**2 - Restrições de Integridade­­­**

**(RI-1) Uma Categoria não pode estar contida em si própria**

Nesta restrição de integridade, considerámos que uma categoria não poder estar contida em si própria implica que, quando inserida uma entrada na relação tem\_outra, deve existir garantia que os nomes da super categoria e da categoria simpes sejam diferentes.

Uma vez que esta restrição de integridade consiste apenas em verificar que 2 atributos inseridos numa relação são diferentes, esta poderia ser implementada com uma instrução “check” na criação da base de dados. No entanto, por uma questão de consistência com as soluções utilizadas nas restrições seguintes, optámos por utilizar um trigger.

**(RI-4) O número de unidades repostas num Evento de Reposição não pode exceder o número de unidades especificado no planograma**

Para implementarmos esta restrição de integridade, utilizámos um trigger. A solução consistiu em: procurar o planograma correspondente ao evento de reposição (o planograma que continha os mesmos ean, nro, num\_serie e fabricante), obter o número de unidades especificado no referido planograma e comparar o número obtido com o número de unidades contido no evento de reposição.

**(RI-5) Um Produto só pode ser reposto numa Prateleira que apresente (pelo menos) uma das Categorias desse Produto**

Na implementação desta restrição considerámos que, para a prateleira apresentar a categoria do produto, tem obrigatoriamente de apresentar a categoria diretamente associada ao produto na relação tem\_categoria, não podendo o produto ser reposto se a categoria exposta na prateleira for uma super categoria da categoria do produto.

Tendo o referido anteriormente em consideração, a nossa solução passou, mais uma vez, por um trigger com o seguinte comportamento: seleciona a categoria da praleira referida no planograma, obtém todos os eans que pertençam a essa categoria e verifica se o ean do planograma está contido nessa lista de eans.

**4 – Vistas**

create view vendas(ean,cat,ano,trimestre,mes,dia\_mes,

                dia\_semana,distrito,concelho,unidades) as

    select t.ean,t.cat,

            EXTRACT(YEAR from e.instante),

            EXTRACT(QUARTER from e.instante),

            EXTRACT(MONTH from e.instante),

            EXTRACT(DAY from e.instante),

            EXTRACT(DOW from e.instante),

            p.distrito,p.concelho,e.unidades

    from ponto\_de\_retalho as p

    INNER JOIN instalada\_em as i on i.nome=p.nome

    INNER JOIN evento\_reposicao as e on (e.num\_serie=i.num\_serie and

                                        e.fabricante LIKE i.fabricante)

    INNER JOIN tem\_categoria as t on (t.ean=e.ean)

Na solução apresentada optámos por utilizar INNER JOIN em vez de NATURAL JOIN, pois devido ao elevado número de tabelas e atributos envolvidos poderia existir um comportamento anómalo e a vista obtida não ser a pretendida.

**6 – Consultas OLAP**

1 – Na primeira consulta, apresentamos o número de vendas entre 2 datas (no caso, uma vez que não eram fornecidas datas, considerámos entre os anos 2016 e 2022), por dia da semana, por concelho e no total, ou seja, fizemos uma contagem para cada atributo pretendido isoladamente (Fig. 1). Com o objetivo descrito, foram utilizados os seguintes grouping sets:

GROUPING SETS ((dia\_semana), (concelho), ())

2 – Na segunda consulta, apresentamos o número de vendas num determinado distrito, por concelho, categoria,dia da semana e no total, ou seja, fizemos uma contagem para cada tuplo (concelho,categoria,dia da semana) e outra para o total (Fig. 2). Foram, assim, utilizados os seguintes grouping sets:

GROUPING SETS ((concelho, cat, dia\_semana), ())

|  |  |  |  |
| --- | --- | --- | --- |
| Dia da Semana | Concelho | Categoria | COUNT |
| 0 | Lisboa | Pão | 3 |
| 0 | Porto | Pão | 4 |
| 1 | Lisboa | Pão | 4 |
| ­­­1 | Porto | Pão | 3 |
| null | null | null | 7 |

­

|  |  |  |
| --- | --- | --- |
| Dia da Semana | Concelho | COUNT |
| 0 | null | 3 |
| 1 | null | 4 |
| null | Lisboa | 4 |
| null | Porto | 3 |
| null | null | 7 |

Fig.2

Fig.1

**3 – SQL**

**1 – Qual o nome do retalhista (ou retalhistas) responsáveis pela reposição do maior número de categorias?**

Nesta situação, considerámos que era pretendido o nome dos retalhistas com maior número de participações diretas na relação “responsavel\_por”, não sendo contabilizadas as sub categorias das possíveis super categorias da sua responsabilidade.

**2 – Qual o nome do ou dos retalhistas que são responsáveis por todas as categorias simpels?**

Neste caso, considerámos que para um retalhista ser responsável por todas as categorias simples tinha que, para cada categoria simples existente na base de dados, existir uma entrada na tabela “responsavel\_por” que associasse essa categoria ao retalhista.

**3 – Quais os produtos (ean) que nunca foram repostos?**

Nesta query, selecionámos todos os produtos cujos eans não constavam em nenhum evento de reposição.