**Anexo 1**

**As 12 Regras de Codd**

Uma base de dados caracterizada como base de dados relacional (RDB) pode ser abordada de uma maneira simplista como uma base de dados que adota um modelo de dados identificado como relacional, como foi proposto em [Codd 1970] e abordado no Capítulo 2 deste documento.

O termo base de dados relacional foi definido por E.F.Codd em 1970 no artigo intitulado ["A Relational Model of Data for Large Shared Data Banks"](https://doi.org/10.1145%2F362384.362685) (ver referências [Codd 1970] [Codd 1985a] [Codd 1985b] [Wikipedia 2024] [Wikipedia 2024a]). A definição de uma estrutura de dados que pode ser qualificada como base de dados relacional sob a ótica de Codd é caracterizada pela satisfatibilidade das chamadas 12 regras de Codd.

Como informado em [Wikipedia-2024a], Codd teve como principal objetivo, ao criar o conjunto de regras, evitar que o conceito original de RDB fosse adulterado à medida que fornecedores de DBs tentavam promover a comercialização das bases de dados já existentes, vendendo-as como relacional. As 12 regras podem ser consideradas como parte de um procedimento de teste para determinar se um determinado produto, que se autoproclama ser completamente relacional, é realmente completamente relacional. Várias implementações comerciais do modelo relacional de base de dados, entretanto, não seguem todas as regras de Codd − como consequência, o termo tem sido gradualmente usado para descrever um vasto grupo de sistemas de DB que, no mínimo:

(1) Apresentam dados ao usuário que são representados como relações em uma representação em formato tabular *i.e.*, como uma coleção de tabelas, cada uma consistindo de um conjunto de linhas e de colunas.

(2) Fornecem operadores relacionais para manipular os dados em formato tabular.

Presentemente a definição mais comum de um RDBMS (Relational Data Base Management System) é a de ser um produto que apresenta uma visão dos dados como uma coleção de linhas e colunas, mesmo que não seja estritamente baseada na teoria relacional. Considerando essa definição, RDBMSs tipicamente implementam algumas mas não todas as 12 regras.

Codd em [Codd 1985a, 1985b] procura agrupar algumas das 12 regras, comentando que (a) as regras 8, 9,10 e 11 especificam e requerem quatro tipos diferentes de independência, com o objetivo de proteger o investimento de clientes em programas de aplicação, atividades em terminais e treinamento; (b) as regras 8 e 9 que estão relacionadas à independência física e lógica respectivamente, têm sido discutidas por muitos anos; (c) as regras 10 e 11 dizem respeito à independência de integridade e independência de distribuição, que são aspectos da abordagem relacional que receberam atenção inadequada durante os anos iniciais em que foram propostas. Codd, entretanto, previu que as regras 10 e 11 se tornariam tão importantes quanto as regras 8 e 9. O autor também informou que as regras propostas foram baseadas em uma única regra fundamental, que ele nomeou Regra Zero. Na sequência são apresentadas e discutidas as 13 regras como estabelecidas em [Codd 1985a, 1985b].

**Regra 0 Regra Zero**

*Qualquer sistema que é veiculado ou proclamado ser um sistema de administração de bases de dados relacional deve ser capaz de administrar bases de dados inteiramente por meio de suas capacidades relacionais.*

A Regra Zero deve ser satisfeita independentemente de o sistema lidar ou não com qualquer característica não-relacional de administrar dados. Qualquer RDBMS que não satisfizer a Regra 0 não pode ser caracterizado como um RDBMS. Possíveis consequências da Regra Zero:

(a) qualquer sistema divulgado como um RDBMS deve poder realizar inserções, atualizações e deleções na DB em nível relacional *i.e.*, *multiple-record-at-a-time* (múltiplos registros de uma vez). A execução de atividades via expressão *multiple-record-at-a-time* inclui, como casos especiais, situações em que nenhum ou apenas um registro é recuperado, inserido, atualizado ou deletado. A relação (tabela) que será discutida oportunamente pode ter 0 (zero) linhas ou uma linha e ainda assim ser uma relação válida.

(b) a necessidade de suporte à regra de informação (Regra 1) e regra de acesso garantido (Regra 2).

Como comentado anteriormente, Codd em [Codd 1985a] discute o perigo de compradores e usuários de um sistema que é comercializado como relacional e que não satisfaz a Regra 0: compradores e usuários estarão esperando pelas vantagens de um verdadeiro DBMS relacional e elas não virão. As 12 regras de Codd juntamente com as 9 estruturais, as 18 manipulativas e as 3 características de integridade do modelo relacional determinam, com detalhes, a extensão da veracidade da declaração de um vendedor ao estar vendendo um DBMS totalmente relacional.

**Regra 1. Regra de Informação**

*Toda informação em uma base de dados relacional é representada explicitamente em nível lógico e exatamente de uma única maneira, pelos valores nas tabelas.*

Até mesmo nomes de tabelas, nomes de colunas e nomes de domínios são representados como cadeias de caracteres em algumas tabelas. As tabelas que contêm esses nomes normalmente fazem parte do catálogo interno do sistema. O catálogo é, portanto, uma base de dados relacional dinâmica e ativa que representa os metadados (*i.e.*, dados que descrevem o restante dos dados no sistema). A Regra 1 é mandatória não apenas para a promover a produtividade do usuário, mas também para facilitar o trabalho de fornecedores de software quando da especificação de pacotes adicionais de software (*e.g.*, desenvolvimento de aplicativos auxiliares, sistemas especialistas, etc.) que têm interface com DBMS relacionais e que, por concepção, estão bem integrados ao DBMS relacional. Os softwares disponibilizados nos pacotes recuperam informações já existentes no catálogo e, dependendo da situação, inserem novas informações no catálogo por meio da própria utilização do DBMS relacional. O uso da Regra 1 também torna mais simples e mais eficaz a tarefa realizada pelo administrador da DB relacional, que é a de manter a DB relacional em estado de integridade geral.

**Regra 2. Regra do Acesso Garantido**

*Garante que todo e qualquer dado (valor atômico) em uma base de dados relacional seja logicamente acessível por meio de uma combinação de nome de tabela, valor de chave primária e nome de coluna.*

Cada dado em um DB relacional pode ser acessado por meio de um vasto número de diferentes expressões lógicas. É importante, entretanto, que pelo menos uma das expressões seja prevalente, independentemente da DB relacional sendo considerada, ressaltando que muitos conceitos relacionados a computador e à sua estrutura (tal como verificação de endereços sucessivos) foram deliberadamente omitidos do modelo relacional. A Regra 1 expressa um esquema de endereçamento associativo que é específico do modelo relacional e que independe do endereçamento usual orientado a computador. No entanto, o conceito de chave primária, detalhado na Capítulo 02 é uma parte essencial do endereçamento.

**Regra 3. Regra do Tratamento Sistemático de Valores Nulos**

*Valores nulos (distintos de uma cadeia de caracteres vazia ou de uma cadeia de caracteres em branco e distintos de zero ou de qualquer outro número) são completamente mente aceitos para representar informações ausentes e informações que não são aplicáveis de forma sistemática em DBMS totalmente relacional, independentemente do tipo de dados.*

Para garantir a integridade da DB relacional deve ser possível especificar “nulos não permitidos” para cada coluna de chave primária e para quaisquer outras colunas que o administrador da DB considere uma restrição de integridade apropriada. É importante lembrar que técnicas anteriores envolviam a definição de um valor especial (peculiar a cada coluna ou campo) para representar a informação faltante. Essa prática, entretanto, resultaria em um procedimento assistemático em um DB relacional uma vez que usuários teriam que empregar técnicas diferentes para cada coluna ou domínio, o que traria dificuldades devido ao alto nível da linguagem em uso e, também, contribuiria para diminuir a produtividade do usuário.

**Regra 4. Catálogo dinâmico *on-line* baseado no modelo relacional**

*A descrição da base de dados é representada em nível lógico da mesma forma que dados comuns, de modo que os usuários autorizados possam usar em suas interrogações a mesma linguagem relacional que eles usam em dados regulares.*

O uso da Regra 4 tem duas consequências:

(1) cada usuário (seja um programador de aplicação ou um usuário final) precisa aprender apenas um modelo de dados, o que pode ser considerado uma vantagem que sistemas não relacionais geralmente não oferecem (o IMS da IBM, juntamente com seu dicionário, exige que o usuário aprenda dois modelos distintos);

(2) usuários autorizados podem facilmente estender o catálogo para torná-lo um dicionário de dados relacional ativo e completo sempre que o fornecedor não o fizer.

**Regra 5. Regra da sublinguagem de dados compreensível**

*Um sistema relacional pode disponibilizar diversas linguagens bem como várias maneiras de uso de terminal (por exemplo, por meio do preenchimento de lacunas). No entanto, deve haver pelo menos uma linguagem que permite que instruções possam ser expressas de acordo com uma sintaxe bem definida, tal como cadeias de caracteres, e que essa linguagem seja abrangente o suficiente para que possa ser usada em vários processos:*

*• Definição de dados.*

*• Definição de views.*

*• Manipulação de dados (interativa e via programa).*

*• Restrições de integridade.*

*• Autorização.*

*• Limites de transação (início, confirmação e reversão).*

A abordagem relacional é intencionalmente bem dinâmica. Raramente será necessário interromper a atividade do DB (em contraste com o SGBD não relacional). Portanto, não faz sentido separar os processos listados, usando linguagens distintas.

**Regra 6. Atualização de *views***

*Todas as views que teoricamente são atualizáveis são também passíveis de serem atualizadas pelo sistema.*

É importante lembrar que teoricamente uma *view* é atualizável se existir um algoritmo independente de tempo que possa determinar, inequivocamente, uma única série de mudanças nas relações da base, que terão como efeito, precisamente, as mudanças solicitadas na *view*. No contexto em questão o termo “atualização” reflete operações de inserção e exclusão, bem como de modificação.

**Regra 7. Operações de *insert*, *update* e *delete* em alto nível**

*A capacidade de tratar uma relação base ou uma relação derivada como sendo um único operando, se aplica não apenas à recuperação de dados, mas também às operações de inserção, atualização e exclusão de dados.*

O uso da Regra 7 dá ao sistema maior abrangência na otimização da eficiência de suas ações em tempo de execução. A regra estabelece que o sistema possa determinar quais caminhos de acesso devem ser explorados para a obtenção de código mais eficiente. Também, o estabelecido pela regra pode ser de grande relevância na obtenção de um tratamento eficiente de transações em bancos de dados distribuídos. Em um contexto distribuído os usuários prefeririam que custos de comunicação fossem poupados, evitando assim a necessidade de transmitir um pedido separado para cada registo obtido de locais remotos.

**Regra 8. Independência de dados físicos**

*Programas aplicativos e atividades de terminal permanecem logicamente inalterados sempre que quaisquer alterações são feitas ou nas representações de armazenamento ou nos métodos de acesso.*

Para lidar com programas aplicativos e atividades de terminal o SGBD deve dispor de uma fronteira clara e nítida entre (a) aspectos lógicos e semânticos, por um lado, e (b) aspectos físicos e de desempenho das tabelas da BD, por outro. Programas aplicativos devem lidar apenas com os aspectos lógicos. SGBDs não relacionais raramente fornecem suporte completo para esta regra.

**Regra 9. Independência de dados lógicos**

*Programas aplicativos e atividades de terminal permanecem logicamente inalterados quando alterações de preservação de informações de qualquer tipo que teoricamente permitem a integridade são feitas nas tabelas base.*

Considere as duas situações: (1) dividir uma tabela em duas tabelas, seja por linhas usando conteúdo de linha ou por colunas usando nomes de colunas, se as chaves primárias forem preservadas em cada resultado; (2) combinar duas tabelas em uma por meio de uma junção sem perdas (os autores da Universidade de Stanford e do MIT chamam essas junções de “sem perdas”). Para fornecer este serviço sempre que possível, o SGBD deve ser capaz de lidar com inserções, atualizações e exclusões em todas as visualizações que são teoricamente atualizáveis. Esta regra permite que o *design* do banco de dados lógico seja alterado dinamicamente se, por exemplo, tal alteração melhorar o desempenho.

As regras de independência de dados físicos e lógicos permitem que projetistas de DBs para DBMS relacionais cometam erros em seus projetos sem as pesadas penalidades impostas pelos DBMS não relacionais. Isso, por sua vez, significa que é muito mais fácil começar com um DBMS relacional porque não é necessário tanto planejamento orientado ao desempenho antes da “decolagem”.

**Regra 10. Independência de integridade**

*Restrições de integridade específicas a uma determinada DB relacional devem ser passíveis de serem definidas na sub-linguagem de dados relacionais e, também, passíveis de serem armazenadas no catálogo e não nos programas aplicativos.*

Além das duas regras de integridade (integridade de entidade e integridade referencial) que se aplicam a todos os DBs relacionais, fica claro que há ainda a necessidade de poder especificar restrições de integridade adicionais que reflitam políticas de negócios ou regulamentações governamentais.

Suponha que o modelo relacional tenha sido fielmente seguido e, portanto, as restrições de integridade adicionais são definidas em termos da sub-linguagem de dados de alto nível e das definições armazenadas no catálogo, e não nos programas aplicativos. Informações sobre objetos identificados inadequadamente nunca são registradas em um BD relacional. Para promover especificidade, as duas regras de integridade a seguir se aplicam a todos os DBs relacionais:

* Integridade da entidade. Nenhum componente de uma chave primária pode ter um valor nulo.
* Integridade referencial. Para cada valor de chave estrangeira não nulo distinto em um BD relacional, deve existir um valor de chave primária correspondente do mesmo domínio.

Como acontece esporadicamente de políticas empresariais ou regulamentações governamentais mudarem, provavelmente será necessário alterar as restrições de integridade. Normalmente, isso pode ser feito em um DBMS totalmente relacional, alterando uma ou mais instruções de integridade armazenadas no catálogo. Em muitos casos, nem os programas aplicativos nem as atividades do terminal são prejudicados logicamente. Raramente DBMSs não relacionais suportam essa regra como parte do mecanismo do DBMS, ao qual a regra pertence. Em vez disso, eles dependem de um pacote de dicionário, que pode ou não estar presente e que pode ser facilmente ignorado.

**Regra 11. Um DBMS relacional tem independência de distribuição**

*Por independência de distribuição nesse contexto deve ser entendido que o DBMS possui uma sub-linguagem de dados que permite que programas aplicativos e atividades de terminal permaneçam logicamente intactos quando:*

*• a distribuição de dados é implementada pela primeira vez (se o DBMS originalmente instalado gerencia apenas dados não distribuídos);*

*• os dados são redistribuídos (se o DBMS gerencia dados distribuídos).*

Observe que a definição é cuidadosamente redigida para que tanto o DBMS distribuído quanto o não distribuído possam garantir totalmente a Regra 11. É importante distinguir entre processamento distribuído e dados distribuídos. Em processamento distribuído o trabalho (por exemplo, programas) é transmitido aos dados. Já em dados distribuídos os dados são transmitidos para a realização de uma atividade. É fato que muitos DBMS não relacionais suportam processamento distribuído, mas não dados distribuídos. Os únicos sistemas que suportam o conceito de fazer com que todos os dados distribuídos pareçam locais são SGBDs relacionais.

No caso de um SGBD relacional distribuído, uma única transação pode abranger vários *sites* remotos. Esse ‘espalhamento’ é gerenciado inteiramente nos bastidores – o sistema pode ter que executar a recuperação em vários locais. Cada programa ou atividade de terminal trata a totalidade dos dados como se fossem todos locais do site onde o programa aplicativo ou atividade de terminal está sendo executado.

Um SGBD totalmente relacional que não suporta bancos de dados distribuídos tem a capacidade de ser estendido para fornecer esse suporte, deixando os programas aplicativos e as atividades de terminal logicamente intactas, tanto no momento da distribuição inicial quanto sempre que a redistribuição posterior for feita. Existem quatro razões importantes pelas quais um DBMS relacional desfruta dessa vantagem:

• Flexibilidade de decomposição na decisão de como implantar os dados.

• Poder de recomposição dos operadores relacionais ao combinar os resultados de subtransações executadas em diferentes *sites*.

• Economia de transmissão devido ao fato de não ser necessário enviar uma mensagem de solicitação para cada registro a ser recuperado de qualquer site remoto.

• Capacidade de análise da intenção (devido ao alto nível das linguagens relacionais) para uma otimização de execução amplamente melhorada.

**Regra 12. Regra da nãosubversão**

*Se um sistema relacional tiver uma linguagem de baixo nível (registro único por vez), esse baixo nível não pode ser usado para subverter ou contornar as regras e restrições de integridade expressas na linguagem relacional de nível superior (múltiplos registros por vez).*

Na abordagem relacional a preservação da integridade é independente da estrutura lógica de dados para alcançar a independência da integridade. Esta regra é extremamente difícil para um sistema “nascido de novo” obedecer porque tal sistema já suporta uma interface abaixo da interface de restrição relacional. Os fornecedores de sistemas “nascidos de novo” não parecem ter dado a devida atenção a este problema.