**6. Database Tuning**

6.1. Através da análise das *queries* apresentadas, conclui-se que a *query* Q2 corresponde a um padrão típico de acesso a uma aplicação OLAP, uma vez que estas aplicações se caracterizam por uma grande capacidade de analisar e manipular uma grande quantidade de informação, através de várias perspetivas, muitas vezes utilizada para fins estatísticos. Desta forma, são analisados vários cenários, disponibilizando dados contidos em bases de dados operacionais, como *Data Warehouses*, identificados por dois ou mais atributos. Desta forma, na query Q2, observa-se uma agregação de vários tipos de queijo (agrupados por tipo através do GROUP BY), retornando-se assim o número de queijo de cada tipo (através do COUNT(\*)). Trata-se por tipo, de um padrão observável nas aplicações OLAP. De realçar que as *querys* que interagem com este tipo de aplicações são tipicamente de leitura, como é o caso da *query* Q2.

Por outro lado, a query Q1 corresponde a um padrão típico de acesso a uma aplicação OLTP, pois estas são caracterizadas por transações curtas, em que o processamento da *query* é rápido. Têm normalmente como objetivo o registo de todas transações de uma determinada organização. Consequentemente, a *query* Q1 corresponde a uma interrogação relativamente rápida, não apresentando nenhum tipo de agregação (ao contrário da query Q2). É, por isso, um exemplo de acesso típico a uma aplicação OLTP.

6.2. Sabendo que a *query* Q2 corresponde a um acesso típico a uma aplicação OLAP, e que estas aplicações se caracterizam por apenas ser permitido a inserção e leitura de dados (sendo que, para um utilizador, apenas está disponível a leitura), faz mais sentido utilizar um nível de isolamento baixo (por exemplo, READ UNCOMMITED), uma vez que a probabilidade de existir um conflito entre transações é relativamente pequeno.

Por outro lado, relativamente à *query* Q1 que corresponde a um acesso a uma aplicação OLTP, e sabendo que estas aplicações permitem acessos de leitura, inserção, modificação e exclusão de dados muito frequentemente, faz sentido usar um nível de isolamento mais elevado (por exemplo, do tipo SERIALIZABLE), uma vez que a probabilidade de existirem conflitos entre transações é bastante elevado (devido à constante modificação dos dados).

6.3. Relativamente à *query* Q1, duas otimizações que podiam ser consideradas seriam:

* **Do tipo index tuning**: Criando um índice *clustering* (preferencialmente do tipo *hash*, mas um do tipo *BTree* seria também vantajoso) sobre o atributo de junção do *join* faria com que os dados estivessem fisicamente ordenados tendo em conta esse atributo. Dessa forma, quando fosse executada a interrogação, o processamento da mesma seria muito mais rápido.
* **Do tipo query tuning**: Como a cláusula FROM da *query* Q1 será executada antes da cláusula WHERE, e como nesse cláusula (FROM) estão presentes dois *joins*, a tabela resultante seria de dimensões bastante elevadas, tendo em conta que nessa tabela apenas serão selecionados alguns tuplos tendo em conta a cláusula WHERE. Assim, a solução seria inserir em duas tabelas temporárias os dados filtrados (uma para a tabela REGION filtrada com os tuplos que têm apenas *country=’Portugal’* e outra para PRODUCTION filtrada com os tuplos que têm *amount>10,000* e *season=’Winter’*). Desta forma, não existiria uma tabela que contivesse todos os dados de todas as tabelas, mas sim os dados previamente filtrados adequadamente. Ou seja, a solução seria:

|  |
| --- |
| INSERT INTO TEMPR  SELECT \*  FROM REGION  WHERE country = 'Portugal'  INSERT INTO TEMPP  SELECT \*  FROM PRODUCTION  WHERE amount > 10000 AND season = 'Winter'  SELECT type, producer  FROM CHEESE NATURAL JOIN TEMPP NATURAL JOIN PROVENANCE NATURAL JOIN TEMPR |