# Recensão sobre "A Relational Model of Data for Large Shared Data Banks"

Pedro Oliveira nº 35480
Raul Oliveira nº 36240

Abril 2018

CODD, E. F. (1970). A Relacional Model of Data for Large Shared Data Banks. *communications of the ACM*, **13.6:** 377-387.

Edgar Frank "Ted" Codd nasceu no dia 19 de Agosto de 1923, em Inglaterra. Estudou matemática e química em Oxford, e fez o doutoramento em Engenharia Informática na universidade de Michigan. Trabalhou para a IBM até aos anos oitenta. A sua saúde começou a deteriorar-se, o que fez com que parasse de trabalhar. Morreu a 18 de Abril de 2003 de insuficiência cardíaca. Ganhou o Turing Award em 1981. Apesar de ter tido mais trabalhos famosos, este é referenciado como o seu trabalho mais importante.

"A Relational Model of Data for Large Shared Data Banks" será, então, o objeto de análise. O foco deste trabalho é o uso de modelos relacionais de dados para melhorar a performance e simplificar o uso das bases de dados.

O artigo encontra-se dividido em duas grandes partes. A primeira trata do modelo de dados relacional e a forma normal. Na primeira parte são ditas as vantagens do modelo relacional em relação às praticas usadas na época, nomeadamente grafos e o "network model". Algumas destas vantagens são a representação dos dados na sua estrutura original e proporcionar uma base sólida para o tratamento da derivabilidade, redundância e consistência das relações. O modelo relacional permite também uma avaliação mais clara das limitações lógicas dos sistemas em vigor. De seguida o autor debruça-se sobre tabelas descritivas de dados, defendendo que estas apresentam um grande avanço para a independência de dados, facilitando a mudança de certas características, e simplificando o que o utilizador vê.

Segundo o autor, existem três tipos de dependências que ainda devem ser removidas: dependência por ordenação, dependência por indexação e dependência por acesso a passagem.

1. Dependência por Ordenação: Falha, caso seja necessário mudar a ordenação em vigor.

2.Dependência por Indexação: Normalmente usado para melhorar a performance da representação dos dados, este tipo de dependência falha, pois existem várias abordagens ao tipo de indexação, nomeadamente a todos os atributos, à chave primária, e através de "chains". Nos dois primeiros casos, a lógica das aplicações depende dos

índices dados. No último, basta a remoção de uma dessas "chains" para que a lógica falhe.

3. Dependência por Acesso a Passagem: São baseados em árvores ou modelos de redes de dados mais generalizados. Caso as árvores ou as redes sejam minimamente alterados, esta dependência falha.

Depois de enumerar e explicar estas dependências, o autor defende o uso do ponto de vista relacional, mostrando que é organizado e matematicamente lógico, representado por arrays, onde cada array é distinto, as posições dos arrays importam, e cada uma deve ter um nome descritivo. Defende também que os utilizadores não devem ser obrigado a lembrar-se da ordenação do domínio de nenhuma relação, propõe até que se deixem de usar relações e se passem a usar relacionamentos, que são os seus domínios sem organização. Para isto os domínios devem ser unicamente identificáveis, pelo menos dentro dessa mesma relação, sem usar as posições. Sendo assim, os utilizadores não precisam de saber mais de cada relação do que os seus nomes. Esta informação deve ser disponibilizada através de menus, ao pedido do utilizador. Cada domínio deve ter valores que identifiquem unicamente cada elemento dessa relação, chamando-se **chave primária**. Chama então **chave estrangeira** a um elemento do domínio que não é chave primária dessa relação, mas sim de outra.

Outro ponto crucial é a Forma Normal. Obtém-se através de um simples processo de eliminação chamado normalização. Para se realizar, devem ser satisfeitas as seguintes condições:

1. O grafo de relações dos domínios não simples deve ser uma árvore.

2. Nenhuma chave primária tem um componente do domínio que é não simples.

Se todas as relações estiverem na forma normal, não há só uma vantagem no espaço ocupado pela memória, mas também na comunicação em grande escala de sistemas com representações de dados muito diferentes.

Isto proporcionava então três vantagens: não haveria apontadores, deixaria de haver dependências e deixaria de haver indexação, ou listas ordenadas.

Tudo isto culmina na criação de uma sub linguagem universal baseada em predicados de cálculo, que proporcionaria um maior poder a todas as outras linguagens de dados. A abrangência universal desta sub linguagem recai sobre a sua habilidade descritiva. Esta linguagem permitiria uma forma de nomear variáveis mais livre, pois os modelos em vigor na época muitas vezes obrigavam os utilizadores a saber e usar mais nomes de relações do que o estritamente necessário.

Associado a um banco de dados estão duas coleções de relações: as de nomes, e as de expressões. As relações de nomes são as que se identificam com a utilização de nomes, enquanto que as relações de expressões são identificadas utilizando expressões da linguagem de dados. A relação de nomes é um subconjunto da relação de expressões.

A segunda grande parte deste trabalho fala dos problemas de redundância e consistência nos modelos de dados em vigor na época.

O primeiro ponto abordado é o das operações sobre as relações. O autor refere que como as relações são conjuntos, todas as operações são-lhes aplicáveis, no entanto o resultado nem sempre é uma relação. Refere de seguida as operações que se podem fazer, explicando-as. Essas operações são: Permutações, Projeções, Joins, Composições e Restrições. Depois de apresentar vários exemplos de cada uma destas operações, o

autor foca-se no que é a redundância. Explica então que redundância redundância nas relações de nomes é diferente das redundâncias nas relações guardadas, mas que neste artigo apenas irá comentar apenas as redundâncias nas relações guardadas.

Segundo o autor existem redundâncias fortes e redundâncias fracas. Uma redundância forte é quando uma relação contém uma projeção que é derivável de de outras projeções de relações no conjunto. Uma razão para a existência destas redundâncias é a conveniência para o utilizador.

Uma redundância fraca será quando uma relação tem uma projeção que não é derivável de outros membros, mas que é uma projeção de um qualquer join de outras projeções de relações nesta coleção. Nem o sistema, nem o DBA as conseguem remover.

De seguida o tema muda para consistência. Consistência é uma propriedade instantânea do banco de dados e é independente do como esse estado passou a existir. Existem várias formas do sistema detetar inconsistências e corrigi-las.

Este é então o último ponto abordado pelo autor.

Este foi um artigo que posteriormente revolucionou a maneira como se fazem as bases de dados, criando o modelo de dados relacional, removendo do mercado outras representações de dados obsoletas, e oferecendo algo conciso e fácil de entender e trabalhar. Apesar de ter sido publicado em 1970, continua tão atual que devia ser referência para todas as pessoas que operam bases de dados.