INTERFACE TECNOLÓGICA

# LIMITAÇÕES DE UM SISTEMA DE GERENCIAMENTO DE BANCO DE DADOS EM MEIO A UM MODELO TRANSACIONAL

## *LIMITATIONS OF A DATABASE MANAGEMENT SYSTEM IN A TRANSACTIONAL MODEL*

Maria Cristina Diniz Ribeiro – mcristinaderibeiro@gmail.com
Fernando Tiosso – fernando.tiosso@fatectq.edu.br
Erick Eduardo Petrucelli – erick.petrucelli@fatectq.edu.br
Faculdade de Tecnologia de Taquaritinga (FATEC) – SP – Brasil

## RESUMO

O objetivo deste artigo é proporcionar o entendimento do sistema de banco de dados utilizando um Sistema de Gerenciamento de Banco de Dados (SGBD) relacional em meio a um modelo transacional e evidenciar possíveis limitações, bem como algumas soluções para as tarefas periódicas de emissão de relatórios. Nesse ambiente, dependendo do nível de isolamento configurado no SGBD, a tarefa de emissão de relatórios pode gerar uma concorrência desnecessária pelos dados, sobrecarregando os processos executados pelo SGBD e prejudicando o desempenho do sistema. Algumas soluções propõem alterações de níveis de isolamento do SGBD que aumentam o desempenho do sistema, mas geram relatórios com dados não confiáveis. No entanto, após o entendimento do sistema de banco de dados e dos níveis de isolamento existentes em um SGBD, é possível configurar um ambiente seguro, com informações fidedignas, e com alto desempenho para a tarefa de emissão de relatórios.

**Palavras-chave:** Banco de dados. SGBD. Controle de concorrência. Modelo transacional.

## ABSTRACT

The purpose of this article is to provide an understanding of the database system by using a relational Database Management System (DBMS) in a transactional model and highlight possible limitations as well as some solutions to the periodic reporting tasks. In this environment, depending on the level of isolation configured in the DBMS, the reporting task can generate unnecessary competition through the data, overloading the processes executed by the DBMS and damaging the performance of the system. Some solutions propose changes in DBMS isolation levels that increase system performance but generate reports with unreliable data. However, after understanding the database system and isolation levels in a DBMS, one can configure a secure, reliable, and high-performing environment for the reporting task.

**Keywords:** Database. DBMS. Concurrency control. Transactional model.

## 1 INTRODUÇÃO

Na sociedade moderna, os sistemas de bancos de bancos estão intrinsicamente ligados a vida das pessoas. Existem vários tipos de banco de dados, desde mais simples até outros muito complexos. Um banco de dados simples, comumente encontrado no cotidiano das pessoas é a lista telefônica, responsável por armazenar nomes e telefones dos indivíduos e permitir que esses dados sejam consultados de forma organizada.

Segundo Date (2003) em tempos mais antigos, os bancos de dados eram utilizados apenas para armazenar dados numéricos e/ou textuais, mas com os avanços tecnológicos e a crescente utilização de sistemas computacionais, surgiram novas necessidades de armazenamento, como, por exemplo, a possibilidade de armazenar conteúdos multimídias, tais como: imagens, clipes de áudios, *streams* de vídeo e mapas.

Um bom exemplo para demonstrar as novas necessidades de armazenamento dos bancos de dados são as redes sociais, como, por exemplo, o Facebook. Nela, o usuário tem uma conta e pode publicar fotos, vídeos, áudios que serão armazenados pelos bancos de dados e, mesmo com o passar do tempo, talvez anos, os dados armazenados devem estar disponíveis para a utilização do usuário em tempo real. Aliado a isso, deve-se ressaltar que essa necessidade não faz referência a um único usuário, e sim a milhões de usuários que podem realizar seus acessos simultaneamente.

Neste cenário, destacam-se duas características importantes de um sistema de banco de dados: o armazenamento e o controle das operações de inserção, alteração, exclusão e pesquisa dos dados realizadas concomitantemente. No entanto, a maior responsabilidade do banco de dados é o armazenamento dos dados de forma organizada. No exemplo do Facebook, os dados publicados pelo usuário em sua *timeline* foram armazenados de forma organizada, ou seja, a informação total a ser armazenada foi subdivida em dados, que por sua vez foram novamente subdivididos até que o processo de subdivisão gerasse somente dados atômicos (indivisíveis). Assim, não havendo mais possibilidades de subdivisão, os dados foram devidamente organizados em tabelas para que pudessem ser prontamente recuperados (SILBERSCHATZ; KORTH; SUDARSHAN, 2012).

Já para o cenário do controle das operações, destaca-se a importância de outro componente do sistema de banco de dados, o Sistema de Gerenciamento de Banco de Dados (SGBD). Segundo Elmasri e Navante (2011), as quatro características principais de um SGBD são: permitir a construção, definição, manutenção e o compartilhamento dos dados

armazenados no banco de dados entre usuários e aplicações. Assim, ele é o responsável por receber uma solicitação e encaminhá-la ao banco de dados, reunindo ou subdividindo os dados. No exemplo da *timeline* do Facebook, o SGBD foi o responsável por incluir os dados no banco de dados e posteriormente reuni-los a fim de gerar as informações apresentadas para o usuário final. Além de suas quatro principais características, um SGBD também atua no controle de redundância, mantém a consistência, controla o acesso e a concorrência dos usuários aos dados.

A junção do banco de dados, SGBD e as aplicações formam um sistema de banco de dados, que tem a função de garantir uma visão totalmente abstrata do banco de dados para o usuário final, ou seja, o usuário da aplicação não tem a necessidade de saber qual unidade de armazenamento está sendo usada para cuidar dos seus dados, contanto que os mesmos estejam devidamente disponíveis no momento necessário(ELMASRI; NAVANTHE, 2011).

No entanto, dependendo da configuração de determinadas características do SGBD, algumas tarefas, como, por exemplo, a tarefa de emissão de relatórios, podem gerar uma concorrência desnecessária pelos dados do banco de dados, sobrecarregando os sistemas de banco de dados e prejudicando seu desempenho.

Assim, com o objetivo de explorar essa conjuntura, o presente trabalho foi desenvolvido em 5 (cinco) sessões, sendo estas: a sessão 2 (dois) apresenta a metodologia de pesquisa a qual foi utilizada na construção da pesquisa, a sessão 3 (três) apresenta a fundamentação teórica onde está descrito todo funcionamento de um sistema de banco de dados, a sessão 4 (quatro) apresenta uma proposta de caso de uso para futuros trabalhos, e, por fim, a sessão 5 (cinco) apresenta as considerações finais.

## 2 METODOLOGIA DE PESQUISA

Segundo Gil (2002), a revisão bibliográfica é a base que sustenta qualquer pesquisa, visto que ela compreende o procedimento racional e sistemático que tem como objetivo proporcionar respostas aos problemas que são propostos. Com esse entendimento, é possível confirmar que ao se propor fazer uma pesquisa, um pesquisador está disposto a buscar algo que vai além do que já foi apresentado, buscar um maior conhecimento sobre algo que o mesmo deseja se aprofundar e o resultado de um problema pré-definido.

Desta forma, neste artigo, a revisão bibliográfica tornou-se indispensável para a delimitação do problema, evidenciando uma lacuna que originou a investigação para o desenvolvimento de um novo conhecimento (LAKATOS; MARCONI, 2010).

## 3 FUNCIONAMENTO DE UM SISTEMA DE BANCO DE DADOS

Um sistema de banco de dados é composto pelo banco de dados, SGBD, aplicações e usuários. Para compreender seu funcionamento, faz-se necessário entender algumas das partes que o compõe, bem como as fases de seu desenvolvimento, conforme descrevem os tópicos subsequentes.

### 3.1 Iniciando a criação de um banco de dados com um SGBD relacional

Na fase inicial da construção de um banco de dados deve ser elaborado um projeto, que tem como objetivo a organização dos dados que deverão gerar as informações e a utilização de técnicas para que o futuro sistema obtenha o máximo desempenho e facilite as manutenções. Essa construção ocorre em etapas, sendo iniciada com o modelo conceitual, em seguida o modelo logico e, por fim, a implementação do banco de dados com o modelo físico.

O modelo conceitual consiste na elaboração do banco de dados de maneira independente do SGBD, definindo como os dados serão organizados no banco de dados sem priorizar sua implementação. Nesta etapa, os conceitos do Modelo Entidade Relacionamento (MER) são aplicados nos requisitos funcionais do sistema e utilizados como referência para compor a estrutura do banco de dados que suportará as operações. Para ser melhor interpretado e facilmente visualizado pelos projetistas de bancos de dados, o MER é representado por meio do Diagrama Entidade Relacionamento (DER) (SILBERSCHATZ; KORTH; SUDARSHAN, 2012).

Após à definição do modelo conceitual, inicia-se a elaboração do modelo lógico que define como a estrutura do banco de dados a ser utilizada pelo SGBD será implementada: hierárquica, em rede, relacional, orientada a objetos, orientada a documentos, entre outras. Neste momento, também são definidas as restrições de integridade, caso se apliquem ao tipo de SGBD (ELMASRI; NAVANTHE, 2011).

Finalmente, é executada a etapa do modelo físico, onde ocorre a definição do SGBD e a implementação do banco de dados. Uma análise das características e recursos necessários para armazenamento e manipulação das estruturas de dados é realizada e uma sequência de comandos e instruções são executadas para criar as estruturas físicas do banco de dados no SGBD definido (ELMASRI; NAVANTHE, 2011).

Neste projeto, foi adotado o tipo de SGBD relacional que permite organizar e normalizar os dados através das relações entre tabelas e a utilização de restrições de integridade, principalmente as de exclusividade e referencial, que juntamente com os processos transacionais garantem a consistência das operações e a integridade dos dados.

Segundo Silberschatz e Korth, (2012), o conceito de integridade referencial que trata das relações entre as tabelas, é possível devido a existência das *foreign key*s (FK) ou chaves estrangeiras. O dado que é uma chave estrangeira em uma tabela destino, deve ser *primary key* (PK) ou chave primária na tabela origem. Uma PK caracteriza-se pelo menor conjunto possível de um ou mais dados de uma tabela cujo valor nunca será repetido, garantindo sua integridade de exclusividade. Assim, quando uma chave primária ou estrangeira é violada, o SGBD gera um evento de violação de integridade, impossibilitando o armazenamento dos dados e garantindo sua integridade em meio às transações.

Com o objetivo de melhor entender o ambiente transacional e seus principais conceitos, o modelo transacional será explicado a seguir.

## 3.2 Modelo transacional

O modelo transacional é aplicado quando o banco de dados recebe várias transações, sendo muito útil em SGBDs que suportam aplicações de grande porte com centenas, milhares ou até mesmo milhões operações executadas simultaneamente por usuários.

Segundo Elmasri e Navante (2011), o conceito de transação baseia-se na execução de um programa ou processo com um ou mais acessos ao banco de dados de forma completa e sem interferências externas, ou seja, uma transação deve ser tratada como uma sequência de operações indivisíveis, sendo executada em sua totalidade. Caso uma operação falhe, a transação deve ser desfeita em sua totalidade.

### 3.2.1 Conceitos ACID

Para garantir o conceito transacional dos SGBDs, o conceito ACID foi estabelecido, garantindo a Atomicidade, Consistência, Isolamento e Durabilidade das operações. A atomicidade trata uma transação no banco de dados como uma sequência de operações única e indivisível, popularmente conhecida como operação atômica. O isolamento garante que as

ações executadas por uma transação não sofram interferências de ações simultâneas externas à transação, isolando cada transação executada pelo sistema. A durabilidade estabelece que os dados validados pelo sistema estarão disponíveis mesmo após falhas e/ou reinícios por ele apresentados. A consistência garante que a execução de uma transação deve levar o banco de dados de um estado consistente para um outro estado consistente, ou seja, uma transação modifica o estado dos dados para um novo estado válido ou, em caso de falhas, retorna todos os dados ao estado anterior (ELMASRI; NAVANTHE, 2011).

Além destas propriedades, o SGBD precisa garantir que seu módulo de escalonamento de transações não cause perda de informações no momento em que se tenta aplicar um conjunto de transações que podem estar desordenados. Para isso, é necessário que o escalonador garanta que uma transação confirmada não possa ler dados de uma transação abortada. Em casos de transações abortadas, o escalonador deve garantir que todas as outras transações que leram dados modificados pela transação abortada também sejam abortadas (SILBERSCHATZ; KORTH; SUDARSHAN, 2012).

Como o modelo transacional permite a execução de várias transações entre muitos usuários simultaneamente, alguns dados podem sofrer alterações simultâneas, gerando concorrências. O conceito de concorrência em banco de dados trata dos métodos necessários para que mais de um usuário ou aplicação possa acessar os dados de forma compartilhada (FINGER; FERREIRA, 2000).

No entanto, quando ocorre o acesso simultâneo de dados provenientes de uma alteração, é preciso estabelecer um controle com o objetivo de prevenir a perda de informações durante a concorrência. Esse controle é proporcionado pelas técnicas de controle de concorrência, e, de uma forma geral, as mais utilizadas são os bloqueios (*locks*) e os níveis de isolamento, explicados a seguir.

## 3.3 Tipos de bloqueios

Os bloqueios têm o objetivo de prevenir que os dados sejam corrompidos ou invalidados quando são envolvidos em operações simultâneas. Eles podem ser categorizados em bloqueios do tipo pessimista e otimista.

### 3.3.1 Bloqueio otimista e pessimista

No bloqueio otimista, nenhuma interrupção é realizada enquanto os dados sofrem operações de leitura, ou seja, as interrupções são realizadas somente nas operações de atualização dos dados, mais precisamente, no momento exato em que elas ocorrem por questões de milissegundos, pois nesse tipo de bloqueio as atualizações são aplicadas aos dados somente quando a transação alcançar seu estágio final. Durante a execução da transação as atualizações são aplicadas aos dados em cópias locais e, no final da fase de validação sem a ocorrência de violações, os dados são atualizados de acordo com as cópias locais, caso contrário, a transação é abortada. De acordo com Elmasri e Navanthe (2011), uma transação que sofreu um bloqueio do tipo otimista, passou por três fases do controle de concorrência: leitura, validação e gravação.

Assim, esse tipo de bloqueio pode ser chamado de otimista porque considera um cenário ideal para a realização das transações, havendo pouca ou nenhuma interferência de outras transações (ELMASRI; NAVANTHE, 2011).

Porém, nem toda transação ocorre nesse cenário ideal e, a partir disso, sugiram as técnicas de bloqueios pessimistas, que excedem o ambiente do SGBD e entram no nível da aplicação, pois designers de aplicativos, desenvolvedores e administradores de bancos de dados podem escolher níveis de isolamento apropriados para diferentes transações, dependendo do aplicativo e da sua carga de trabalho. Basicamente, o bloqueio pessimista bloqueia todas as tuplas que estão sendo modificadas pelo usuário/aplicação e a partir disto não será realizada nenhuma alteração até que este bloqueio (*lock*) seja liberado.

Este bloqueio é útil para situações onde a modificação do dado durante uma transação pode ocasionar erros ou tomada incorreta de decisões para outras transações. No entanto, com esta característica, dados podem ser bloqueados por longos períodos e, desta forma, surge a necessidade de se criar níveis para esses bloqueios, mais conhecidos como níveis de isolamento que serão apresentados a seguir.

## 3.4 Níveis de Isolamento

Os níveis de isolamento especificam como as transações que modificam o banco de dados serão manipuladas, como uma propriedade que define como e/ou quando as alterações feitas por uma transação irão se tornar visível para outros usuários e/ou aplicações. Em geral, os níveis de isolamento são definidos pela presença ou ausência de:

- Dados sujos (*Dirty Read*): ao realizar uma transação no banco de dados, uma aplicação(A) acessa um dado para alteração e, nesse mesmo momento, outra aplicação(B) acessa o mesmo dado para leitura. No entanto, caso a aplicação A reverta a atualização, a aplicação B continuará fazendo uso de dados desatualizados, originando o termo dado sujo (ORACLE COPORATION, 2003).
- Leitura não repetíveis (*Non-repeatable Read*): ocorre quando em uma única transação mesmas informações são lidas mais de uma vez, fazendo com que a primeira leitura possa utilizar informações diferentes das próximas leituras, inviabilizando a garantia da consistência da informação dentro da mesma transação (ORACLE COPORATION, 2003).
- Fantasmas (*Phanton Read*): um fantasma consiste em uma tupla que corresponde aos critérios de uma pesquisa, mas não foi obtida na pesquisa inicial de uma transação que visa a atualização de dados. Assim, pode-se ocorrer atualizações de tuplas que não foram identificadas em uma pesquisa inicial, mas que atendam os critérios estipulados pela clausula *where* de uma instrução, alterando um número de tuplas diferente do esperado, ocasionando as *phanton reads* (ORACLE COPORATION, 2003).

Um nível de isolamento baixo aumenta a capacidade de acessos simultâneos aos dados e diminui o tempo de obtenção dos mesmos, mas eleva o número de efeitos colaterais provocados pela simultaneidade, caracterizados pela leitura de dados sujos, perdas e/ou excessos de atualizações. Em contrapartida, um nível de isolamento alto reduz os problemas ocasionados pela concorrência dos dados, porém consome mais recursos do sistema e aumenta as chances de uma transação bloquear outra, elevando consideravelmente o tempo de obtenção dos dados, podendo gerar *timeouts* por *lock* de transações (IBM, 2014).

Assim, a definição do nível de isolamento apropriado depende de equilibrar os requisitos de integridade dos dados da aplicação em relação à sobrecarga de cada nível de isolamento (IBM, 2014). Com o objetivo de proporcionar um melhor entendimento dessa relação, os níveis de isolamento são explicados a seguir:

*Serializable*: esse é o nível mais alto de isolamento, garantindo que uma transação recuperará exatamente os mesmos dados toda vez que repetir uma operação de leitura. A transação aguarda até que o bloqueio de gravação por outras transações seja desbloqueado,

impedindo a leitura dos dados "sujos", mas impactando o acesso dos dados por outros usuários em sistemas multiusuários (ORACLE CORPORATION, 2005).

*Read uncommitted*: esse nível de isolamento é o mais baixo, podendo recuperar dados que foram modificados e não confirmados por outras transações. Todos os efeitos colaterais da simultaneidade podem ocorrer neste tipo de isolamento, mas nenhum bloqueio ou controle de versão será realizado (ROTH, 2018).

*Read committed*: nesse nível de isolamento a transação que deseja obter os dados deverá aguardar até o fim do bloqueio executado por outras transações, impedindo a leitura de dados sujos (ORACLE CORPORATION, 2005).

*Read repeat*: esse nível de isolamento mantém os bloqueios de leitura e gravação até o final da transação, no entanto, os bloqueios de intervalo não são gerenciados, portanto, podem ocorrer leituras fantasmas (RABELER, 2018).

Após o entendimento do funcionamento de um sistema de banco de dados desde o alto nível, nível do usuário, até o baixo nível, nível de bloqueios e isolamentos existentes para atuar na concorrência dos dados, analises de cenários podem ser realizadas e soluções podem ser propostas visando obter o melhor desempenho do SGBD em relação aos programas por ele suportados. Assim, no item subsequente, apresenta-se uma proposta para obtenção de dados em um ambiente seguro e com alto desempenho para uma tarefa comumente executada por diversos tipos de sistemas: a emissão de relatórios.

## 4 PROPOSTA

Um cenário comumente utilizado em diversos sistemas multiusuários é a concorrência dos dados pelas ações de atualização e leitura. Como exemplo, cita-se um cenário em que uma empresa necessita de relatórios periódicos para uma visão ampla do seu negócio, por meio da análise de suas vendas realizadas em períodos mensais, semestrais e anuais.

Neste caso, supõe-se que o SGBD esteja configurado com um nível de isolamento que garante a ausência de dados sujos (*Dirty Read*) e fantasmas (*Phanton Read*), como, por exemplo, os níveis de isolamento *Serializable* e *Read* committed. Uma aplicação que irá buscar os dados, agrupá-los e emitir diversos relatórios, certamente terá dificuldades de passar por todo processo de controle de concorrência afim de obter os dados de forma segura para o usuário final. O tempo de exibição dos relatórios será demasiadamente alto e *locks* poderão ocorrer em

diversos pontos do SGBD, podendo até originar timeouts em algumas operações, prejudicando o desempenho do sistema em geral.

Para estes casos, a fim de diminuir bruscamente os problemas de *locks* e timeouts, geralmente o nível de isolamento do SGBD é alterado para *Read Uncommited*, pois, como explicado anteriormente, neste nível de isolamento nenhum bloqueio ou controle de versão será realizado nos dados e o desempenho do sistema será afetado de forma positiva. No entanto, efeitos colaterais da simultaneidade podem ocorrer neste tipo de isolamento, como a leitura de dados sujos e esse fato não será percebido pelos usuários finais, que poderão estar analisando informações equivocadas para tomada de decisão sobre a empresa.

No entanto, existe uma solução que mantém o nível de isolamento do SGBD para não permitir a leitura de dados sujos e/ou dados fantasmas e permite a emissão de relatórios íntegros e com performance aceitável pelo usuário final. É importante ressaltar que a solução foi sugerida, analisada e desenvolvida utilizando o SGBD Oracle com nível de isolamento *Read committed* para um banco de dados relacional, aplicando-se o modelo transacional. Provavelmente, essa solução poderá atender a necessidade de outras limitações que possam surgir em outros modelos de banco de dados, porém, neste trabalho, será apresentado como uma solução viável apenas para o cenário proposto.

Ao invés das aplicações exibirem seus relatórios por meio dos dados obtidos nas tabelas que sofrem alterações a todo momento por diversas partes do sistema, podem ser criadas tabelas para atenderem necessidades especificas de um ou mais relatórios, com estruturas que permitem dados redundantes, sumarizados, mas que não sofrerão atualizações contínuas.

Para exemplificar este cenário, pode-se imaginar um relatório que exibe as vendas realizadas em um determinado período. Para atender seu requisito, uma tabela será criada no banco de dados com uma estrutura viável para armazenar os dados utilizados por este relatório, tais como: código e nome do cliente, data venda, valor total da venda. Pode-se notar que, em uma mesma tabela, existem dados de clientes e de vendas, contrariando o processo de normalização dos dados, mas tornando eficaz o processo de emissão do relatório. No momento em que a exibição do relatório for solicitada pelo usuário, o sistema não precisará buscar os dados em mais de uma tabela e realizar processos de junções e/ou sumarizações, pelo contrário, os dados serão selecionados em uma única tabela, com instruções simples, pois já estão no formato esperado atendendo perfeitamente a necessidade do relatório.

No entanto, algum processo precisa ser executado para inserir os dados nessa nova tabela e este processo é caracterizado por um “processo em lote” (*batch*). O processo *batch* tem

a característica de não exigir a interação do usuário com a aplicação, podendo ser executado de forma automática e seguindo uma agenda previamente definida. Além disso, o horário da sua execução deve ser bem definido, devendo coincidir preferencialmente com períodos em que o SGBD atende poucas transações, evitando concorrências desnecessárias e possibilitando que junções e sumarizações de dados sejam realizadas mais rapidamente.

A partir da funcionalidade correta deste processo, a aplicação não terá mais que concorrer com outras aplicações para obter os dados do relatório em momentos de altas taxas transacionais no sistema, pois os mesmos já estarão prontamente disponibilizados em uma tabela, garantindo um relatório confiável, com baixa concorrência, aumentando a disponibilidade dos dados armazenados no banco de dados.

## 5 CONSIDERAÇÕES FINAIS

Durante o desenvolvimento deste trabalho, foi possível pesquisar e entender o funcionamento de um sistema de banco de dados, desde a construção do banco de dados por meio do modelo conceitual e lógico, até a sua implementação no modelo físico, destacando a concorrência pelos dados, os tipos de bloqueios e níveis de isolamento para o entendimento do funcionamento do SGBD junto com aplicação em um sistema multiusuário, evidenciando possíveis limitações em alguns cenários em meio a um modelo transacional, mais especificamente em atividades que visavam atender a necessidade de sistemas multiusuários para geração de relatórios periódicos.

Neste contexto, verificou-se que a solução mais utilizada é a alteração do nível de isolamento do SGBD para *Read Uncommited*, a fim de diminuir consideravelmente os problemas de *locks* e *timeouts*, mas, por outro lado, eximindo-se da responsabilidade de não utilizar dados sujos e desatualizados para impedir a geração de um relatório com informações duvidosas.

Assim, após analisar este cenário típico e comumente utilizado por diversos sistemas e suas limitações em meio ao modelo transacional, foi possível propor uma solução viável através da configuração de um ambiente seguro, com informações fidedignas, e com alto desempenho para a tarefa de emissão de relatórios e,, em trabalhos futuros a solução proposta neste artigo poderá ser aplicada na prática em um estudo de caso que poderá ser descrito em um novo artigo contendo a análise de todo o experimento realizado.

## REFERÊNCIAS

DATE, C. J. **Introdução a Sistemas de Bancos de Dados**. 8. ed. Rio de Janeiro: Campus, 2003.

ELMASRI, R; NAVANTHE, S. B. **Sistemas de Banco de Dados**. 6. ed. São Paulo: Imprensa Oficial, 2011.

FINGER, M; FERREIRA, J. E. **Controle de Concorrência e Distribuição de Dados**: Teoria Clássica, suas Limitações e Extensões Modernas. Livro preparado para a Escola de Computação, Instituto de Matemática e Estatística, USP, 2000. Disponível em: <ime.usp.br/~jef/ec2000.ps >. Acesso em: 16 de fevereiro de 2019.

GIL, A. C. **Métodos e Técnicas de Pesquisa Social**. 5. ed. São Paulo: Atlas, 1999.

IBM. **Níveis de Isolamento**. IBM Knowledge Center, 2014. Disponível em: <ibm.com/support/knowledgecenter/pt-br/SSEP7J_11.0.0/com.ibm.swg.ba.cognos.ug_cra.doc/c_isolationlevels.html>. Acesso em: 23 de março de 2019.

LAKATOS, E. M; MARCONI, M. A. **Fundamentos de Metodologia Científica**. 5. ed. São Paulo: Atlas, 2003.

ORACLE COPORATION. **Data Concurrency and Consistency**. Oracle Database Concepts, 2003. Disponível em: <docs.oracle.com/cd/B13789_01/server.101/b10743/consist.htm>. Acesso em: 23 de março de 2019.

ORACLE CORPORATION. **SET TRANSACTION**. Database SQL Reference, Oracle Help Center, 2005. Disponível em: <docs.oracle.com/cd/B19306_01/server.102/b14200/statements_10005.htm#i2067247>. Acesso em: 23 de março de 2019.

RABELER, C. **Transaction Isolation Level**. Documentação do SQL Server 2017, Microsoft, 21 de outubro de 2018. Disponível em: <docs.microsoft.com/pt-br/sql/t-sql/statements/set-transaction-isolation-level-transact-sql?view=sql-server-2017>. Acesso em: 22 de março de 2019.

ROTH, J. **Guia de Controle de Versão de Linha e Bloqueio de Transações**. Documentação do SQL Server 2017, Microsoft, 16 de fevereiro de 2018. Disponível em: <docs.microsoft.com/pt-br/sql/relational-databases/sql-server-transaction-locking-and-row-versioning-guide?view=sql-server-2017>. Acesso em: 10 de março de 2019.

SILBERSCHATZ, A; KORTH, H. F; SUDARSHAN, S. **Sistema de Banco de Dados**. 6. ed. Rio de Janeiro: Campus, 2012.