FIAP

Sistemas de informação

Enterprise analytics and data warehousing

Profº Fabiano J. Cury Marques
https://www.linkedin.com/in/fabianocury/

FIAP

ENTERPRISE ANALYTICS AND DATA WAREHOUSING

DW – PROJETO FÍSICO I

Plataforma de hardware

Plataforma de hardware

Introdução

- Após a definição do modelo dimensional e eventualmente o transacional (NDS ou ODS), uma etapa muito importante para o projetista do DW é entender como ele deverá ser criado fisicamente;

- Assumindo que já estão definidos as colunas, tipos de dados e relacionamentos entre as tabelas, deveria ser bem simples criar as bases de dados físicas;

- Porém, alguns detalhes devem ser levados em consideração quando falamos de DW;

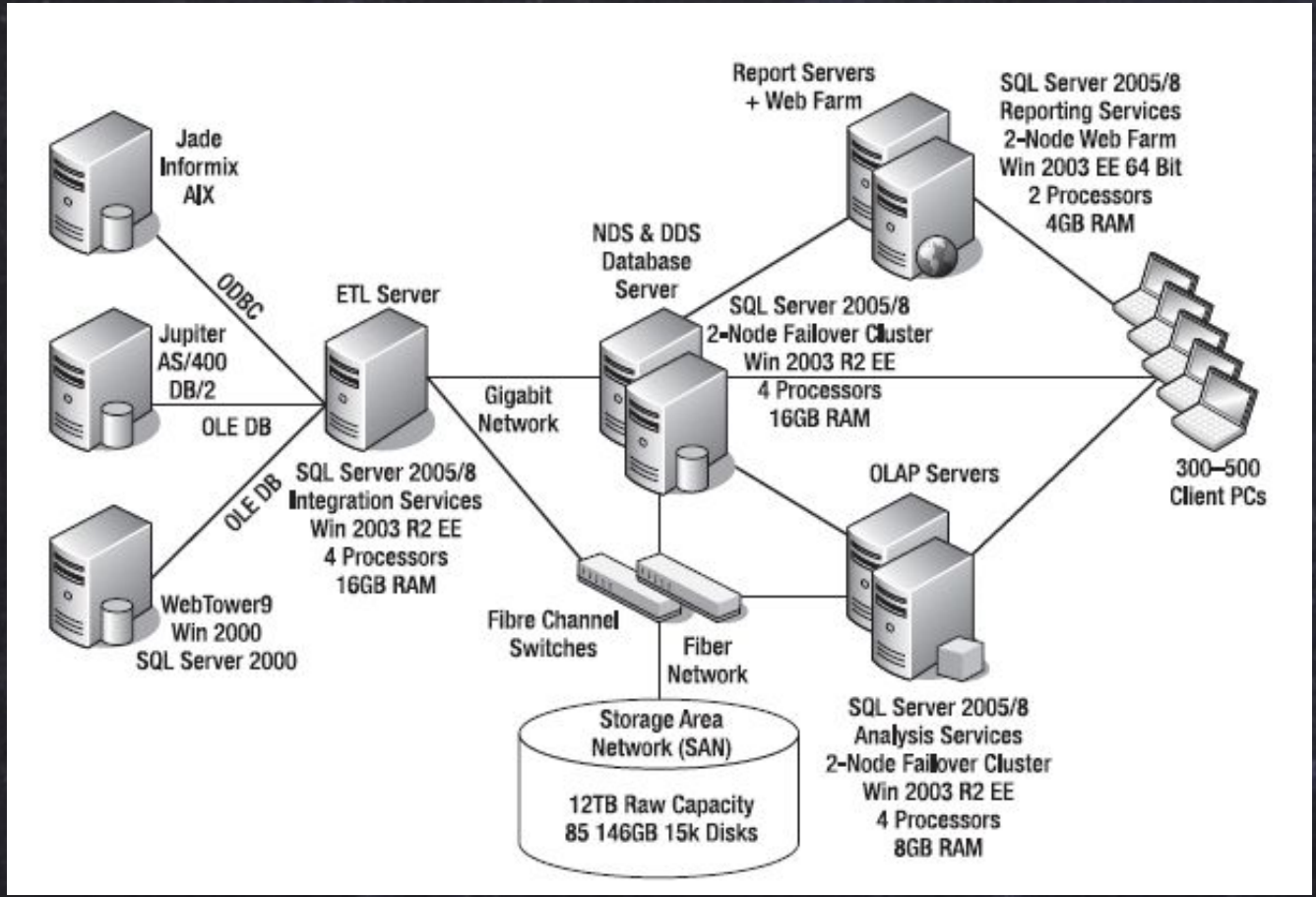
Plataforma de hardware

- A plataforma onde todo o DW será executado tem fundamental importância no desempenho e segurança

- Com frequência, os requisitos não-funcionais direcionam boa parte da arquitetura de hardware do DW

- Exemplos de requisitos não-funcionais comuns para DW:
  - O DW deve estar disponível 24/7
  - Espera-se que o downtime não seja maior que uma hora/mês
  - Deve ser possível reprocessar o DW várias vezes ao dia
  - O Tempo de acesso aos relatórios não podem levar mais que 10 segundo

Plataforma de hardware



Plataforma de hardware

- Algumas decisões claras baseadas nos requisitos:
  - Não podemos contar com apenas um servidor para a base de dados. Precisamos de failover clusters (instalação idêntica em vários servidores – nós – quando um para, outro assume.
  - Em alguns casos pode ser necessário trabalhar com NLB – Network Load Balanced (tráfego de dados é distribuído entre diversos servidores idênticos)
  - Utilização de um SAN (Storage Area Network) ligado com fibra óptica redundante
  - Detalhes como memória e processador de cada servidor devem ser dimensionados de acordo com o uso previsto do DW, no caso do principal servidor (database) alguns critérios devem ser levados em consideração (a seguir).

DIMENSIONAMENTO DO DATABASE SERVER

- ✘ Provavelmente o ponto mais difícil de dimensionar pois depende muito da frequência e de como é utilizado e só sabemos o qução agressivas são nossas queries após elas estarem construídas

- ✘ Pontos a considerar:
  - O número e complexidade de relatórios, aplicações e consultas diretas ao DDS
  - Será usado processo ETL ou ELT para carregar NDS/DDS
  - Cálculos da Stage para NDS/DDS e a complexidade das regras
  - O número e tamanho dos repositórios de dados
  - Como os repositórios são projetados fisicamente (índices, partições etc)
  - A quantidade de outras bases de dados hospedadas no mesmo servidor e crescimento futuro

Storage

Storage

Storage

- A próxima preocupação que devemos ter é o espaço em disco
- Esta estimativa só pode ser feita após o modelo da NDS e DDS estarem definidos
- Isto é feito calculando-se o tamanho das tabelas fato e dimensão para chegar ao tamanho do DDS
- Então estima-se o tamanho do NDS baseado no tamanho da linha e o número de linhas em cada tabela
- Para a Stage, usamos os sistemas que serão fontes de dados para estimar o tamanho

Dimensionando uma tabela

- Inteiro = 4 bytes
- Decimal = 5 bytes
- Money / Datetime = 8 bytes
- Varchar = média do total de bytes

Cálculo

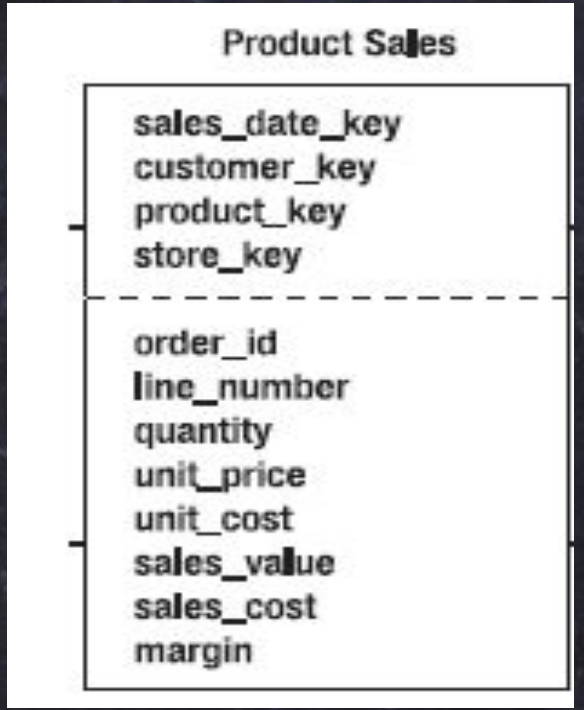
- 6 x 4 + 1 x 5 + 5 x 8 = 69 bytes por linha

Adicione 50% para acomodar novos campos

- 69 + 50% = 69 + 34,5 = 103,5 = 104 bytes

500.000 vendas diárias (usa-se um pouco mais para acomodar o crescimento. Em 2 anos: 500.000 x 365 x 2 x 104 bytes = 37,96 = 40 GB

Deve-se fazer o mesmo tipo de cálculo para cada tabela do DDS, NDS e Stage:

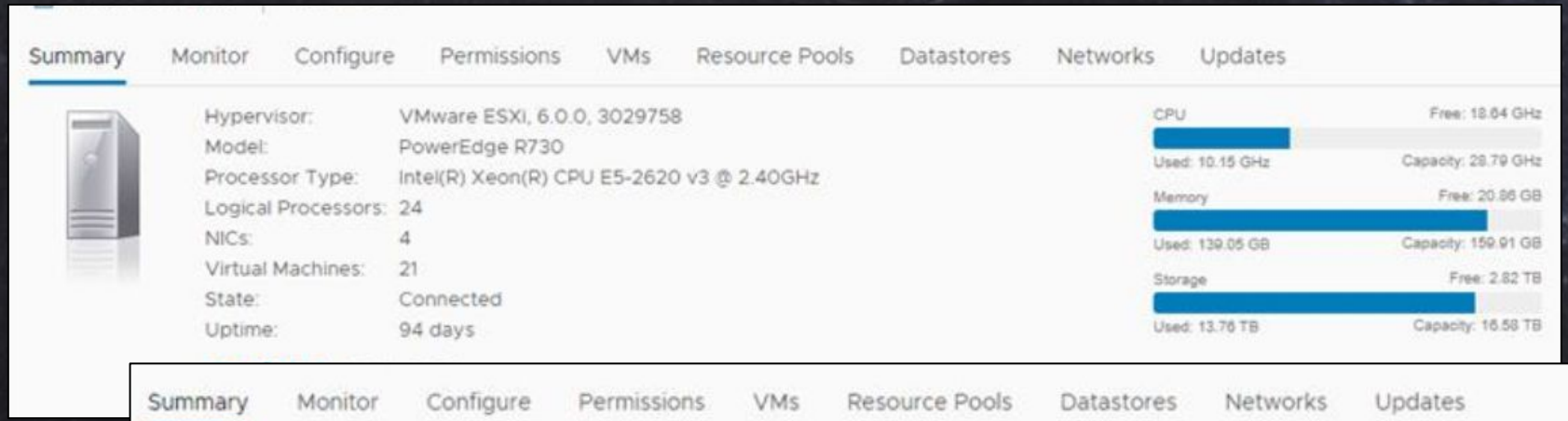


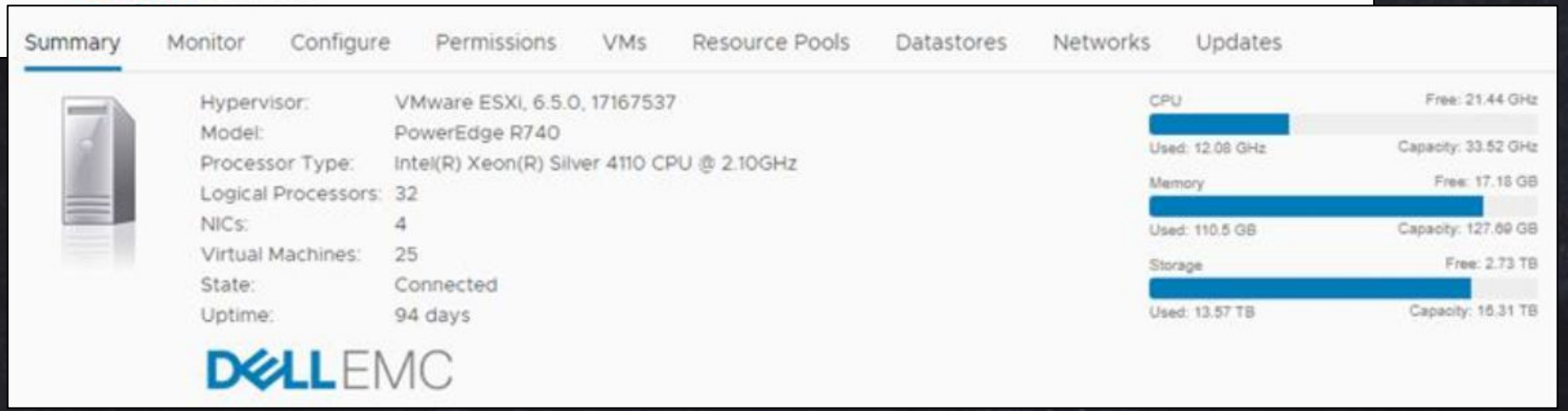
Storage

- Deve-se lembrar de calcular espaço para metadados, índices (30%) e mecanismos internos da base de dados
- Deve-se colocar cada database dividida em vários arquivos de dados e estes divididos em vários volumes físicos
- Para isso, utiliza-se quando RAID 5 forem possíveis (para melhorar o desempenho)
- O arquivo de log deve também ficar em um disco separado

Maquinas On Premisse – DELL

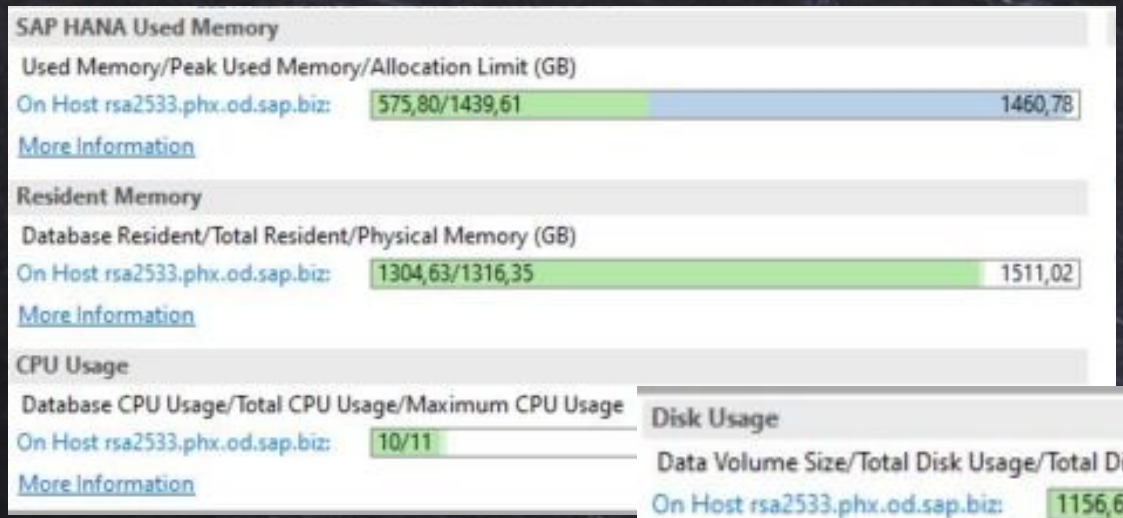
✘ Segue o exemplo de duas máquinas DELL alocadas fisicamente em um empresa:

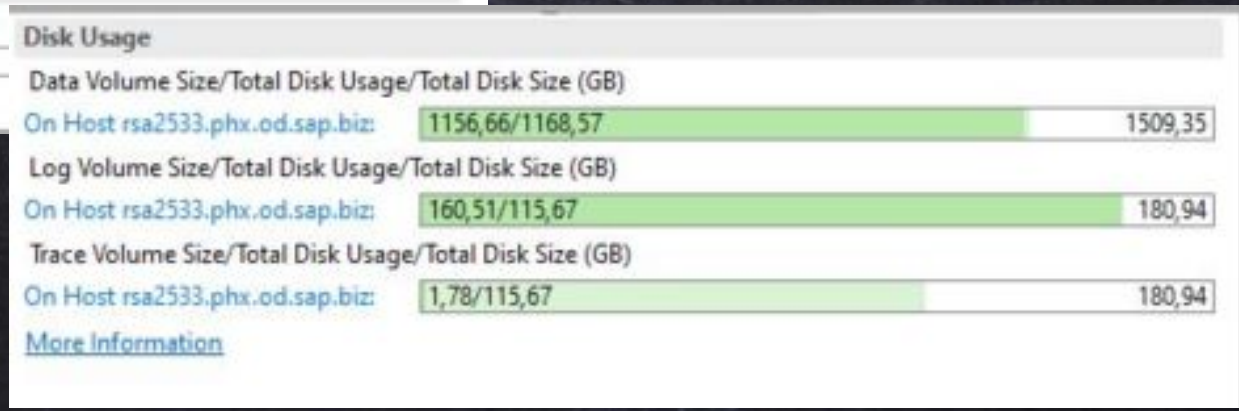




Maquinas On Premisse – HANA

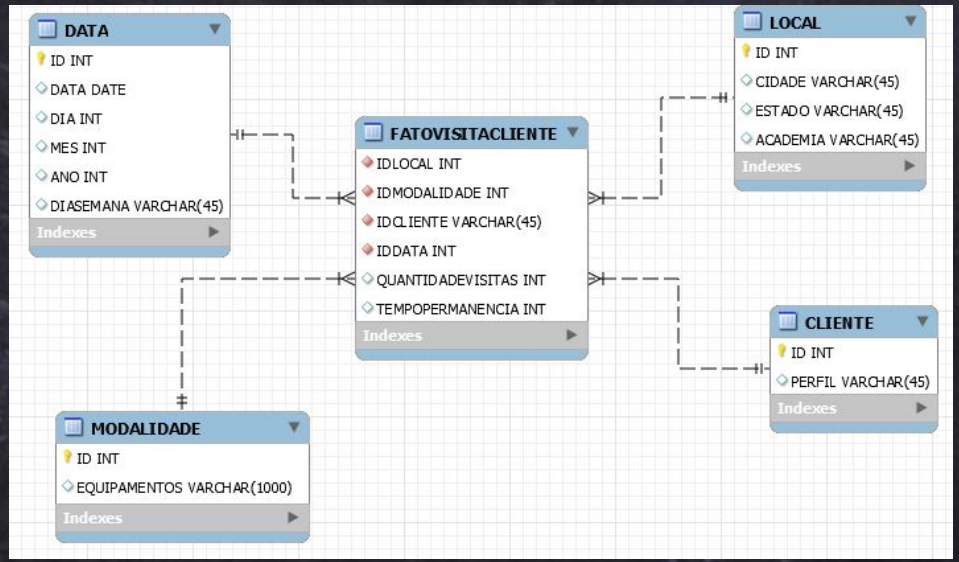
- Segue o exemplo uso de recurso computacional de uma máquina SAP HANA:





Exercício

- Considerando que a empresa "Intelligent Fit" esta definindo o seu primeiro DataMart e quer saber o que deve esperar de dados armazenados no período de 2 anos, faça uma conta rápida para estimar quantos GB a infraestrutura deve suportar. considere:



- Inteiro = 4 bytes
- Decimal = 5 bytes
- Money / Datetime = 8 bytes
- Varchar = média do total de bytes

- Um volume de 1.000.000 de visita Diárias em toda a rede de academia.
- 5.000.000 de clientes cadastrados;
- Qual infra estrutura de máquina onpremisse seria mais adequada ?

FIAP

Base de Dados

Configurando as bases de dados

CONFIGURANDO OS DATABASES

- Alguns detalhes importantes:
  - Usar nomes curtos e claros exemplo: DDS, NDS, Stage etc. Os nomes serão usados como prefixo de outros processos, como ETLs e stored procedures
  - Manter o mesmo collation para todas as bases do DW
  - Considerar case sensitivity com muito cuidado
  - Para cada base de dados criar mais de um filegroup (quanto mais melhor), cada um localizado em um disco físico
  - Defina bons números de tamanho inicial e auto-incremento do tamanho da base de acordo com o crescimento previsto
  - Deixe o máximo grau de paralelismo possível para utilizar todos os processadores
  - Desabilitar funcionalidade não utilizadas

Exemplo de criação de Database

- Exemplo SQL Server:

```
create database DDS
on primary (name = 'dds_fg1'
, filename = 'h:\disk\data1\dds_fg1.mdf'
, size = 30 GB, filegrowth = 5 GB)
, filegroup dds_fg2 (name = 'dds_fg2'
, filename = 'h:\disk\data2\dds_fg2.ndf'
, size = 30 GB, filegrowth = 5 GB)
, filegroup dds_fg3 (name = 'dds_fg3'
, filename = 'h:\disk\data3\dds_fg3.ndf'
, size = 30 GB, filegrowth = 5 GB)
, filegroup dds_fg4 (name = 'dds_fg4'
, filename = 'h:\disk\data4\dds_fg4.ndf'
, size = 30 GB, filegrowth = 5 GB)
, filegroup dds_fg5 (name = 'dds_fg5'
, filename = 'h:\disk\data5\dds_fg5.ndf'
, size = 30 GB, filegrowth = 5 GB)
, filegroup dds_fg6 (name = 'dds_fg6'
, filename = 'h:\disk\data6\dds_fg6.ndf'
, size = 30 GB, filegrowth = 5 GB)
log on (name = 'dds_log'
, filename = 'h:\disk\log1\dds_log.ldf'
, size = 1 GB, filegrowth = 512 MB)
collate SQL_Latin1_General_CP1_CI_AS
go
```

FIAP

Exemplo de criação de tabela

Exemplo SQL Server:

Table Name	Table Type	Filegroup
dim_week	Dimension	6
dim_store	Dimension	6
dim_campaign	Dimension	6
dim_channel	Dimension	6
dim_communication	Dimension	6
dim_customer	Dimension	6
dim_date	Dimension	6
dim_delivery_status	Dimension	6
dim_format	Dimension	6
dim_lead	Dimension	6
dim_package	Dimension	6
dim_product	Dimension	6
dim_subscription_status	Dimension	6
dim_supplier	Dimension	6
fact_product_sales	Fact	2
fact_subscription_sales	Fact	3
fact_supplier_performance	Fact	3
fact_communication_subscription	Fact	5
fact_campaign_result	Fact	5
Fact tables' PKs and indexes		4
Dimension tables' PKs and indexes		6

```
create table dim_date
( date_key              int         not null
, date                  char(10)    not null
, system_date           char(10)    not null
, sql_date              datetime    not null
, julian_date           int         not null
, day                   tinyint     not null
, day_of_the_week       tinyint     not null
, day_name              varchar(9)  not null
, day_of_the_year       smallint    not null
, week_number           tinyint     not null
, month                 tinyint     not null
, month_name            varchar(9)  not null
, short_month_name      char(3)     not null
, quarter               char(2)     not null
, year                  smallint    not null
, fiscal_week           tinyint
, fiscal_period         char(4)
, fiscal_quarter        char(3)
, fiscal_year           char(6)
, week_day              tinyint     not null
, us_holiday            tinyint
, uk_holiday            tinyint
, month_end             tinyint     not null
, period_end            tinyint
, short_day             tinyint
, source_system_code    tinyint     not null
, create_timestamp      datetime    not null
, update_timestamp      datetime    not null
, constraint pk_dim_date
  primary key clustered (date_key)
  on dds_fg6
) on dds_fg6
go
```

PARTICIONAMENTO

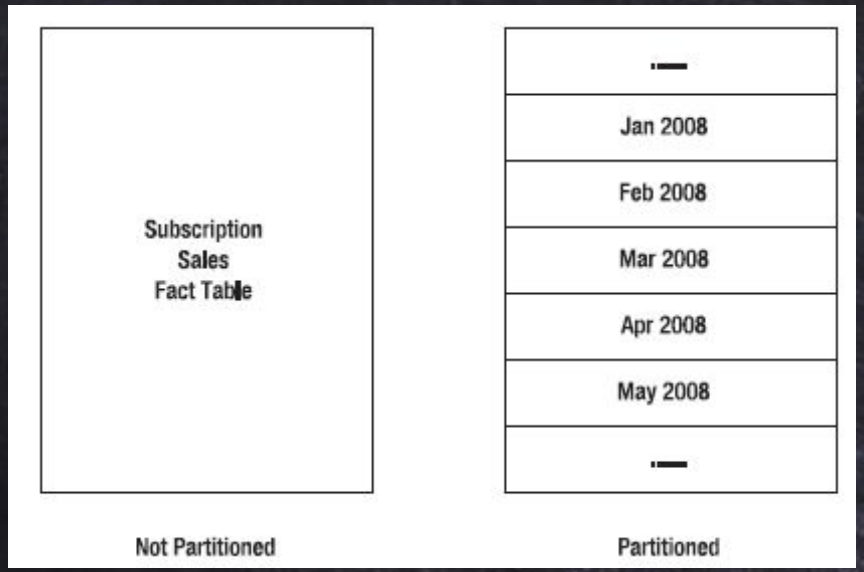
- Existem dois tipos:
  - Vertical: divide uma tabela em várias menores, cada uma contendo algumas colunas da tabela original
  - Horizontal: divide uma tabela em várias menores, cada uma contendo algumas linhas da tabela original

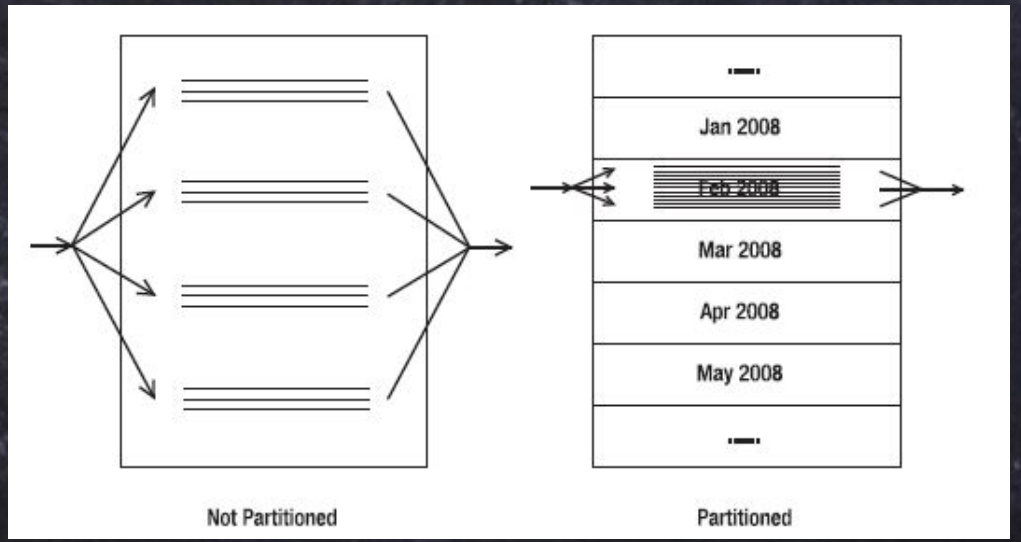
- Em DW usamos normalmente horizontal partitioning

- A natureza das tabelas fatos em um DW é que seu conteúdo é cronológico

- Por causa desta característica, é melhor organizar fisicamente uma tabela pela data. Isto torna a carga muito mais rápida. As consultas também são mais rápidas se a informação vem da mesma partição (mês e ano por exemplo).

Particionamento Horizontal





Banco de dados em cloud

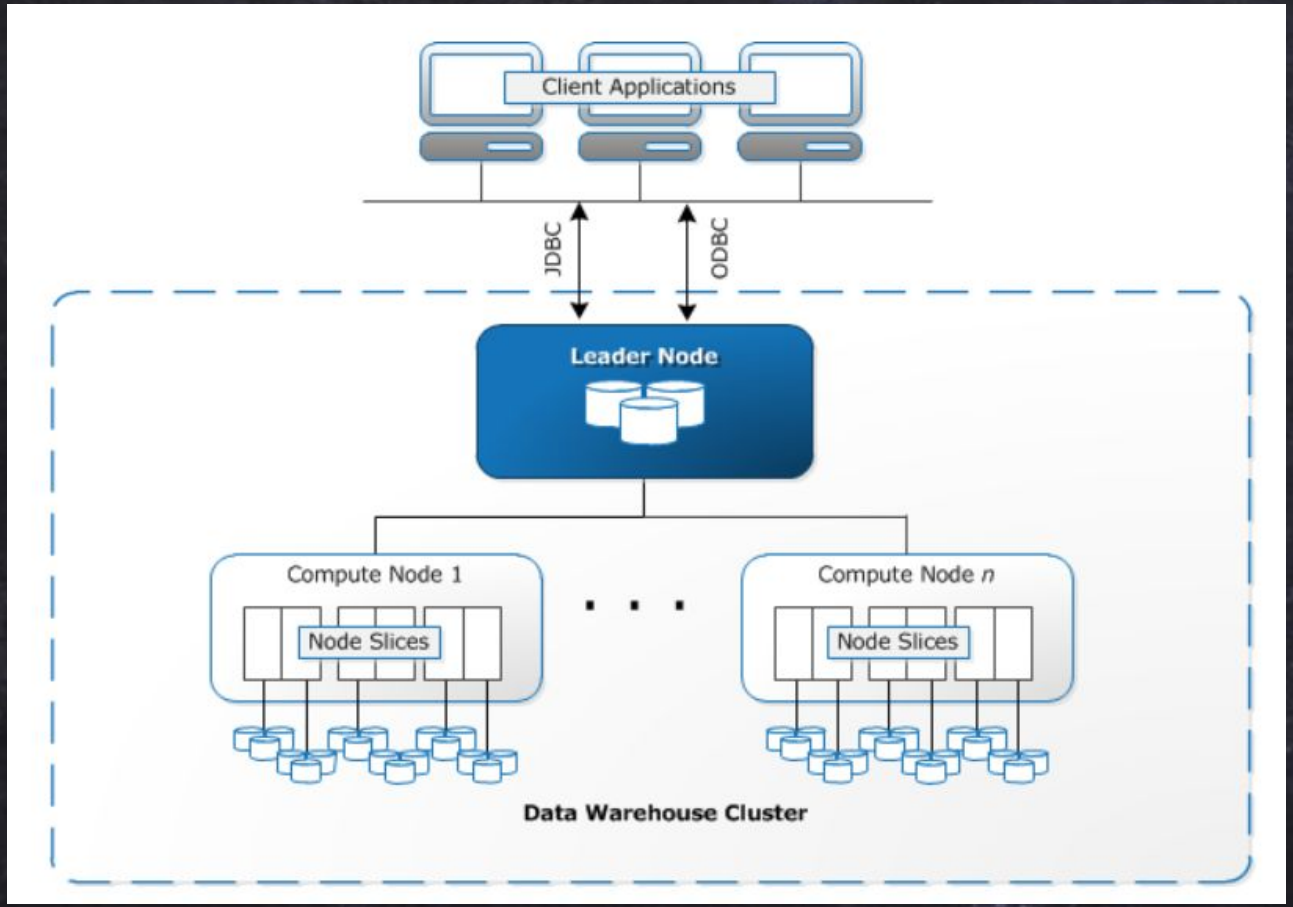
- Existem vários Banco de dados gerenciado disponíveis em ambiente cloud:

  - Big Query – Solução da google consultas de alto desempenho utilizando SQL;

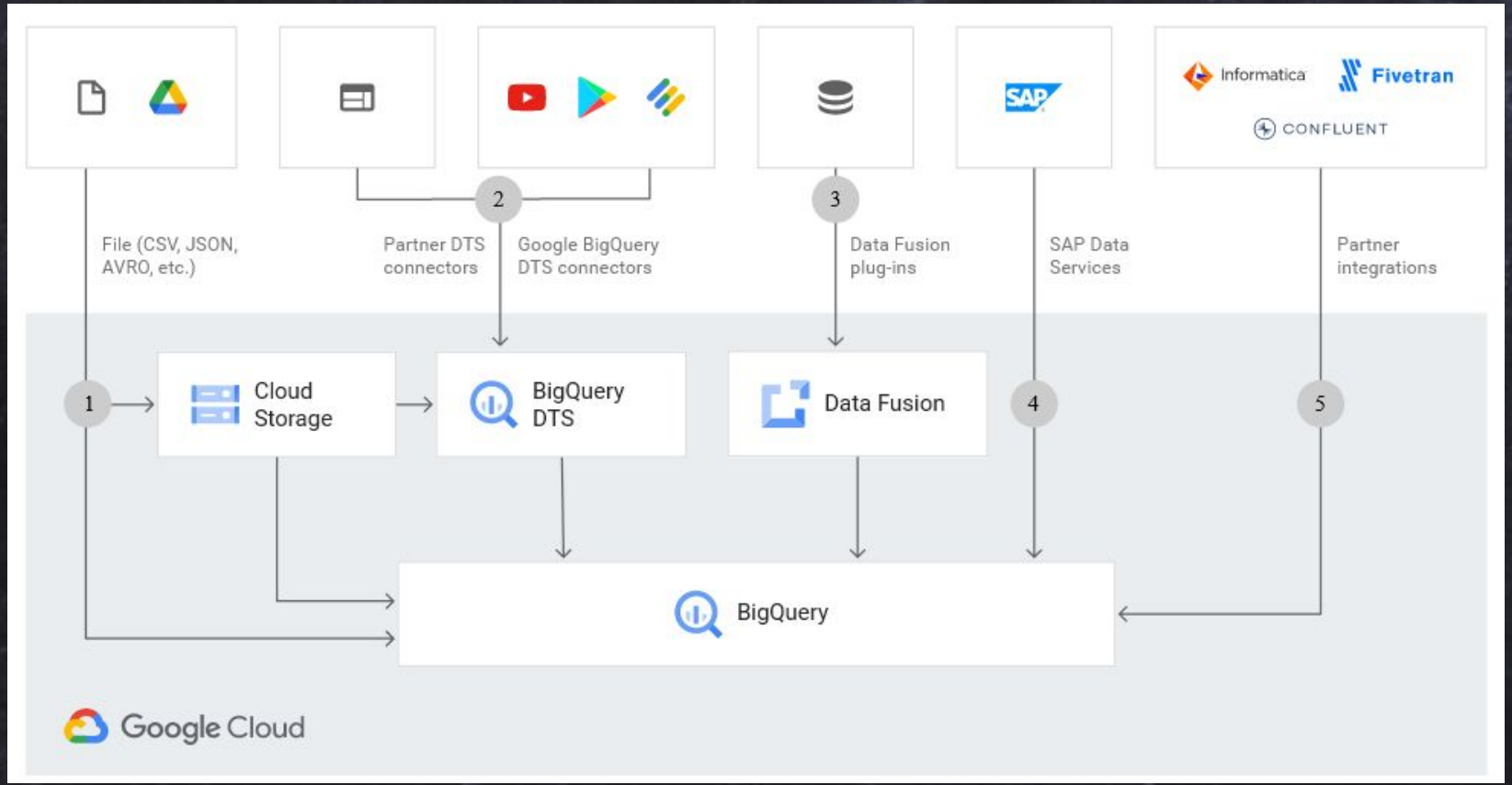
  - RedShift – Solução da AWS para consultas de alto desempenho utilizando SQL;

FIAP

RedShift



BigQuery


Informatica
Fivetran
CONFLUENT
SAP
2
3
File (CSV, JSON, AVRO, etc.)
Partner DTS connectors
Google BigQuery DTS connectors
Data Fusion plug-ins
SAP Data Services
Partner integrations
1
Cloud Storage
BigQuery DTS
Data Fusion
4
5
BigQuery
Google Cloud

BigQuery – Preços

Preços de armazenamento

O preço de armazenamento é o custo para armazenar dados que você carrega no BigQuery. Você paga pelo *armazenamento ativo* e pelo armazenamento de *longo prazo*.

- O **armazenamento ativo** inclui qualquer tabela ou partição de tabela que tenha sido modificada nos últimos 90 dias.
- O **armazenamento de longo prazo** inclui qualquer tabela ou partição de tabela que não tenha sido modificada por 90 dias consecutivos. O preço de armazenamento dessa tabela cai aproximadamente 50% de maneira automática. Não há diferença no desempenho, na durabilidade ou na disponibilidade entre o armazenamento ativo e o de longo prazo.

Os primeiros 10 GB de armazenamento por mês são gratuitos.

São Paulo (southamerica-east1) ▾ | Mensal

Operação	Preços	Detalhes
Armazenamento ativo	$0.023 per GB	Os primeiros 10 GB são gratuitos todos os meses.
Armazenamento de longo prazo	$0.016 per GB	Os primeiros 10 GB são gratuitos todos os meses.

Fonte: https://cloud.google.com/bigquery/pricing#storage

BigQuery - Preços

Preços de análise sob demanda

Por padrão, as consultas são cobradas usando o modelo de preços sob demanda.

Com os preços sob demanda, o BigQuery cobra pelo número de bytes processados (também chamados de bytes lidos). Você recebe cobranças por esses bytes, mesmo se os dados estiverem armazenados no BigQuery ou em uma fonte de dados externa, como o Cloud Storage, o Google Drive ou o Cloud Bigtable. Os preços das consultas sob demanda são calculados apenas conforme o uso.

O sistema de preços de consulta sob demanda é o seguinte:

São Paulo (southamerica-east1) ▾ | Mensal

Operação	Preços	Detalhes
Consultas (sob demanda)	$9.00 per TB	O primeiro terabyte (1 TB) por mês é gratuito.

Fonte: https://cloud.google.com/bigquery/pricing#on_demand_pricing

Exercício

- Considerando o volume de dados calculado no exercício anterior, estime qual seria o custo mensal de armazenamento de dados no BigQuery;

- Considere o seguinte cenário para estimar um custo de análise sob demanda:
  - Existe um mecanismo de ingestão de dados (data flow na gcp) que realiza uma carga de dados diariamente no BigQuery;
  - Por uma decisão de projeto, decidiu-se aplicar o comando CREATE OR REPLACE na tabela CLIENTE para manter todos os dados da tabela cliente atualizados no DW;
  - Para garantir que a tabela de CLIENTE fosse atualizada, quando ocorre um erro na ingestão, o fluxo repete o procedimento de carga até que o procedimento ocorra com sucesso (cada tentativa demora 1 minuto);
  - Ocorre que os dados de CLIENTE continham um erro e isso foi notado 72 horas depois;
  - Estime o custo considerando como custo de analise sob demanda;

Exercício

- ✘ Desenhe um esboço de arquitetura de hardware para um DW em um ambiente on premisse considerando o caso abaixo:

- ✘ A empresa tem 3 data marts:
  - Data mart de Vendas: volume total de 210 GB de dados armazenados em um intervalo de 3 anos. Acessado por 20 usuários com 10 consulta diárias em média;
  - Data mart de Marketing: volume total de 180 GB de dados armazenados em um intervalo de 3 anos. Acessado por 20 usuários com 2 consulta diárias em média;
  - Data mart Logística: volume total de 150GB de dados armazenados em um intervalo de 3 anos. Acessado por 20 usuários com 30 consulta diárias em média;

- ✘ Obs.: baixe a ferramenta https://www.diagrams.net/

Índices

Índices

Índices

- Os índices podem melhorar significativamente o desempenho das consultas do DW

- No DDS temos tabelas fato e dimensão

- Elas precisam de diferentes índices e chaves primárias

Índices nas dimensões

- Cada tabela dimensão tem uma coluna surrogate key (SK)
- Esta coluna deve ser um sequencial (ex. Identity(1,1)) e os valores são únicos
- Esta surrogate key deve ser a chave primária da tabela dimensão
- Além disso a SK deve ser o índice clusterizado da tabela dimensão. A razão para isso é que a ligação entre a tabela fato e dimensão é feita por essa coluna, e assim teremos um desempenho melhor.
- Um índice clusterizado é aquele que determina a ordem que as linhas são fisicamente armazenadas no disco

Índices nas dimensões

- ✘ Tabelas dimensões contém atributos que geralmente são colunas de tipo de dados caracter (char, varchar etc.);

- ✘ Colunas que são geralmente utilizadas em cláusulas WHERE das consultas devem ser definidas como um índice não-clusterizado, mas apenas se a seletividade for alta. Se tiver muitos valores duplicados, pode não compensar;

Índices nas fatos

- Descubra quais colunas fazem a linha da tabela fato ser única;
- Por exemplo, em uma tabela de vendas pode-se ter que o grão é uma linha para cada cliente por dia, assim a chave data (SK) e a chave cliente (SK) devem fazer parte da chave primária da tabela fato;
- Deve-se definir o índice clusterizado nestas duas colunas então para a tabela ficar organizada de acordo com a data e depois de acordo com o cliente;
- Uma consulta com filtro pela PK é aproximadamente 10x mais rápida do que quando indexada por uma dessas colunas (data ou cliente);

Considerações

- ✘ Não compensa indexar tabelas pequenas. O overhead gerado (manutenção dos índice durante a carga de dados) é maior que o benefício de desempenho nas consultas

- ✘ 1.000 linhas é um bom número para se basear se deve ou não indexar uma tabela. Menos que isso, geralmente não vale a pena

- ✘ Obviamente estudar o plano de execução deve ser considerado quando trabalhando com os índices

O que indexar além do básico

Dimensão

- Em um primeiro momento, crie índices baseado no conhecimento da aplicação e do negócio (pelo que se filtra mais, os joins feitos etc.)
- Depois, faça o tunning baseado nas informações de ferramentas da base de dados, como por exemplo SQL Server Profiler.

Fato

- Geralmente ajuda o desempenho de consultas que juntam as tabelas fato e dimensão se for criado um índice não-clusterizado para cada surrogate key de dimensão na tabela fato
- Esta estratégia é conhecida como index bitmapping.

Exercício

- Avalie juntamente com o seu grupo os dois artigos fornecidos, que tratam dos bancos de dados RedShift e BigTable:

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