

PROGRAMAÇÃO EM BANCO DE DADOS

# TRATANDO DE **EXCEÇÃO**

MILTON GOYA



## LISTA DE QUADROS

Quadro 4.1 – Lista das exceções predefinidas.....	9
Quadro 4.2 – Valores do SQLCODE .....	11

EXEMPLO

**LISTA DE CÓDIGOS-FONTE**

Código-fonte 4.1 – Exemplo de erro de compilação.....	5
Código-fonte 4.2 – Exemplo de erro em tempo de execução.....	6
Código-fonte 4.3 – Sintaxe da seção EXCEPTION de um bloco PL/SQL.....	7
Código-fonte 4.4 – Exemplo de erro em tempo de execução.....	9
Código-fonte 4.5 – Exemplo de captura de erro.....	11
Código-fonte 4.6 – Exemplo do uso de exceção nomeado pelo desenvolvedor .....	12
Código-fonte 4.7 – Sintaxe da instrução OPEN .....	13
Código-fonte 4.8 – Exemplo de uso da instrução PRAGMA .....	14
Código-fonte 4.9 – Sintaxe do procedimento RAISE_APPLICATION_ERROR .....	14
Código-fonte 4.10 – Exemplo de uso de RAISE_APPLICATION_ERROR na seção EXCEPTION.....	15
Código-fonte 4.11 – Exemplo de uso de RAISE_APPLICATION_ERROR na seção executável .....	16
Código-fonte 4.12 – Exemplo de propagação de exceções para outros blocos.....	17

## SUMÁRIO

4 TRATAMENTO DE EXCEÇÃO .....	5
4.1 Introdução .....	5
4.2 Exceções predefinidas nomeadas.....	7
4.2.1 Funções para captura de erro .....	10
4.2.2 Exceções nomeadas pelo desenvolvedor .....	12
4.2.3 Associando exceções a erros-padrão do servidor .....	13
4.2.4 Procedimento raise_application_error .....	14
4.2.5 Propagando uma exceção para um bloco externo .....	16
CONCLUSÃO .....	18
REFERÊNCIAS .....	19

## 4 TRATAMENTO DE EXCEÇÃO

### 4.1 Introdução

Um dos objetivos de um bom desenvolvedor é que seus programas estejam corretos, atendam às necessidades para as quais foram projetados e que sempre funcionem. Infelizmente, erros em tempo de execução ocorrem e suas origens podem ser as mais diversas, desde uma falha de hardware até erros simples de programação. Não é fácil prever todos os possíveis erros, mas é possível preparar o seu programa para lidar com os erros mais significativos do PL/SQL. Damos o nome de tratamento de exceções ao processo por meio do qual o programa irá tratar os erros detectados por ele.

Para Feurstein et al. (2014), os erros podem ocorrer durante a compilação ou durante a execução do programa. Os erros de compilação estão associados à sintaxe das instruções: nome dos comandos, declaração das variáveis, organização dos comandos, entre outros. Nesse tipo de erro, o compilador da linguagem informa ao programador, durante a tentativa de executar uma instrução, a ocorrência do erro.

```
SELET * FROM emp;  
SP2-0734: unknown command beginning "SELET * FR..." - rest  
of line ignored.
```

Código-fonte 4.1 – Exemplo de erro de compilação  
Fonte: Elaborado pelo autor (2020)

No exemplo, o compilador retornou a mensagem de erro SP2-0734, porque o comando não foi digitado corretamente. O compilador esperava que o comando SELECT fosse digitado e, em vez disso, encontrou a palavra SELET e isso produziu o erro exibido. Apesar de ser possível identificar que um comando tenha sido digitado de forma errada, ainda não é possível capturar esse erro para tratá-lo automaticamente.

Para Dillon et al. (2013), erros em tempo de execução, que também são conhecidos como erros de **RUNTIME**, estão associados à utilização do programa.

Nesse caso, quando as exceções não são previstas e tratadas, o erro gerado interrompe o processamento e uma mensagem de erro é devolvida para a aplicação.

Vejamos um exemplo simples de erro em tempo de execução:

```
SET SERVEROUTPUT ON

DECLARE
    cinco NUMBER := 5;
BEGIN
    DBMS_OUTPUT.PUT_LINE ( cinco / ( cinco - cinco ) );
END;
/

DECLARE
*
ERROR at line 1:
ORA-01476: divisor is equal to zero
ORA-06512: at line 4
```

Código-fonte 4.2 – Exemplo de erro em tempo de execução  
Fonte: Elaborado pelo autor (2020)

No exemplo, estamos definindo uma variável numérica denominada CINCO e atribuindo o valor cinco a ela. Durante a execução do programa, executamos uma operação aritmética (CINCO menos CINCO) que produz o valor zero, em seguida, dividimos o valor da variável CINCO por zero. Como regra, não é possível efetuar divisões por zero, e isso gera um erro em tempo de execução. Note que não há nenhum erro de sintaxe no programa, mas, mesmo assim, o programa não funciona.

Para Puga et al. (2015), uma exceção é, na verdade, uma ocorrência não esperada ou diferente daquela programada para ser executada, em outras palavras, uma exceção é um erro.

Algumas linguagens de programação oferecem recursos para o tratamento de exceções. Normalmente, o tratamento de uma exceção consiste em identificar um erro em tempo de execução e tratá-lo de forma que: o usuário seja informado sobre o que está acontecendo e o programa não seja interrompido bruscamente.

O banco de dados Oracle possui várias exceções predefinidas, mas o desenvolvedor pode definir exceções personalizadas.

A seção de tratamento de exceções do bloco PL/SQL possui a seguinte estrutura:

```
EXCEPTION
  WHEN exceção1 [OR exceção2 ...] THEN
    comando1;
    comando2;
    ...
  [WHEN exceção3 [OR exceção4 ...] THEN
    comando1;
    comando2;
    ...]
  [WHEN OTHERS THEN
    comando1;
    comando2;
    ...]
```

Código-fonte 4.3 – Sintaxe da seção EXCEPTION de um bloco PL/SQL  
Fonte: ORACLE (2016)

#### Nessa estrutura:

**EXCEPTION** indica o início da seção de tratamento de exceções do bloco PL/SQL.

**exceção** indica o nome-padrão de uma exceção predefinida ou o nome de uma exceção definida pelo usuário, declarada dentro da seção declarativa.

**comando** indica uma ou mais instruções PL/SQL ou SQL.

**OTHERS** indica uma cláusula de tratamento de exceções opcional que intercepta qualquer exceção que não tenha sido explicitamente tratada.

## 4.2 Exceções predefinidas nomeadas

Para Puga et al. (2015), a Oracle predefiniu exceções para alguns erros mais comuns. Nesse caso, não existe a necessidade de que essas exceções sejam declaradas para serem utilizadas porque elas fazem parte do pacote STANDARD. Algumas dessas exceções são muito comuns e, por isso, além do identificador ORA, foram nomeadas. O nome atribuído a uma exceção também pode ser denominado HANDLER.

Nome da Exceção	Número do Erro do Servidor	Descrição

	<b>Oracle</b>	
ACCESS_INTO_NULL	ORA-06530	Tentativa de designar valores aos atributos de um objeto não inicializado.
CASE_NOT_FOUND	ORA-06592	Nenhuma das opções das cláusulas WHEN de uma instrução CASE foi selecionada, e não existe nenhuma cláusula ELSE.
COLLECTION_IS_NULL	ORA-06531	Tentativa de aplicar métodos de coleta diferentes de EXISTS a uma tabela aninhada inicializada ou um VARRAY.
CURSOR_ALREADY_OPEN	ORA-06511	Tentativa de abrir um cursor que já estava aberto.
DUP_VAL_ON_INDEX	ORA-00001	Tentativa de inserir um valor duplicado.
INVALID_CURSOR	ORA-01001	Operação com cursor inválido.
INVALID_NUMBER	ORA-01722	Falha na conversão de string de caracteres em número.
LOGIN_DENIED	ORA-01017	Logon no servidor Oracle com o nome de usuário ou senha inválidos.
NO_DATA_FOUND	ORA-01403	SELECT de uma única linha não retornou dados.
NOT_LOGGED_ON	ORA-01012	O programa PL/SQL executa uma chamada de banco de dados sem estar conectado ao servidor Oracle.
PROGRAM_ERROR	ORA-06501	O código PL/SQL tem um problema interno.
ROWTYPE_MISMATCH	ORA-06504	A variável de cursor host e a variável de cursor PL/SQL, envolvidas em uma designação, têm tipos de retorno incompatíveis.
STORAGE_ERROR	ORA-06500	O código PL/SQL está sem memória ou a memória está danificada.
SUBSCRIPT_BEYOND_COUNT	ORA-06533	Referência a uma tabela aninhada ou um elemento VARRAY, usando um número de índice maior que o número de elementos do conjunto.
SUBSCRIPT_OUTSIDE_LIMIT	ORA-06532	Referência a uma tabela aninhada ou a um elemento VARRAY, usando um número de índice que está fora da faixa válida (por exemplo, -1).



SYS_INVALID_ROWID	ORA-01410	Falha na conversão de uma string de caracteres em um ROWID universal, porque a string não representa um ROWID válido.
TIMEOUT_ON_RESOURCE	ORA-00051	Timeout enquanto o servidor Oracle esperava por um recurso.
TOO_MANY_ROWS	ORA-01422	SELECT de uma única linha retornou várias linhas.
VALUE_ERROR	ORA-06502	Erro aritmético, de conversão, de truncamento ou de restrição de tamanho.
ZERO_DIVIDE	ORA-01476	Tentativa de divisão por zero.

Quadro 4.1 – Lista das exceções predefinidas  
Fonte: ORACLE (2016)

Note que existe uma exceção predefinida denominada ZERO\_DIVIDE. Vamos usá-la no nosso código anterior.

```
SET SERVEROUTPUT ON

DECLARE
    cinco NUMBER := 5;
BEGIN
    DBMS_OUTPUT.PUT_LINE (cinco / ( cinco - cinco ));
EXCEPTION
    WHEN ZERO_DIVIDE THEN
        DBMS_OUTPUT.PUT_LINE ('Divisao por zero');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE ('Erro imprevisto');
END;
/
```

Código-fonte 4.4 – Exemplo de erro em tempo de execução  
Fonte: Elaborado pelo autor (2020)

O programa continua com o mesmo erro em tempo de execução, mas, ao invés do usuário receber um erro ORA-01476, recebe a mensagem de “Divisão por zero”. Isso é possível, porque adicionamos a seção EXCEPTION no programa para que possamos tratar o erro. No nosso exemplo, adicionamos dois tratamentos de erro ZERO\_DIVIDE e OTHERS. Agora, caso ocorra um erro de divisão por zero, a seção de EXCEPTION capturará o erro por meio da exceção predefinida ZERO\_DIVIDE e exibirá a mensagem; também passamos a capturar qualquer outro erro por meio da exceção predefinida OTHERS. No nosso exemplo, só estamos

exibindo mensagens de erro, mas poderíamos executar outras operações em conjunto com elas.

A seção de tratamento de exceção captura somente as exceções especificadas; quaisquer outras exceções não serão capturadas, exceto se for utilizado o HANDLER de exceção OTHERS. Esse HANDLER captura qualquer exceção que ainda não esteja tratada. Por essa razão, OTHERS é o último HANDLER de exceção definido.

Podem ser estabelecidos vários HANDLERS de exceção para o bloco, cada um deles com o seu próprio conjunto de ações.

Quando ocorre uma exceção, o código PL/SQL processa somente um HANDLER antes de sair do bloco. As exceções não podem aparecer em instruções de atribuição ou instruções SQL.

#### 4.2.1 Funções para captura de erro

Para a Oracle (2016), quando ocorre uma exceção, pode-se identificar o código ou a mensagem de erro associado, usando duas funções SQLCODE e SQLERRM e, com base nos valores do código ou mensagem, você pode decidir qual ação subsequente tomar com base no erro.

SQLERR é uma função numérica que devolve o código do erro ocorrido. Enquanto SQLCODE é uma função caractere que devolve o texto da mensagem do erro.

Quando uma exceção é capturada no HANDLER de exceção WHEN OTHERS, pode-se usar um conjunto de funções genéricas para identificar esses erros.

Vejamos um exemplo de uso:

```
CREATE TABLE erros
(usuario VARCHAR2(30),
 data      DATE,
 cod_erro  NUMBER,
 msg_erro  VARCHAR2(100));

SET SERVEROUTPUT ON
```

```
DECLARE
    cod    erros.cod_erro%TYPE;
    msg    erros.msg_erro%TYPE;
    cinco  NUMBER := 5;
BEGIN
    DBMS_OUTPUT.PUT_LINE (cinco / ( cinco - cinco ));
EXCEPTION
    WHEN ZERO_DIVIDE THEN
        cod := SQLCODE;
        msg := SUBSTR(SQLERRM, 1, 100);
        insert into erros values (USER, SYSDATE, cod, msg);
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE ('Erro imprevisto');
END;
/
```

Código-fonte 4.5 – Exemplo de captura de erro  
Fonte: Elaborado pelo autor (2020)

No exemplo, os valores de SQLCODE e SQLERRM são atribuídos a variáveis que, em seguida, são usadas em uma instrução SQL. Se, mais tarde, você consultar a tabela ERROS, verá o nome do usuário que executou o programa PL/SQL, o horário do erro, seu código e a mensagem de erro associada a esse código.

Talvez você tenha percebido que usamos a função SUBSTR para truncar o tamanho da mensagem de erro para um tamanho conhecido. Fizemos isso porque não sabemos o tamanho final da mensagem que iremos atribuir à variável e, com isso, evitamos introduzir outro erro no programa.

A seguir, você pode ver os valores que podem ser assumidos pelo SQLCODE.

Valor SQLCODE	Descrição
0	Nenhuma exceção encontrada.
1	Exceção definida pelo usuário.
+100	Exceção NO_DATA_FOUND.
Número Negativo	Número de erro do servidor Oracle.

Quadro 4.2 – Valores do SQLCODE  
Fonte: ORACLE (2016)

#### 4.2.2 Exceções nomeadas pelo desenvolvedor

Para Feuerstein et al. (2014), as exceções nomeadas pelo desenvolvedor são similares às exceções predefinidas, e sua diferença é que precisam ser declaradas na seção DECLARE e são chamadas por meio da instrução RAISE.

Vejamos um exemplo de uso:

```
DECLARE
    e_meu_erro EXCEPTION;
    emprec emp%ROWTYPE;
    CURSOR cursor_emp IS
        SELECT empno, ename
        FROM emp
        WHERE empno = 1111;
BEGIN
    OPEN cursor_emp;
    LOOP
        FETCH cursor_emp INTO emprec.deptno, emprec.sal;
        IF cursor_emp%NOTFOUND THEN
            RAISE e_meu_erro;
        END IF;
        DBMS_OUTPUT.PUT_LINE ('Codigo : ' || emprec.empno);
        DBMS_OUTPUT.PUT_LINE ('Nome      : ' || emprec.ename);
        EXIT WHEN cursor_emp%NOTFOUND;
    END LOOP;
EXCEPTION
    WHEN E_MEU_ERRO THEN
        DBMS_OUTPUT.PUT_LINE ('Codigo nao cadastrado');
        ROLLBACK;
END;
/
```

Código-fonte 4.6 – Exemplo do uso de exceção nomeado pelo desenvolvedor  
Fonte: Elaborado pelo autor (2020)

Observe, no exemplo, que, na seção DECLARE, estamos definindo a exceção E\_MEU\_ERRO, usando o atributo EXCEPTION. Na seção executável, estamos testando, com a cláusula IF, se %NOTFOUND retornou o valor TRUE, ou seja, se a consulta não retornar dados, o programa acionará a exceção E\_MEU\_ERRO. A seção EXCEPTION capturará o erro, emitirá a mensagem “Codigo nao cadastrado” e desfazá quaisquer alterações que ainda não tenham sido gravadas com a instrução ROLLBACK.

#### 4.2.3 Associando exceções a erros-padrão do servidor

Para Dillon et al (2013), algumas exceções declaradas pelo usuário podem ser associadas a erros Oracle predefinidos, mas não nomeados. Podemos associar essas exceções a um nome, da seguinte maneira:

```
PRAGMA EXCEPTION_INIT(nome_exceção, código_Oracle_erro);
```

Código-fonte 4.7 – Sintaxe da instrução OPEN

Fonte: ORACLE (2016)

Nesse exemplo:

**nome\_exceção** é o nome da exceção. O nome da exceção deve ser declarado na área declarativa (DECLARE).

**código\_Oracle\_erro** é o código do erro dentro do padrão Oracle que será associado à exceção declarada.

Para Puga et al. (2015), na área de exceções, deve-se fazer referência à exceção declarada dentro da rotina de tratamento de exceção correspondente.

Para a Oracle (2016), o PRAGMA EXCEPTION\_INIT é uma diretiva de compilação que informa ao compilador a necessidade de associar um nome de exceção a um número de erro do Oracle. Isso permite que possa ser consultada qualquer exceção interna por nome, e criado um HANDLER específico para ela.

PRAGMA (também chamado pseudoinstruções) é uma palavra-chave que significa que a instrução é uma diretiva de compilador que não é processada ao executar o bloco PL/SQL. Em vez disso, orienta o compilador do PL/SQL para interpretar todas as ocorrências do nome da exceção dentro do bloco como o número de erro do Oracle Server associado.

Vejamos um exemplo:

```
DECLARE
  e_meu_erro EXCEPTION;
  PRAGMA EXCEPTION_INIT (e_meu_erro, -2292);
BEGIN
  DELETE FROM dept
  WHERE deptno = 10;
  COMMIT;
EXCEPTION
  WHEN e_meu_erro THEN
    DBMS_OUTPUT.PUT_LINE ('Integridade Referencial Violada');
```

```
ROLLBACK;  
END;  
/
```

Código-fonte 4.8 – Exemplo de uso da instrução PRAGMA  
Fonte: Elaborado pelo autor (2020)

O exemplo cria, na seção DECLARE, uma exceção de nome E\_MEU\_ERRO usando o atributo EXCEPTION. Logo em seguida, associamos essa exceção ao código de erro -2292, usando a definição PRAGMA EXCEPTION\_INIT. O erro -2292 é o código usado para indicar uma violação de integridade referencial. Na seção executável, fizemos a tentativa de apagar o departamento 10, gerando um erro de integridade referencial. A seção EXCEPTION captura o erro por meio da exceção declarada E\_MEU\_ERRO, exibe a mensagem “Integridade Referencial Violada” e desfaz todas as alterações efetuadas até o momento, usando a instrução ROLLBACK.

#### 4.2.4 Procedimento raise\_application\_error

Para a Oracle (2016), o procedimento RAISE\_APPLICATION\_ERROR é utilizado para comunicar uma exceção predefinida interativamente, retornando um código ou uma mensagem de erro não padronizada. Com RAISE\_APPLICATION\_ERROR, é possível relatar erros para a aplicação e evitar o retorno de exceções não tratáveis.

```
RAISE_APPLICATION_ERROR (numero_erro, mensagem [, {TRUE | FALSE}]);
```

Código-fonte 4.9 – Sintaxe do procedimento RAISE\_APPLICATION\_ERROR  
Fonte: ORACLE (2016)

##### Nesse caso:

**numero\_erro** é um número especificado pelo usuário para a exceção entre –20000 e –20999.

**mensagem** é a mensagem especificada pelo usuário para a exceção. Trata-se de uma string de caracteres com até 2.048 bytes.

**TRUE | FALSE** é um parâmetro Booleano opcional (Se TRUE, o erro será colocado na pilha de erros anteriores. Se FALSE, o default, o erro substituirá todos os erros anteriores.).

O procedimento `RAISE_APPLICATION_ERROR` pode ser usado tanto na seção de exceção quanto na seção executável.

Vejam os exemplos de uso de `RAISE_APPLICATION_ERROR` na seção `EXCEPTION`.

```
DECLARE
    cinco NUMBER := 5;
BEGIN
    DBMS_OUTPUT.PUT_LINE (cinco / ( cinco - cinco ));
EXCEPTION
    WHEN ZERO_DIVIDE THEN
        RAISE_APPLICATION_ERROR (-20901, 'Erro aritmetico.
Reveja o programa');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE ('Erro imprevisto');
END;
```

```
/

ERROR at line 1:
ORA-20901: Erro aritmetico. Reveja o programa
```

Código-fonte 4.10 – Exemplo de uso de `RAISE_APPLICATION_ERROR` na seção `EXCEPTION`  
Fonte: Elaborado pelo autor (2020)

No exemplo, devido a um erro aritmético, ocorre uma divisão por zero; esse erro é capturado na seção `EXCEPTION` pelo `HANDLER ZERO_DIVIDE`. Como estamos usando o procedimento `RAISE_APPLICATION_ERROR`, a mensagem de erro foi exibida no formato-padrão Oracle “ORA-20901: Erro aritmético. Reveja o programa”.

Também podemos usar `RAISE_APPLICATION_ERROR` na seção executável, veja abaixo:

```
DECLARE
    e_meu_erro EXCEPTION;
    PRAGMA EXCEPTION_INIT (e_meu_erro, -2292);
BEGIN
    DELETE FROM dept
    WHERE deptno = 33;
    IF SQL%NOTFOUND THEN
        RAISE_APPLICATION_ERROR (-20901, 'Departamento
Inexistente');
        ROLLBACK;
    END IF;
    COMMIT;
EXCEPTION
    WHEN e_meu_erro THEN
```

```
DBMS_OUTPUT.PUT_LINE ('Integridade Referencial Violada');  
ROLLBACK;  
END;
```

```
/
```

ERROR at line 1:  
ORA-20901: Departamento Inexistente

Código-fonte 4.11 – Exemplo de uso de RAISE\_APPLICATION\_ERROR na seção executável  
Fonte: Elaborado pelo autor (2020)

No exemplo, tentamos apagar os dados de um departamento que não está cadastrado em nossa base de dados. O aplicativo retorna o valor TRUE para SQL%NOTFOUND e, com isso, acionamos o procedimento RAISE\_APPLICATION\_ERROR que irá exibir a mensagem de erro no formato-padrão Oracle “ORA-20901: Departamento Inexistente”.

#### 4.2.5 Propagando uma exceção para um bloco externo

Para a Oracle (2016), ao invés de se capturar uma exceção dentro do bloco PL/SQL, é possível propagar a exceção para permitir que ela seja tratada pelo ambiente de chamada. Cada ambiente de chamada tem seu próprio modo de exibir e acessar erros.

Quando um sub-bloco trata uma exceção, ele termina normalmente e o controle retorna ao bloco delimitado imediatamente após a instrução END do sub-bloco.

Entretanto, se o código PL/SQL criar uma exceção e o bloco atual não tiver um HANDLER para ela, a exceção propagará em blocos delimitados sucessivos, até localizar um HANDLER. Se nenhum desses blocos tratar a exceção, o resultado será uma exceção não tratável no ambiente de HOST (chamador).

Quando a exceção propaga para um bloco delimitado, as ações executáveis restantes desse bloco são ignoradas.

Uma vantagem desse comportamento é que se podem delimitar instruções que exijam seus próprios tratamentos de erro exclusivos em seu próprio bloco, enquanto deixam o tratamento de exceção mais geral para o bloco delimitado.

Vejamos um exemplo:



```
DECLARE
    cod    erros.cod_erro%TYPE;
    msg    erros.msg_erro%TYPE;
    cinco  NUMBER := 5;
BEGIN
    BEGIN
        DELETE FROM dept
            WHERE deptno = 10;
    EXCEPTION
        WHEN ZERO_DIVIDE THEN
            DBMS_OUTPUT.PUT_LINE ('Erro no bloco interno');
    END;
    DBMS_OUTPUT.PUT_LINE (cinco / ( cinco - cinco ));
EXCEPTION
    WHEN OTHERS THEN
        cod := SQLCODE;
        msg := SUBSTR(SQLERRM, 1, 100);
        insert into erros values (USER, SYSDATE, cod, msg);
END;
/
```

Código-fonte 4.12 – Exemplo de propagação de exceções para outros blocos  
Fonte: Elaborado pelo autor (2020)

No exemplo, temos dois blocos PL/SQL aninhados, você notará que a instrução DELETE está no bloco mais interno e a divisão por zero está no bloco mais externo. Ao tentar efetuar a exclusão do departamento 10, ocorre um erro de violação de integridade referencial. O programa procura, dentro do bloco mais interno, uma forma de tratar esse erro, caso não encontre, transfere o erro para a seção de EXCEPTION do bloco mais externo, que insere o código de erro na tabela ERROS. Caso esse erro não fosse tratado no bloco mais externo, o programa propagaria o erro para o ambiente chamador.

## CONCLUSÃO

Tratar as exceções na execução de qualquer bloco de programação é indispensável a qualquer sistema que se proponha a solucionar problemas com qualidade. Códigos-fonte são gerados por seres humanos, que são falhos, assim como seus usuários e, além disso, exceções ao processamento podem ser causadas por condições adversas de hardware e outros fatores externos à programação. O tratamento adequado de tais causas pode impedir que o procedimento seja encerrado de forma abrupta ou impedir que o usuário fique confuso quanto à conclusão ou não de um procedimento.

## REFERÊNCIAS

DILLON, Sean; BECK, Christopher; KYTE, Thomas; KALLMAN, Joel; ROGERS, Howard. **Beginning Oracle Programming**. Apress, 2013.

FEUERSTEIN, Steven; PRIBYL, Bill. **Oracle PL/SQL Programming**. O'Reilly Media, 2014.

ORACLE. **Oracle Database: PL/SQL Language Reference 12c Release 2 (12.2) B28370-05**. Oracle Press, 2016.

PUGA, Sandra; FRANÇA, Edson; GOYA, Milton. **Banco de dados**. São Paulo: Pearson, 2015.