FIAP

# Sistemas de informação

## enterprise analytics and data warehousing

Profº Fabiano J. Cury Marques

FIAP

# enterprise analytics and data warehousing

## ETL - Projeto físico II

Boa noite!

# Agenda

- ✘ Introdução
- ✘ Abordagens e Arquitetura
- ✘ Extração de dados
  - ○ Bases relacionais
  - ○ Outras fontes
- ✘ Exercícios
- ✘ Referências


# Introdução

Introdução

# Introdução

- ETL = Extract, Transform, and Load;
- É o processo de recuperar e transformar dados dos sistemas fontes;
- Começaremos a falando sobre a extração de dados;
- Existem diversos princípios básicos envolvidos na extração de dados dos sistemas fontes;

# Extração de dados – Princípios Básicos

- O volume de dados que será recuperado é grande, provavelmente centenas de megabytes ou até dezenas gigabytes
- Um sistema OLTP é projetado para que os dados sejam recuperados em pequenos pedaços, não em grandes quantidades
- Assim deve-se ter cuidado para não deixar os sistemas fontes muito lentos durante as extrações
- Deseja-se que a extração seja o mais rápido possível, como cinco minutos, e não três horas. Também que seja o menor possível, como 10 Mb por dia e não 1 Gb. Por fim, menos frequente possível, 1 vez ao dia e não a cada 5 min
- Deve-se alterar o menos possível os sistemas fontes;

# Extração de dados – Princípios Básicos

- Se você tivesse que se lembrar de uma coisa para sempre sobre o processo de extração de dados é:

  - Quando extrair dados de um sistema fonte, tenha muito cuidado para não atrapalhar demais estes sistemas;

# Extração de dados – Princípios Básicos

- Depois da extração de dados, deseja-se colocá-la no DW o mais breve possível, idealmente sem sequer passar por um disco intermediário

- É necessário aplicar algumas transformações nos dados vindos das fontes para que eles possam ser enquadrados no formato e estrutura desejado pelo NDS ou DDS

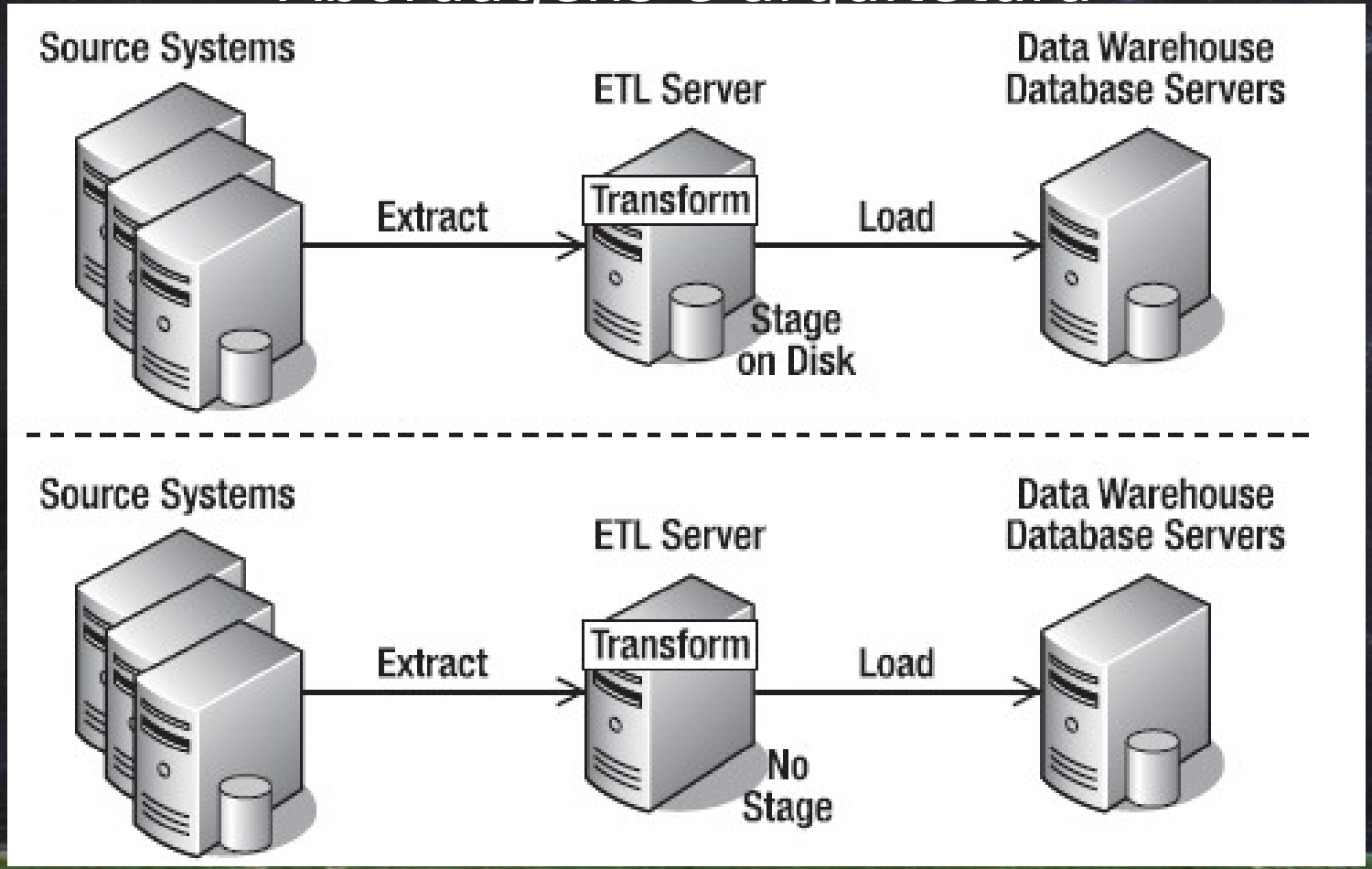
- Dois outros princípios importantes:
  - Leakage: quando o processo de ETL acha que trouxe toda a informação dos sistemas fontes mas na verdade perdeu algumas linhas
  - Recoverability: processo deve ser robusto o suficiente para que, se ocorrer algum problema, ele possa ser recuperado sem perda de dados


Abordagens e arquitetura

# Abordagens e arquitetura

- Existem diversas abordagens para implementar ETL
- **Uma abordagem tradicional** é puxar os dados dos sistemas fonte, colocá-los na área de stage, e então transformá-los e carregá-los nos repositórios NDS ou DDS
- **Uma alternativa** que pode ser utilizada é, em vez de colocar os dados na área de stage, fazer as transformações na memória e então atualizar o NDS ou DDS diretamente
- Colocar os dados na memória é mais rápido do que colocar no disco. Se a quantidade de dados for pequena é possível, porém, com muitos dados torna-se inviável

# Abordagens e arquitetura



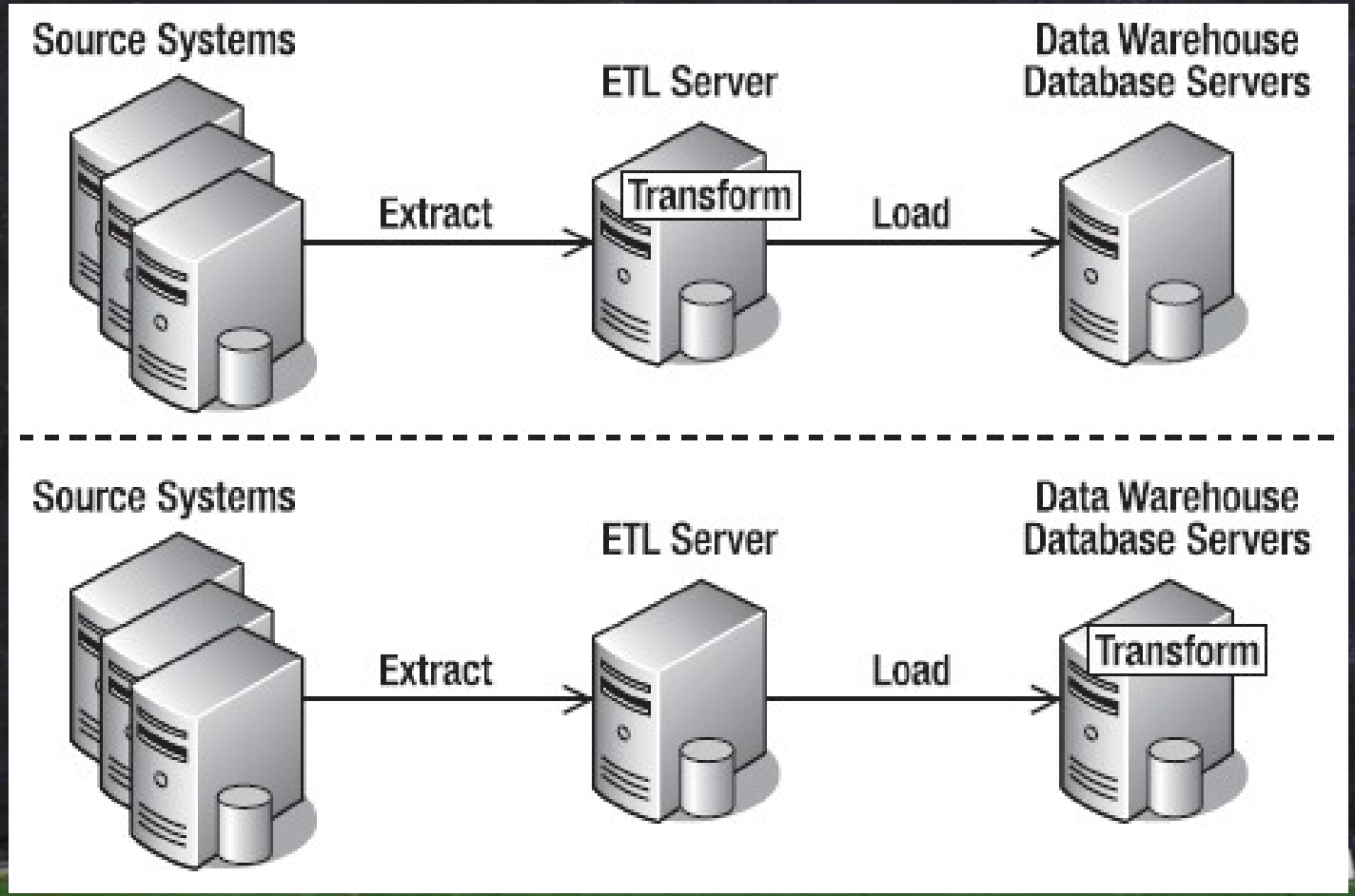
# Abordagens e arquitetura

- A alternativa para as duas abordagens apresentadas anteriormente é chamada de ELT: Extract, Load, and Transform

- Nesta abordagem, puxa-se os dados dos sistemas fontes, carrega no DW e então aplica as transformações  atualizando o dado nos repositórios NDS ou DDS

- O ELT é mais utilizado quando existe um servidor de base de dados muito robusto, geralmente com processamento paralelo etc. porém não tem um servidor de ETL robusto o suficiente para fazer o processamento das transformações

# Abordagens e arquitetura



# Abordagens e arquitetura

- Considerando a forma como o dado sai dos sistemas fontes, podemos classificar os métodos de ETL em:

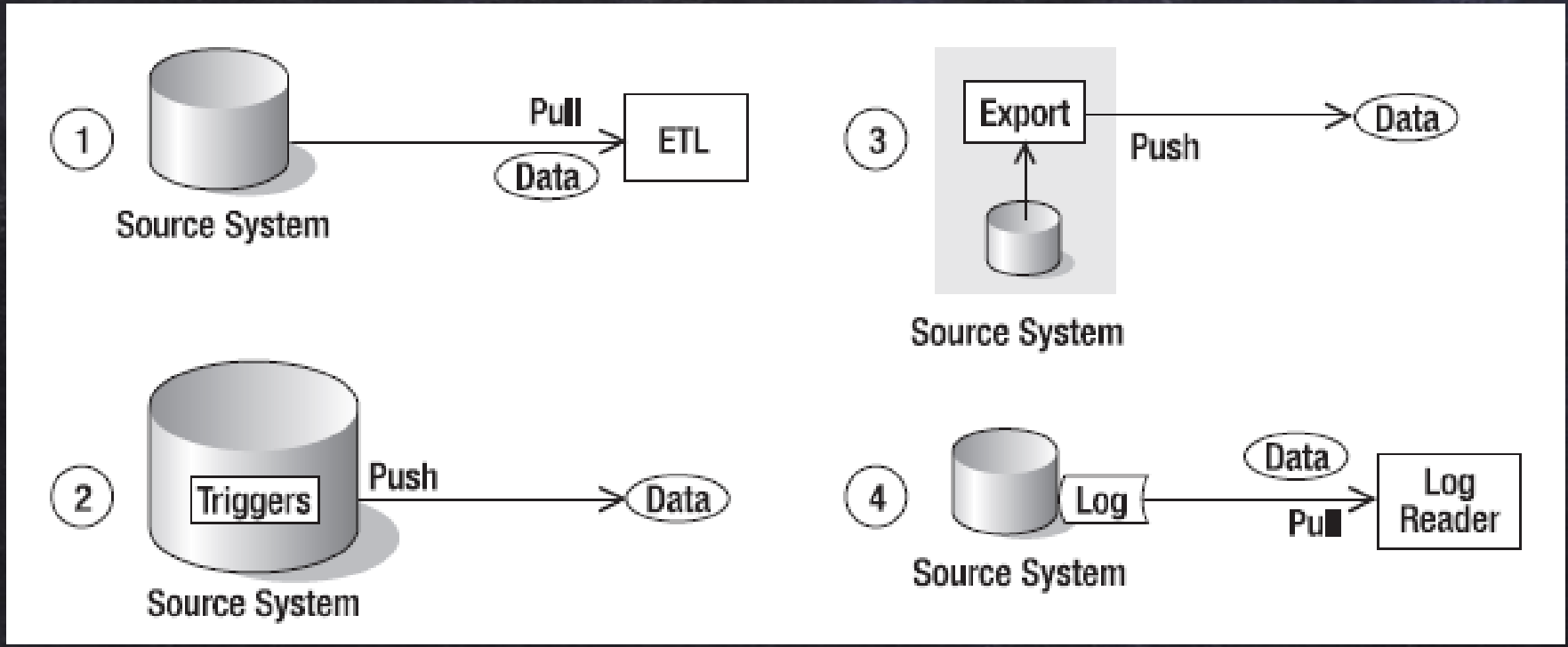
1. Um processo ETL obtém os dados consultando a base de dados do sistema fonte regularmente

2. Triggers na base de dados dos sistema fontes gravam as informações no DW

3. Um processo agendado dentro do sistema fonte exporta os dados regularmente (exportações agendadas)

4. Um log reader lê os arquivos de log para identificar as mudanças de dados feitas na base. Com isso, lê a informação e grava no DW

# Abordagens e arquitetura



# Abordagens e arquitetura

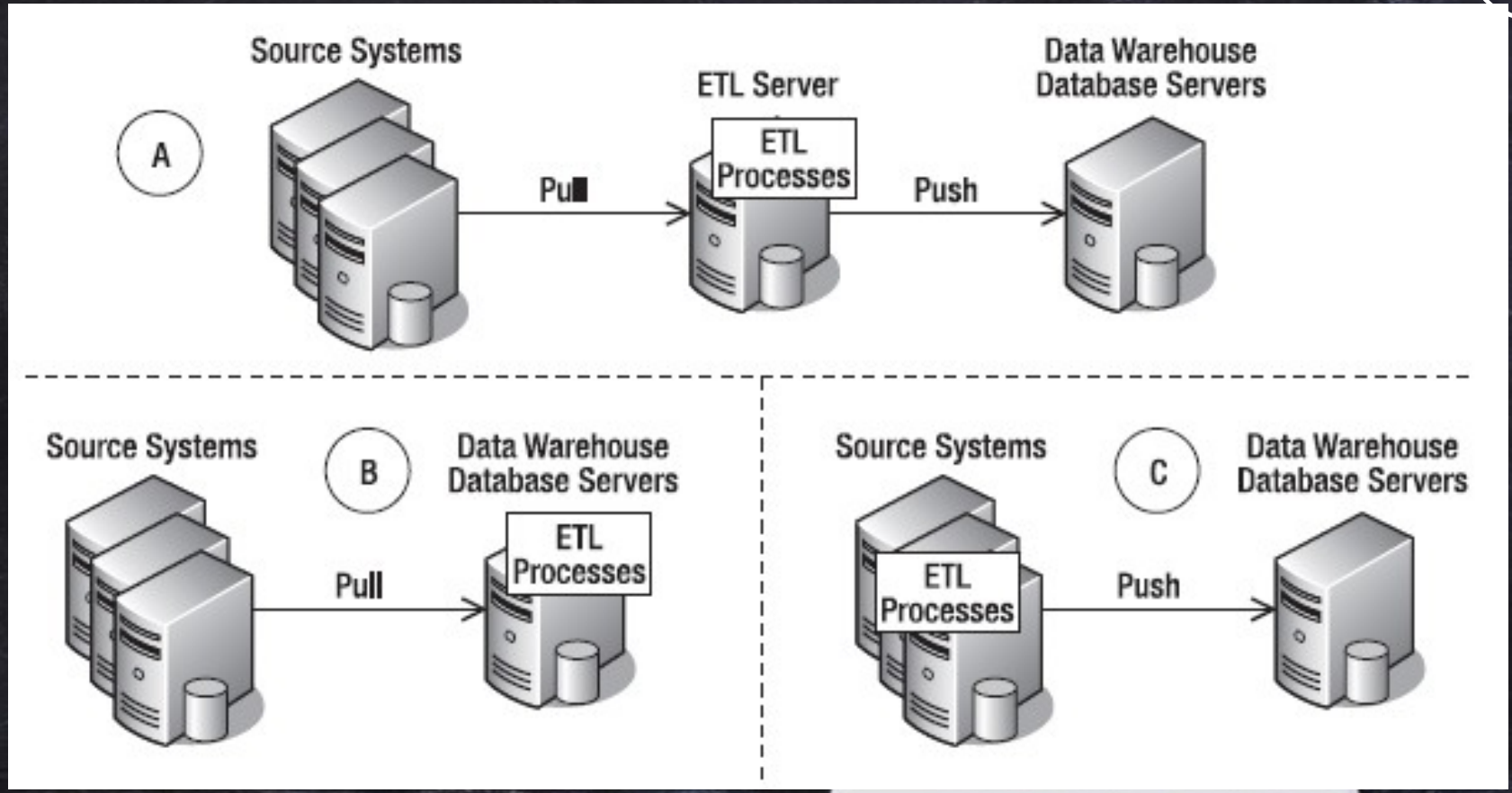
- Em termos de onde são executados os processos que extraem os dados temos:

A. Executar os processos ETL em um servidor separado que fica entre os sistemas fontes e as bases de dados do DW

B. Executar os processos ETL no servidor das bases de dados do DW

C. Executar os processos ETL nos servidores que hospedam os sistemas fontes

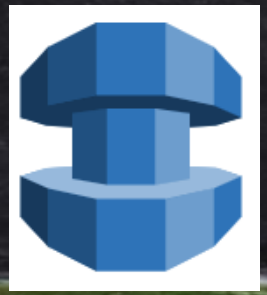
# Abordagens e arquitetura



# Ferramentas de Extração de dados AWS

- **Lambda Function** – Serviço Serverless que pode ser utilizado para aplicações de ETL;
- Limitado ao tempo de execução de 1 segundo a 15 minutos;
- Limitado à alocação de memória de 128 MB a 10.240 MB;

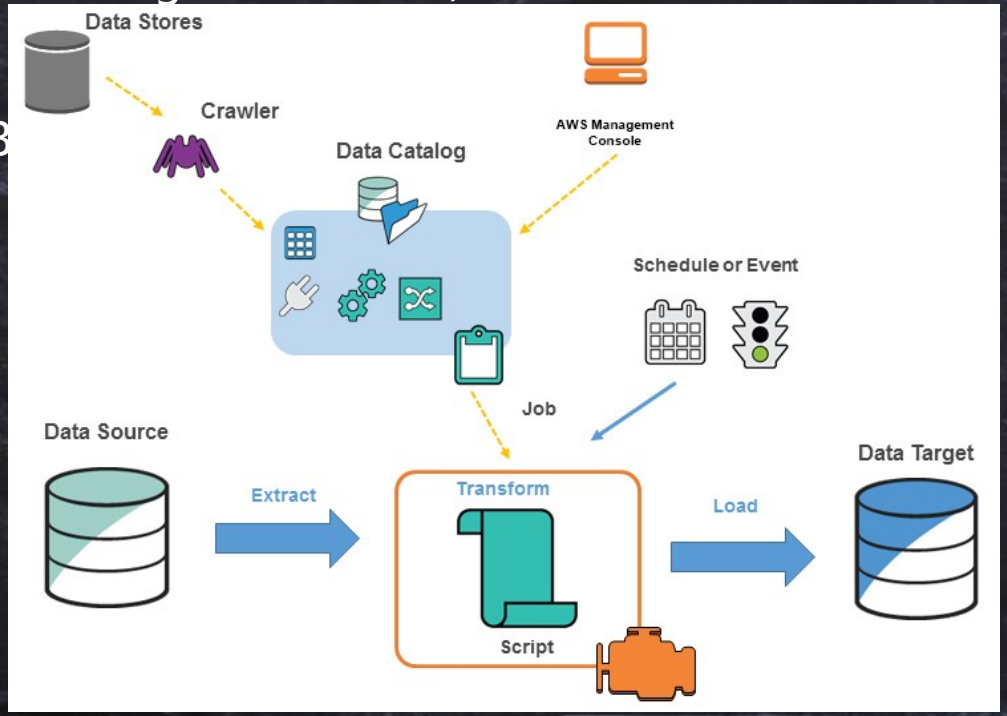
- Glue – Serviço Serverless da utilizado para realizar extração, transformação e carga de dados;



- DMS – (DataBase Migration Service) Migração de dados diretamente do sistema fonte, minimizando o tempo de inatividade do lado transacional;

# Arquitetura do AWS Glue

- Glue – Serviço Serverless da utilizado para realizar extração, transformação e carga de dados;
- Data Stores – Bucket ou um DB Relacional;
- Data Source – Dado de entrada;
- Data Catalog – Metadados;
- Data Target – Dado de saída;



# Exercício

1. Utilizando o aplicativo web draw.io, desenho uma arquitetura mínima de DW utilizando componentes AWS:

Premissas:

- Devem ser realizadas extrações periódicas de dois DBs do Ambiente Transacional de forma transparente, sem que sejam notadas as extrações. Nesse caso, algumas tabelas serão mapeadas e copiadas identicamente na região de stage
- Um canal relacionado a cotação de determinados papeis da bolsa de valores devem ser monitorados e estima-se que produzem, no total, 500 kb/hora. A leitura deve ser periódica por meio de chamada API;


3.
Extração de dados
Extração de dados

# Extração de bases de dados relacionais

- Depois de estabelecer conexão com a fonte de dados (via JDBC, ODBC, ADO.NET, OLEDB ou outros adaptadores), deve-se começar a extrair os dados

- Pode-se usar um dos seguintes métodos:
  - Tabela inteira sempre
  - Extração incremental
  - Intervalo fixo
  - Abordagem Push

# Tabela inteira sempre

- Usa-se quando a tabela é pequena
- Outro motivo comum pode ser a falta de colunas timestamp ou sequenciais que usa-se para descobrir quais linhas foram atualizadas desde a última extração
- Exemplo:

| Table | Code | Description |
|---|---|---|
| PMT | 1 | Direct debit |
| PMT | 2 | Monthly invoice |
| PMT | 3 | Annual in advance |
| STS | AC | Active |
| STS | SU | Suspended |
| STS | BO | Balance outstanding |
| SUB | S | Subscribed |
| SUB | U | Unsubscribed |
| ... | | |

# Tabela inteira sempre

- ✘ No exemplo anterior, como não temos uma coluna timestamp, não temos data de transação (não é uma tabela de transação) e nem coluna sequencial, não há como saber quais linhas são novas, quais foram excluídas ou incluídas
- ✘ Nestes casos, com sorte as tabelas são pequenas para não impactar demais no processo de ETL
- ✘ Alguma vezes mesmo com os campos  acima mas quando a tabela é muito pequena, exemplo, 1000 linhas, é mais rápido puxar a tabela inteira do que fazer uma consulta com um filtro específico:

```
example, select * from stores where (createtimestamp > 'yyyy-mm-dd hh:mm:ss' and
createtimestamp <= 'yyyy-mm-dd hh:mm:ss') or (updatetimestamp > 'yyyy-mm-dd
hh:mm:ss' and updatetimestamp <= 'yyyy-mm-dd hh:mm:ss'), it will take the source datab
```

# Extração incremental

- As tabelas de transação das grandes empresas são enormes, contendo centenas de milhares ou até mesmo centenas de milhões de linhas
- Pode-se levar dias para extrair uma tabela inteira, o que é uma operação muito intensiva para os discos e que degrada demais o desempenho do transacional por causa do gargalo da base de dados
- Extração incremental é a técnica de recuperar apenas as linhas alteradas no sistema fonte e não a tabela completa
- Pode-se usar diversas abordagens para extrair de forma incremental, tais como, colunas timestamp, colunas sequenciais, datas de transação, triggers ou uma combinação destes

# Extração incremental

- Imagine uma tabela de pedido como esta:

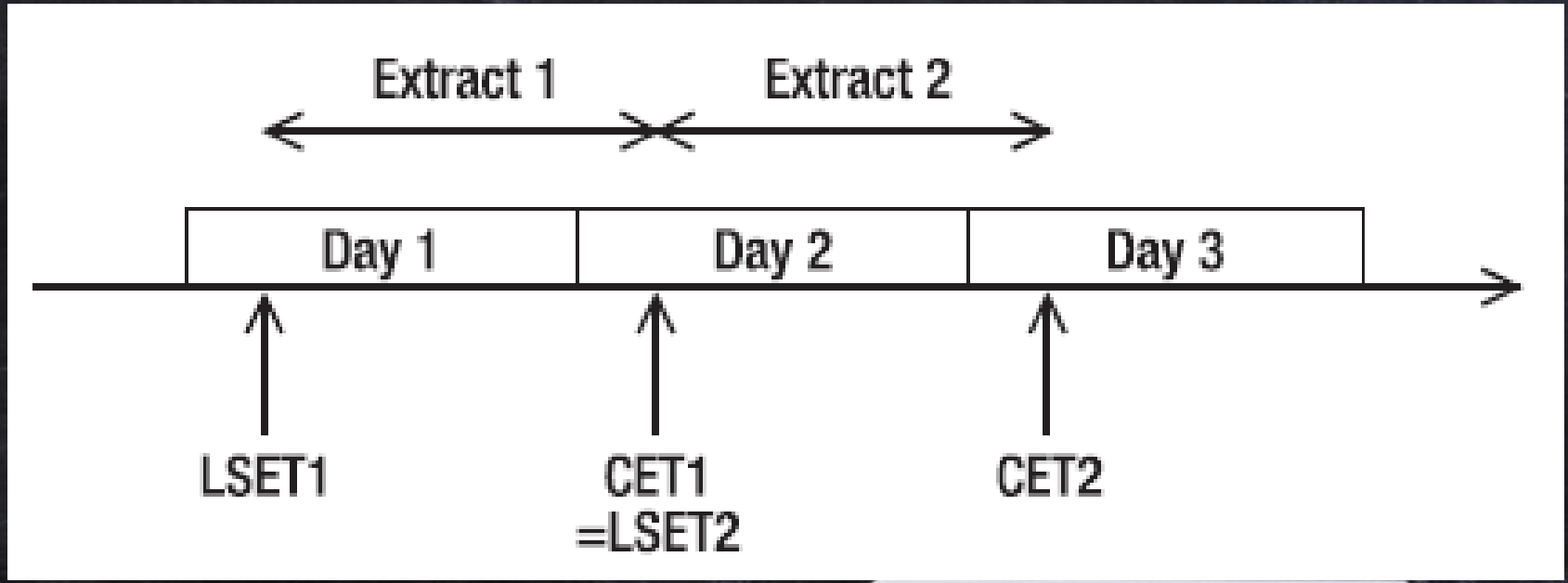
| Order ID | Order Date | Some Columns | Order Status | Created | Last Updated |
|---|---|---|---|---|---|
| 45433 | 10/10/2007 | Some Data | Dispatched | 10/11/2007 10:05:44 | 10/12/2007 11:23:41 |
| 45434 | 10/15/2007 | Some Data | Open | 10/16/2007 14:10:00 | 10/17/2007 15:29:02 |
| 45435 | 10/16/2007 | Some Data | Canceled | 10/16/2007 11:23:55 | 10/17/2007 16:19:03 |
| ... | | | | | |

- Esta é uma tabela ideal para extração incremental. Tem uma coluna created e uma last updated que são timestamp, além disso tem uma coluna sequencial order id e tem também uma data de transação, order date

- Primeiro deve-se verificar se as colunas timestamp são confiáveis, se sim, podemos usá-las

# Extração incremental

- Se as colunas timestamp são confiáveis podemos usá-las nas extrações incrementais como segue:



# Extração incremental

- Basicamente:
  - Recupere o LSET (Last Succesfull Extraction Time) da base de metadados. LSET memoriza o momento da última extração
  - Pega o CET (Current Extraction Time), que é passado pelo processo pai do ETL (de nível mais alto). CET é o momento em que o pacote ETL começou, não o momento em que a tarefa será iniciada
  - Extrai os dados usando

```
SELECT * FROM order_header where
(created >= LSET and created < CET) OR
(last_updated >= LSET and last_updated < CET)
```

  - Se a extração ocorre com sucesso, atualiza a base de metadados escrevendo o CET como o novo valor de LSET

# Extração incremental

- Esta lógica é tolerante a falhas
- Se o processo não rodas ou falhar, pode-se apenas re-executar sem perder dados ou carregando informações que já foram carregadas
- O motivo de restringir o limite de query com CET é para que as linhas que são criadas após começar o processo sejam ignoradas
- Se não tiver como usar colunas timestamp use as colunas data de transação, mas tome cuidado para não perder dados (novos dados com datas antigas)
  - Pode-se usar LSET – 28 dias, por exemplo (28 é definido pelo negócio)

# Extração incremental

- Também é possível recuperar as informações de maneira incremental por um id sequencial
- Recupera o LSEI (Last Successfully Extracted ID) do metadado
- Seleciona o maior id da tabela que será recuperada e coloca o valor no CEI (Current Extraction ID)
- Extrai as linhas entre LSEI e CEI como segue:

```
SELECT * FROM order_header WHERE order_id >= LSEI AND order_id < CEI
```

- Se a extração ocorre com sucesso, guarda CEI no metadado como LSEI

# Extração incremental

- Como tratar os registros excluídos:
- Se for soft delete (marcar como excluído, cancelado etc.) não tem problema pois é uma atualização
- Porém, se for um hard delete (exclusão física da linha), existem duas possibilidades:
  - Comparar a PK entre a tabela fonte e a tabela do DW
  - Usar uma trigger de exclusão que grava em uma tabela de auditoria contendo a PK da linha excluída.
- O processo ETL irá usar essas informações para marcar a linha do DW como excluída

# Intervalo fixo

- Se não é possível carregar a tabela inteira pois é muito grande e também não é possível a carga incremental pois não existem os meios necessários, existe ainda uma abordagem possível conhecido como intervalo fixo
- Basicamente é extrair um certo número de registros ou um certo período de tempo
- Por exemplo, extrair os últimos 6 meses de informação, baseado na data de transação
- Se não existe uma coluna de data, pode-se utilizar um id sequencial do sistema, por exemplo, row id. Assim, pode-se buscar as últimas 100.000 linhas da tabela

# Tabelas Relacionadas

- Se uma linha na tabela fonte é atualizada, é necessário extrair a linha correspondente na tabela relacionada também

- Por exemplo, se o pedido 34552 na tabela order_header é atualizado e extraído para o DW, as linhas deste pedido na tabela order_detail também precisam ser extraídas e vice-versa

- Para isso, use a primeira tabela que está sendo extraída, relacionado as informações pelas FK

# Outras fontes

- Nos projetos é possível que a extração não seja sempre de bases de dados relacionais

- Pode-se ler de arquivos do sistema
  - Flat files (delimitados ou posicionais)
  - XML
  - Excel
  - Web logs

- Pode-se ler também de
  - Web Services
  - Filas de mensagens
  - E-mails

# Exercício

1. Apresente o script SQL para a extração dos dados da tabela abaixo, considerando execuções diárias:

| Order ID | Order Date | Some Columns | Order Status | Created | Last Updated |
|---|---|---|---|---|---|
| 45433 | 10/10/2007 | Some Data | Dispatched | 10/11/2007 10:05:44 | 10/12/2007 11:23:41 |
| 45434 | 10/15/2007 | Some Data | Open | 10/16/2007 14:10:00 | 10/17/2007 15:29:02 |
| 45435 | 10/16/2007 | Some Data | Canceled | 10/16/2007 11:23:55 | 10/17/2007 16:19:03 |
| ... | | | | | |

2. (opcional) Crie a tabela acima utilizando sqlite e implemente um executável para realizar a execução periódica do script previamente construído.

3. (opcional) Partindo do exercício anterior, persista os dados extraídos em um arquivo no formato XML ou JSON. O nome do arquivo deve se diferenciar pela data da extração.

# Referências


- KIMBALL, R., ROSS, M. The Data Warehouse Toolkit. 2ª ed., John Wiley Professional, 2002.
- MACHADO, F. N. R. Tecnologia e Projeto de Data Warehouse. 1ª ed., São Paulo: Ed. Érica, 2004.

# Obrigado!



https://br.linkedin.com/in/wakimsaba