Avaliação da Disciplina SCC5968 Processamento Analítico de Dados em Larga Escala

André Marcos Perez[1], 8006891

Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo
São Carlos, Brasil

1 Introdução

1.1 Contextualização

Avaliação da disciplina SCC5968 Processamento Analítico de Dados em Larga Escala, oferecida no segundo semestre do ano de 2019 para os alunos do programa de mestrado profissional em Matemática, Estatística e Computação Aplicadas à Indústria (MECAI) pela professora doutora Cristina Dutra de Aguiar Ciferri.

2 Materiais e Métodos

2.1 Materiais

Base de dados Os *scripts*, desenvolvidos na linguagem *Structured Query Language* (SQL), de criação e consultas da base de dados utilizada na condução das atividades deste trabalho estão presentes em [4].

Sistema de gerenciamento de base de dados O sistema de gerenciamento de base de dados (SGBD) utilizado neste trabalho foi o MySQL Server, versão 8.0.18, disponível via Docker *container* em [2]. Já o ambiente de desenvolvimento utilizado foi o MySQL Workbench, também na versão 8.0.18, disponível em [3].

2.2 Métodos

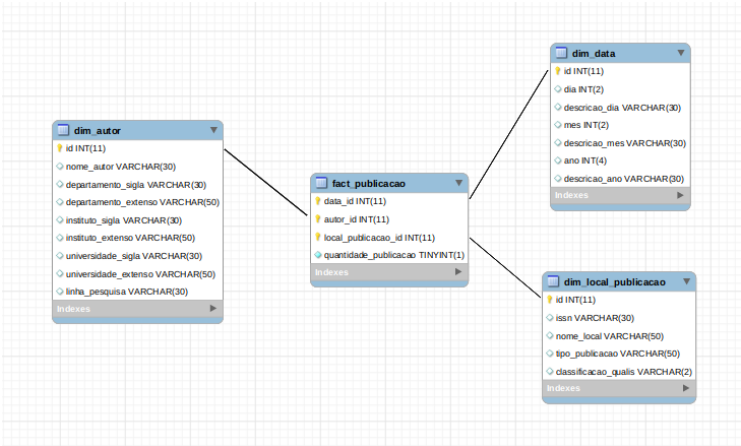
Convenções A nomenclatura das entidades esquema, tabelas (fatos e dimensões) e colunas (atributos) seguem a seguinte convenção. Esquemas utilizam o *snake_case* sem formatação, como em publicacao. Tabelas seguem o *snake_case* e são formatadas em negrito, como **dim_autor**. Por fim, colunas também seguem o *snake_case* e são formatadas em itálico, como *linha_pesquisa*.

3 Questões

3.1 Questão 1

A resolução da questão 1 utiliza o esquema estrela star_publicacao, presente na figura 1. Este é a implementação do esquema Publicação proposto na figura 1 do texto base da avaliação.

Figura 1: Implementação do esquema estrela Publicação.



Letra A O atributo *quantidade_publicacao* é adicionado na tabela de fatos **fact_publicacao** apenas para registrar a ocorrência dos fatos, uma vez que esta não possui medidas numéricas. Seu valor sempre igual a 1 permite que a ocorrência dos fatos seja eficientemente contada através da sua agregação por uma simples operação de soma, eventualmente filtrada pelos atributos presentes nas tabelas de dimensão do esquema. Dentro do contexto do esquema, o atributo *quantidade_publicacao* registra a publicação de um autor em um local em uma determinada data. Caso o fato não tenha ocorrido, nenhuma linha é adicionada ao esquema, pois sistemas relacionais de processamento analítico de dados (ROLAP), como o proposto, não armazenam dados esparsos.

Letra B A medida numérica presente no esquema estrela star_publicacao é o atributo *quantidade_publicacao*, presente na tabela de fatos **fact_publicacao**. O atributo é classificado como uma medida numérica aditiva pois pode ser agregado através da sua soma em cada uma das dimensões do esquema. Exemplos:

1. Quantidade de publicações no mês de novembro de 2019.

```
select sum(fact.quantidade_publicacao) as total_pub
from fact_publicacao fact
join dim_data data
on fact.data_id = data.id
where 1=1
and data.mes = 11
and data.ano = 2019;
```

   – Resultado: 3.

2. Quantidade de publicações do SCC-ICMC-USP.

```
select sum(fact.quantidade_publicacao) as total_pub
from fact_publicacao fact
join dim_autor autor
on fact.autor_id = autor.id
where 1=1
and autor.departamento_sigla = 'SCC'
and autor.instituto_sigla = 'ICMC'
and autor.universidade_sigla = 'USP';
```

   – Resultado: 7.

3. Quantidade de publicações com classificação Qualis A2.

```
select sum(fact.quantidade_publicacao) as total_pub
from fact_publicacao fact
join dim_local_publicacao local_publicacao
on fact.local_publicacao_id = local_publicacao.id
where local_publicacao.classificacao_qualis = 'A2';
```

   – Resultado: 4.

Letra C A tabela de fatos do esquema é a tabela **fact_publicacao** e as tabelas de dimensão são as seguintes: *dim_autor*, *dim_data* e *dim_local_publicacao*.

Letra D A tabela de fatos e as tabelas de dimensão diferem-se quanto a quantidade de atributos e dados armazenados. Seja $\mathbf{F}_{\alpha\times\beta}$ e $\mathbf{D}_{i\alpha\times\beta}$ respectivamente a tabela de fatos e a tabela da i-ésima dimensão, temos que o número de linhas α é diretamente proporcional a quantidade de dados armazenados na tabela enquanto a quantidade de colunas β é diretamente proporcional a quantidade dos seus atributos. Sendo assim, temos as seguintes comparações:

1. Comparação da quantidade de atributos β.

   Em geral, a quantidade de atributos da tabela de fatos é muito menor que a de uma tabela de dimensão. Este fato se dá pois os atributos da tabela de fatos consistem em uma chave primária composta pelas chaves primárias das tabelas de dimensão (não utiliza-se a restrição de chave estrangeira para preservar o desempenho da inserção de dados na tabela) e um conjunto pequeno de medidas numéricas ou fatos. Logo, $\beta_{\mathbf{F}}$ é igual a soma da quantidade total de tabelas de dimensão do esquema estrela (I) com a quantidade total fatos registrados (F) na tabela de fatos (equação 1).

$$\beta_{\mathbf{F}} = I + F, \quad \begin{cases} I = \text{Quant. dimensões} \\ F = \text{Quant. fatos} \end{cases} \tag{1}$$

Já uma tabela de dimensão apresenta como atributos uma chave primária e um conjunto de atributos provenientes da de-normalização de tabelas relacionais. Além disso, cada um destes atributo pode apresentar diversão versões do mesmo dado, adicionando assim redundância a tabela. Logo, o número de colunas $\beta_{\mathbf{D}_i}$ é igual a soma da quantidade de chaves primárias (unitário para tabelas de dimensão) com o fator de redundância k_ij de cada um dos seus j-ésimos atributos (equação 2).

$$\beta_{\mathbf{D}_i} = 1 + \sum_{j=1}^{J_i} k_{ij}, \quad \begin{cases} J_i = \text{Quant. atributos da i-ésima dimensão} \\ k_{ij} = \text{j-ésimo fator de redundância} \end{cases} \tag{2}$$

Como a quantidade total de atributos J_i da i-ésima tabela de dimensão tende, por si, só ser maior que a soma da quantidade de tabelas de dimensão I e a quantidade de fatos registrados F devido a sua proveniência da de-normalização de tabelas relacionais, tabelas de dimensão tendem a apresentar um número de atributos maior que tabelas de fatos. Além disso, se considerando o fator de redundância k_ij de cada atributo, essa relação tende a uma diferença ainda maior, conforme expresso na equação 3.

$$\begin{aligned} F + I &< J_i \\ F + I &\ll 1 + \sum_{j=1}^{J_i} k_{ij} \\ \beta_{\mathbf{F}} &\ll \beta_{\mathbf{D}_i} \end{aligned} \tag{3}$$

2. Comparação da quantidade de dados α.

   Em geral, a quantidade de dados armazenados da tabela de fatos é maior que a de uma tabela de dimensão qualquer. Para todo novo fato inserido no esquema estrela, uma nova tupla será adicionada na tabela de fatos $\mathbf{F}$, incrementando $\alpha_{\mathbf{F}}$. Contudo, $\alpha_{\mathbf{D}_i}$ poderá ou não ser incrementada, pois um novo fato pode utilizar uma tupla já existente em $\mathbf{D}_i$ e novas tuplas inseridas em $\mathbf{D}_j$ qualquer para compor a chave primária de $\mathbf{F}$. De fato, no melhor caso, uma tabela de dimensão $\mathbf{D}_i$ qualquer pode apresentar o mesmo número de linhas da tabela de dimensão $\mathbf{F}$. Já no pior caso, uma tabela de dimensão $\mathbf{D}_i$ qualquer pode esperar por diversas inserções de tuplas na tabela de fatos sem que nenhuma inserção seja feita em si mesma. Sendo assim, no caso médio, a relação entre a quantidade de dados armazenados na tabela de fatos $\alpha_{\mathbf{F}}$ em relação a quantidade em de uma tabela de dimensão $\alpha_{\mathbf{D}_i}$ qualquer segue a relação exposta na equação 4.

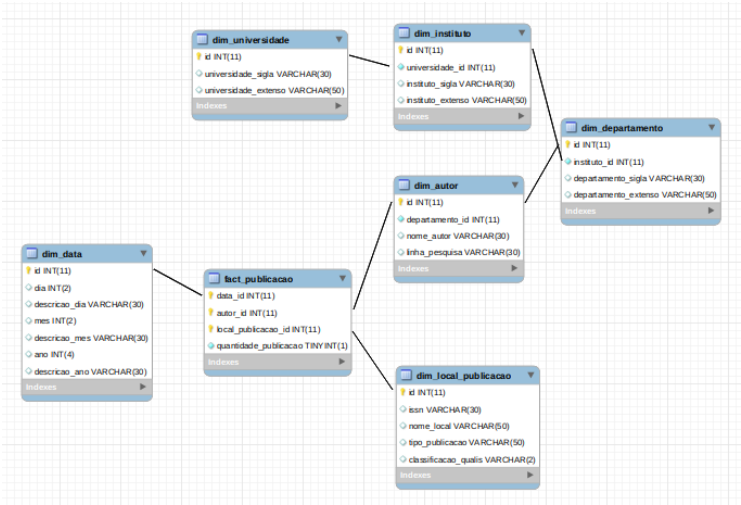
$$\alpha_{\mathbf{F}} \gg \alpha_{\mathbf{D}_i} \tag{4}$$

Letra E O esquema estrela é projetado para armazenar dados redundantes em suas tabelas de dimensão para que consultas analíticas *ad hoc* possam ser feitas sem que nenhum processamento intermediário de dados seja necessário, acelerando assim a velocidade e a integridade da apresentação do resultado para o usuário final. Um exemplo de redundância presente no esquema estrela star_-publicacao são os atributos *universidade_sigla* e *universidade_extenso* da tabela de dimensão **autor** pois ambos representam a entidade universidade de maneiras diferentes mas complementares, uma vez que tanto a sigla quanto o nome por extenso de uma universidade são amplamente utilizados.

Letra F A figura 2 apresenta o esquema floco de neve snowflake_publicacao proposto. Em contraste com o esquema estrela star_publicacao, o esquema apresenta a normalização da tabela de dimensão **dim_autor** em outras três tabelas de dimensão: **dim_departamento**, **dim_instituto** e **dim_universidade**. O critério utilizado para sua construção foi a redução da redundância da tabela de dimensão **dim_autor** através da exploração da hierarquia natural inerente das entidades em questão, conforme equação 5.

$$\text{universidade} \prec \text{instituto} \prec \text{departamento} \prec \text{autor} \tag{5}$$

Figura 2: Implementação do esquema floco de neve Publicação.



Letra G

1. O esquema estrela é mais eficiente que o esquema floco de neve no processamento de consultas que utilizam os atributos das dimensões normalizada

no esquemas floco de neve. Essa fato se dá pois enquanto o esquema estrela precisa resolver apenas a junção estrela com a tabela de fatos, o esquema floco de neve precisa adicionalmente resolver a junção com as suas dimensões normalizadas. Exemplo:

(a) star_publicacao: Quantidade de publicações por universidade.

```
select sum(fact.quantidade_publicacao) as total_pub
from fact_publicacao fact
join dim_autor autor
on fact.autor_id = autor.id
where autor.universidade_sigla = 'USP';
```

- Quantidade de clausulas *join*: 1.

(b) snowflake_publicacao: Quantidade de publicações por universidade.

```
select sum(fact.quantidade_publicacao) as total_pub
from fact_publicacao fact
join dim_autor autor
on fact.autor_id = autor.id
join dim_departamento departamento
on autor.departamento_id = departamento.id
join dim_instituto instituto
on departamento.instituto_id = instituto.id
join dim_universidade universidade
on instituto.universidade_id = universidade.id
where universidade.universidade_sigla = 'USP';
```

- Quantidade de clausulas *join*: 4.

2. O esquema floco de neve é mais eficiente que o esquema estrela no processamento de consultas que não utilizam os atributos das dimensões normalizadas no esquema floco de neve. Esse fato se dá pois a clausula *select* precisa selecionar menos colunas para retornar o resultado ao usuário final. Exemplo:

(a) star_publicacao: Quantidade de publicações por linha de pesquisa.

```
select autor.linha_pesquisa,
sum(fact.quantidade_publicacao) as total_pub
from fact_publicacao fact join dim_autor autor
on fact.autor_id = autor.id
group by 1;
```

- Quantidade de colunas descartadas: 8.

(b) snowflake_publicacao: Quantidade de publicações por linha de pesquisa.

```
select autor.linha_pesquisa,
```

```
sum(fact.quantidade_publicacao) as total_pub
from fact_publicacao fact join dim_autor autor
on fact.autor_id = autor.id
group by 1;
```

- Quantidade de colunas descartadas: 3.

Letra H

1. **slice**: Reduz o conjunto de dados analisados a um subconjunto destes utilizando como restrição um valor único para cada atributo selecionado de uma ou mais dimensões. No nível conceitual, representa a extração de uma única fatia de uma ou mais dimensões do hipercubo de dados. A operação é equivalente a clausula SQL *where* com uma ou mais condições de igualdade separadas pelo operador *and*. Exemplo:

   - Liste a quantidade de publicações por autor em 11/19.

   ```
   select autor.nome_autor,
   sum(fact.quantidade_publicacao) as total_pub
   from fact_publicacao fact
   join dim_autor autor on fact.autor_id = autor.id
   join dim_data data on fact.data_id = data.id
   where 1=1
   and data.mes = 11
   and data.ano = 2019
   group by 1;
   ```

2. **dice**: Reduz o conjunto de dados analisados a um subconjunto destes utilizando como restrição uma faixa de valores para cada atributo selecionado de uma ou mais dimensões. No nível conceitual, representa a extração de múltiplas fatias de uma ou mais dimensões do hipercubo de dados. A operação é equivalente a clausula SQL *where* com uma ou mais condições de igualdade separadas pelo operador *or* ou por operadores de conjuntos como *between* e *in*. Exemplo:

   - Liste a quantidade de publicações por autor entre 06/19 e 12/19.

   ```
   select autor.nome_autor,
   sum(fact.quantidade_publicacao) as total_pub
   from fact_publicacao fact
   join dim_autor autor on fact.autor_id = autor.id
   join dim_data data on fact.data_id = data.id
   where 1=1
   and data.mes between 6 and 12
   and data.ano = 2019
   group by 1;
   ```

3. **roll-up**: Reduz o conjunto de dados analisados a um subconjunto destes em um nível de agregação de maior granularidade. No nível conceitual, representa a transformação das fatias de uma ou mais dimensão do hipercubo de dados em fatias maiores, mais agregadas, portanto menos detalhadas. A operação é equivalente a uma combinação das clausulas SQL *select* e *group by* onde um conjunto menor de atributos de uma ou mais dimensões são selecionados e agregados. Exemplo:

   – Liste a quantidade de publicações por instituto por universidade por ano.

```
select autor.instituto_sigla,
autor.universidade_sigla, data.ano,
sum(fact.quantidade_publicacao) as total_pub
from fact_publicacao fact
join dim_autor autor on fact.autor_id = autor.id
join dim_data data  on fact.data_id = data.id
group by 1, 2, 3;
```

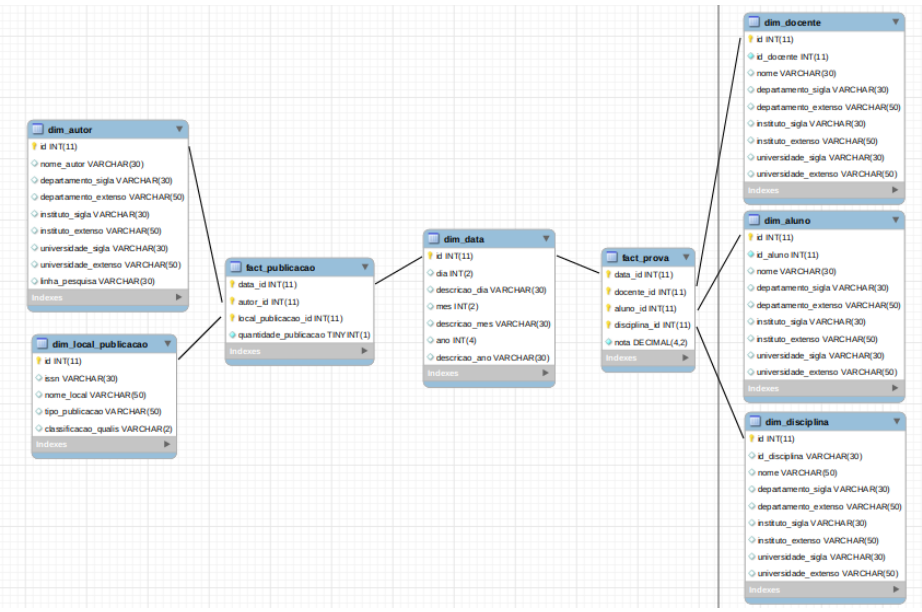
4. **drill-down**: Reduz o conjunto de dados analisados a um subconjunto destes em um nível de agregação de menor granularidade. No nível conceitual, representa a transformação das fatias de uma ou mais dimensão do hipercubo de dados em fatias menores, menos agregadas, portanto mais detalhadas. A operação é equivalente a uma combinação das clausulas SQL *select* e *group by* onde um conjunto maior de atributos de uma ou mais dimensões são selecionados e agregados. Exemplo:

   – Liste a quantidade de publicações por departamento por instituto por universidade por ano.

```
select autor.departamento_sigla,
autor.instituto_sigla,
autor.universidade_sigla, data.ano,
sum(fact.quantidade_publicacao) as total_pub
from fact_publicacao fact
join dim_autor autor on fact.autor_id = autor.id
join dim_data data  on fact.data_id = data.id
group by 1, 2, 3, 4;
```

Letra I A figura 3 apresenta o esquema da constelação de fatos constellation_-publicacao-prova criado através do compartilhamento da dimensão **dim_data** entre os esquemas estrelas star_publicacao e star_prova.

Figura 3: Implementação do constelação de fatos constellation_publicacao-prova.



Letra J

1. **drill-across**: Combina dois ou mais subconjuntos de dados de esquemas diferentes através de uma dimensão em comum. No nível conceitual, representa a combinação de fatias de hipercubos diferentes, mas que compartilham ao menos uma dimensão em comum. A operação é equivalente a clausula SQL *join* entre tabelas derivadas de esquemas diferentes, obtidas através de *sub-queries*, mas que possuem ao menos uma tabela de dimensão em comum. Exemplo:

   – Liste a nota média que um docente atribui as seus alunos por mês por quantidade de publicação no ano de 2019.

```
select prova.docente, prova.mes,
prova.nota_media, publicacao.total_pub
from (
    select autor.nome_autor as autor, data.mes,
    sum(fact.quantidade_publicacao) as total_pub
    from fact_publicacao fact
    join dim_autor autor
    on fact.autor_id = autor.id
    join dim_data data
```

```
    on fact.data_id = data.id
    where data.ano = 2019
    group by 1, 2
) publicacao
join (
    select docente.nome as docente, data.mes,
    avg(fact.nota) as nota_media
    from fact_prova fact
    join dim_docente docente
    on fact.docente_id = docente.id
    join dim_data data
    on fact.data_id = data.id
    where data.ano = 2019
    group by 1, 2
) prova
on 1=1
and publicacao.mes = prova.mes
and publicacao.autor = prova.docente;
```

Letra K De acordo com [1], o fluxo lógico de processamento de uma consulta SQL é realizado na ordem descrita na tabela 1, onde algumas operações foram suprimidas por simplicidade. Na tabela, nota-se a precedência da clausula de junção *join* em relação as clausulas de filtragem *where*, agrupamento *group by* e seleção *select*. Logo, a primeira operação realizada pela junção estrela é junção entre a tabela de fatos com as tabelas de dimensão especificadas, seguidas de operações filtragem, agrupamento e seleção de atributos. Como a tabela de fatos naturalmente apresenta um número muito elevado de linhas, sua junção com um tabela de dimensão produzirá uma tabela com um número de linhas ainda maior (produto cartesiano) que posteriormente será filtrado pela restrição imposta pela clausula *on*. Concluí-se então que a junção estrela em um esquema dimensional é caro devido ao inerente elevado número de linhas de suas tabelas de fatos.

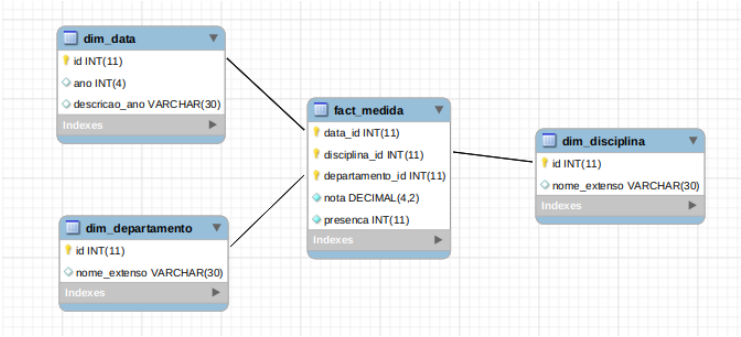
Tabela 1: Fluxo lógico de processamento de uma consulta SQL.

Precedência	**Operação**
1	*from*
2	*on*
3	*join*
4	*where*
5	*group by*
6	*select*

3.2 Questão 2

A resolução da questão 2 utiliza o esquema estrela star_medida, presente na figura 4. Este é a implementação do esquema Medida proposto na figura 2 do texto base da avaliação.

Figura 4: Implementação do esquema estrela Medida.



Letra A Sejam os vetores **computacao**, **2003** e **2004** os vetores de índices de *bitmap* da tabela de fatos **medidas** referentes ao domínio ativo computação do atributo *nome_departamento* da tabela de dimensão **dim_departamento**, domínio ativo 2003 do atributo *ano* da tabela de dimensão **dim_data** e domínio ativo 2004 do atributo *ano* da tabela de dimensão **dim_data**, respectivamente, obtém-se a resolução da consulta através das operações presentes na equação 6.

$$\textbf{computacao} \text{ AND } (\textbf{2003} \text{ OR } \textbf{2004}) \tag{6}$$

A aplicação destas operações retorna as tuplas da tabela de fatos **medida** presentes na tabela 2. Por fim, calcula-se a média das valores do atributo *nota*, obtendo-se o valor aproximado de 5.33.

Tabela 2: Tuplas da tabela de fatos **medida** retornadas após a aplicação das operação lógicas da equação 6 a seus índices de *bitmap*.

chave_data	chave_disciplina	chave_departamento	nota	presença
3	1	1	3	30
4	3	1	9	100
4	2	1	4	30

Segue a consulta SQL equivalente:

– Qual a média das notas para as disciplinas ministradas pelo departamento de nome computação nos anos de 2003 e 2004?

```
select avg(fact.nota) as nota_media
from fact_medida fact
join dim_data data
on fact.data_id = data.id
join dim_departamento departamento
on fact.departamento_id = departamento.id
where 1=1
and departamento.nome_extenso = 'computacao'
and (data.ano = 2003 or data.ano = 2004);
```

- Resultado: 5.333.

Letra B Índices de *bitmap* são amplamente utilizados pois evitam a computação das junções da junção-estrela nos cálculos de agregações das medidas numéricas dos atributos da tabela de fatos. Sem o uso dos índices, as junções são necessárias pois os filtros das tuplas da tabela de fatos (clausula *where*) atuam exclusivamente nos atributos das tabelas de dimensão. Já com os índices, as tuplas são filtradas por operações lógicas bit-a-bit na própria tabela de fatos, operações estas realizadas com alto desempenho por qualquer sistema computacional moderno, evitando-se assim o calculo das junções. Caso a computação da junção-estrela seja estritamente necessária, como em situações na qual se deseja retornar os valores dos atributos das tabelas de dimensão junto com as agregações dos fatos, os índices de *bitmap* reduzem seu custo computacional pois seu efeito na redução do número de tuplas da tabela de fatos faz com que o produto cartesiano da tabela com as tabelas de dimensão seja muito menor, além de não haver mais a necessidade de filtros posteriores.

Letra C Visões materializadas são tabelas físicas que armazenam dados derivados de outras tabelas, como agregações de atributos resultantes da resolução de junções-estrela, por exemplo. As visões materializadas são amplamente utilizadas pois aumentam o desempenho de consultas analíticas uma vez que os resultados das agregações já foram previamente computados e armazenados, evitando-se assim a computação da junção-estrela no momento da consulta. Visões materializadas são especialmente úteis em situações em que uma mesma consulta analítica é realizada com frequência.

Letra D

1. **SGBD Oracle**: A visão materializada proposta registra a nota média por disciplina por ano. Para tanto, realiza-se a junção-estrela da tabela de fatos **fact_medida** com as tabelas de dimensão **dim_data** e **dim_disciplina** e agrega-se a medida numérica não aditiva do atributo nota da tabela de fatos através do operador *avg*.

- Criação da visão materializada utilizando o SGBD Oracle.

```
create materialized view
mv_nota_media_por_discplina_por_ano
build immediate
refresh on demand
as select disciplina.nome_extenso as disciplina,
data.ano, avg(fact.nota) as nota_media
from fact_medida fact
join dim_data data
on fact.data_id = data.id
join dim_disciplina disciplina
on fact.disciplina_id = disciplina.id
group by 1, 2;
```

2. **SGBD MySQL**: Como o SGBD utilizado não possui suporte nativo para criação de visões materializadas, segue uma proposta que busca replicar a funcionalidade através do uso de gatilhos e tabelas lógicas. Nesta proposta, a tabela lógica da visão materializada será atualizada a cada novo *commit* na tabela de fatos **fact_medida**. Essa característica é equivalente ao uso da clausula *refresh on commit* ao invés da clausula *refresh on demand* utilizada na proposta de visão materializada original.

   - Proposta de criação da visão materializada utilizando o SGBD MySQL.

```
# - Visao materializada logica
create table mv_nota_media_por_disciplina_por_ano (
    id int not null auto increment,
    disciplina varchar(30),
    ano int(4),
    nota_media decimal(4, 2) not null default 0,
    constraint
    pk_mv_nota_media_por_disciplina_por_ano
    primary key (id)
);

# - Gatilho
delimiter //
create trigger
trigger_update_mv_nota_media_por_disciplina_por_ano
after insert on fact_medida
for each row begin
        set @disciplina = (
        select distinct nome_extenso
        from dim_disciplina
        where id = new.disciplina_id
    );
```

```
    set @ano = (
        select distinct ano
        from dim_data
        where id = new.data_id
    );
    set @nota_media = (
        select avg(nota)
        from fact_medida
        where 1=1
        and data_id = new.data_id
        and disciplina_id = new.disciplina_id
    );
        if exists(
        select 1
        from mv_nota_media_por_disciplina_por_ano
        where 1=1
        and disciplina = @disciplina
        and ano = @ano
    ) = true
    then
        update
        mv_nota_media_por_disciplina_por_ano
        set nota_media = @nota_media
        where 1=1
        and disciplina = @disciplina
        and ano = @ano;
        else
        insert into
        mv_nota_media_por_disciplina_por_ano
        (disciplina, ano, nota_media)
        values
        (@disciplina, @ano, @nota_media);
    end if;
end;//
delimiter ;
```

Referências Bibliográficas

[1] Microsoft, C.: Select (transact-sql). https://docs.microsoft.com/en-us/sql/t-sql/queries/select-transact-sql?redirectedfrom=MSDN&view=sql-server-ver15#logical-processing-order-of-the-select-statement, acessado: 2019-18-12

[2] Oracle, C.: Mysql official docker image. https://hub.docker.com/_/mysql, acessado: 2019-11-12

[3] Oracle, C.: Mysql workbench 8.0.18. https://dev.mysql.com/downloads/workbench/, acessado: 2019-11-12

[4] Perez, A.M.: Scc5968 large scale analytical data processing github repository. https://github.com/andreMarcosPerez/SCC5968-OLAP, acessado: 2019-11-12