e a base de dados:
  - XML
  - Annotations (as anotações que começam com @ no java – por exemplo: @Override)



# JAVA PERSISTENCE API (JPA)

- O **SQL é gerado automaticamente** a partir dos metadados.
- A escrita e manutenção de metadados necessita de esforço nas **etapas de implementação**.
  - Esforço bem menor do que o necessário para fazer a conversão manualmente.
- A conversão entre os tipos de representação pode trazer **perda de performance**.
  - Ferramentas maduras otimizam a conversão em diversos pontos.



# CONS VS PROS

- Vantagens:
  - Produtividade.
    - Maior **tempo disponível para implementar a lógica da aplicação**.
  - Elimina a necessidade de **escrita de grande parte do código** relativo a persistência.
    - Camada de **abstração**.
  - Menos código.
    - **Manutenção** mais rápida.
    - Facilita a **refatoração**.
- Desvantagens:
  - Mais lento (**perde eficiência**).
  - Preso a um framework/biblioteca específica e preso ao JPA.



# VEREMOS

- Mais pra frente veremos:
  - **Serialização** em Java
  - Uma biblioteca/framework que faz a conversão de **objetos para SQL** (Hibernate)

**Pergunta:** Qual a diferença entre biblioteca e framework?