CONSULTAS EM BANCO DE DADOS

Anderson Sene Gonçalves

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Intersecção de conjuntos em SQL

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Explicar a operação de intersecção em álgebra relacional.
- Exemplificar a operação de intersecção em banco de dados relacional.
- Reconhecer a implementação da operação de intersecção em SQL.

Introdução

A operação de grandes bases de dados requer o conhecimento de boas técnicas para obter um bom desempenho e, principalmente, o resultado esperado em cada operação. Para isso, torna-se necessário compreender maneiras de elaborar determinadas funções para construção consultas bem estruturadas, como a partir dos conceitos de álgebra relacional.

Neste capítulo, você observará os conceitos de intersecção aplicados a partir de uma consulta em banco de dados, além da importância de conhecer e saber aplicar essa operação binária da álgebra relacional, o que contribuirá diretamente na construção de grandes consultas em banco de dados a fim de obter as informações desejadas com o melhor tempo e desempenho possíveis. Em suma, você verá tanto as questões teóricas quanto as práticas em relação à construção de consultas em banco de dados por meio do operador de intersecção.

1 Intersecção em álgebra relacional

A álgebra relacional refere-se a uma linguagem formal de consulta, cujos conceitos teóricos estão relacionados diretamente às linguagens de consulta em bancos de dados. Os operadores de álgebra relacional possibilitam realizar a projeção, a seleção, a diferença e a intersecção, a última justamente o objeto de estudo deste capítulo.

A aplicação de um operador de álgebra relacional requer a existência de um conjunto de operações, em que se tem a "entrada" como um ou mais conjuntos de valores e, quando aplicado um dos operadores lógicos, apresentará uma "saída" composta de um novo resultado proveniente do operador aplicado.

Existem diversas formas de aplicar os operadores para alcançar um novo conjunto de dados. Para Ramakrishnan e Gehrke (2008), a álgebra relacional é mais uma linguagem de consulta formal com base em um grupo de operadores para a manipulação relações, que pode ser tão poderosa quanto o cálculo.

Neste capítulo, você estudará como o operador de intersecção está associado à álgebra relacional, como pode ser visto na Figura 1, na qual há a representação do diagrama de Venn, o que representa e como ocorre a intersecção em um conjunto de dados.

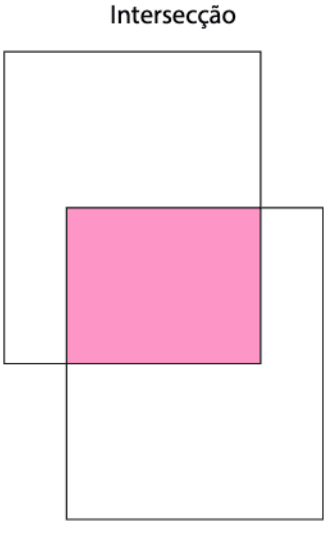

Figura 1. Diagramas de Venn para uma operação de intersecção.

Fonte: Adaptada de Mannino (2008).

De acordo com Mannino (2008), um exemplo prático da operação de intersecção é a listagem de alunos que estão matriculados em duas instituições de ensino. Assim, a intersecção entre dois conjuntos de dados gerará um terceiro conjunto de dados com as linhas de ambos os conjuntos de dados.

Sua representação gráfica é dada por:

- simbologia: ∩;
- sintaxe: **(Relação 1) ∩ (Relação 2).**

Para Ramakrishnan e Gehrke (2008), uma propriedade fundamental dos operadores de algébricos é a de aceitar uma ou duas instâncias de relação como seu argumento e retornar uma instância de relação como seu resultado. A operação de intersecção em uma base de dados relacional é representada por **r1** ∩ **r2**, onde:

- **r1**: representa um conjunto de tuplas;
- ∩: é o símbolo da operação de intersecção;
- **r2**: representa outro conjunto de tuplas.

Ainda conforme Ramakrishnan e Gehrke (2008), a intersecção: **R** ∩ **M** tem como retorno uma instância de relação que contém todas as tuplas que ocorrem em ambas, **R** e **M**. As relações **R** e **M** devem ser compatíveis à união, e o esquema do resultado é definido de forma idêntica ao esquema de **R**.

Para ilustrar a operação de intersecção em álgebra relacional, considere os dois conjuntos de dados ilustrados na Figura 2, denominados "**R**" e "**M**".

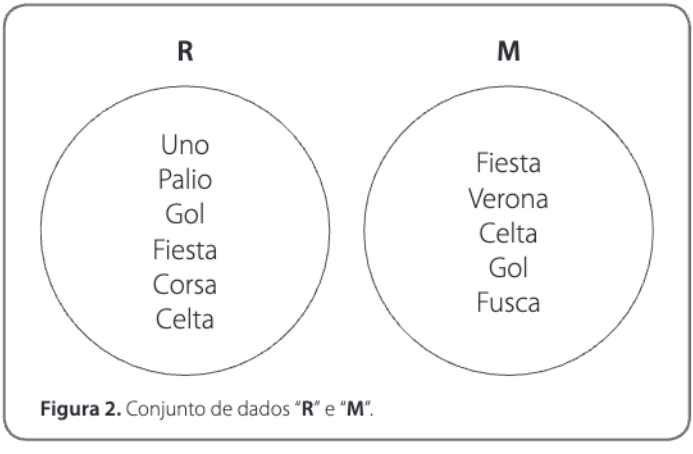

Figura 2. Conjunto de dados "**R**" e "**M**".

Observe que, nos dois conjuntos de dados, existem valores presentes em ambos os conjuntos "**R**" e "**M**", enquanto outros existem somente no conjunto "**R**" ou "**M**".

Diante disso, ao se realizar a operação "**R** ∩ **M**", somente serão exibidos os valores presentes em ambos os conjuntos, como pode ser visto na Figura 3.

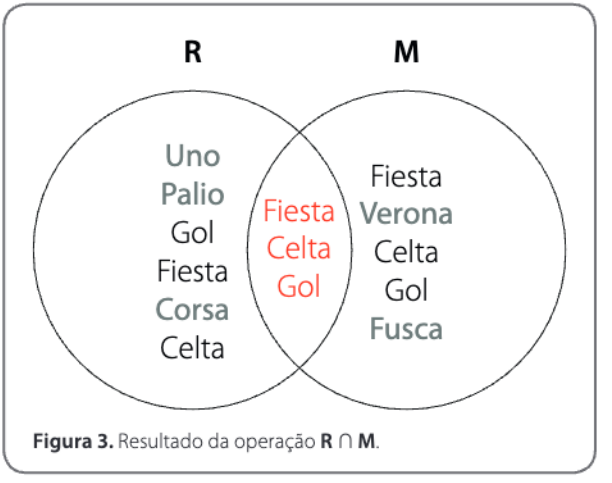

Figura 3. Resultado da operação **R** ∩ **M**.

Portanto, podemos afirmar que conhecer os operadores da álgebra relacional contribuirá diretamente na construção de consultas em bancos de dados, bem como a função desempenhada pelo operador de intersecção, que, como visto, consiste em retornar os valores existentes em ambos os conjuntos de dados.

Fique atento

Você sabia que a operação de intersecção, além de compreender um conceito de álgebra relacional e ser aplicada a bancos de dados relacionais, tem um emprego semelhante em bancos de dados geográficos?

Para saber mais, leia o livro *Banco de Dados Geográficos*, organizado pelos pesquisadores Marco Antonio Casanova, Gilberto Câmara, Clodoveu A. Davis Jr., Lúbia Vinhas e Gilberto Ribeiro de Queiroz, disponível livremente na internet.

2 Intersecção em banco de dados relacional

Para o armazenamento de informações, as empresas fazem uso de um programa intitulado Sistema Gerenciador de Banco de Dados (SGBD), ferramenta que tem como funções armazenar, gerenciar e manipular os dados gravados.

Atualmente, existem diversos pacotes de *software* que realizam a função de um SGBD, como os *software* proprietários (aqueles cujo uso depende de aquisição de licença), como Microsoft SQL Server e Oracle, e os *open source* (aqueles sob licença pública, como a GNU), como PostgreSQL, MariaDB e MySQL.

Além disso, os SGBD citados operam sob um modelo de bancos de dados chamado modelo relacional, que diz respeito à existência de relações entre as tabelas, as quais permitem a criação de restrições de dados de uma tabela em relação à outra.

Com a criação de diversos SGBD, definiu-se uma linguagem específica para a criação e a manipulação desses dados: a SQL (acrônimo de Structured Query Language — Linguagem de Consulta Estruturada), que torna possível criar tabelas, inserir e atualizar valores e remover tuplas, além de manipular os dados por meio das consultas na base de dados.

De acordo com Elmasri e Navathe (2011), a definição de banco de dados tem as seguintes propriedades implícitas:

- representação de aspectos do mundo real;
- coleção lógica e coerente de dados com algum significado inerente;
- projeto, construção e povoamento por dados, atendendo a uma proposta específica.

Um banco de dados é composto por um conjunto de tabelas, as quais têm por objetivo definir a estrutura dos dados que serão armazenados. As tabelas, por sua vez, são formadas por atributos que especificam informações a respeito das primeiras, e os atributos dispõem de tipos de dados, associados diretamente ao tipo de informação que será armazenada naquele atributo.

Quando se aplica a operação de intersecção em uma base de dados, o SGBD busca comparar todas as tuplas dos conjuntos de dados envolvidos, para localizar as correspondências existentes simultaneamente nos dois conjuntos de dados.

Com base nisso, considere o banco de dados de uma universidade. Nesse modelo, há uma tabela denominada `professor`, que será responsável por armazenar as informações relacionadas a todos os docentes da instituição, enquanto a tabela `coordenador` terá a função de armazenar os dados de todos os professores que, além da docência, exercem a função de coordenador de curso.

A seguir, há a representação da tabela `professor`, que tem cinco atributos (`ID`, `nome`, `dataNascimento`, `curso` e `dataInicio`) e um total de 10 tuplas ou registros (as linhas da tabela). Nessa tabela, consta a representação de informações relacionadas aos professores:

- `ID`: atributo único e usado para identificar cada tupla;
- `nome`: nome do professor;
- `dataNascimento`: data de nascimento do professor;
- `curso`: curso a que o professor está associado;
- `dataInicio`: data em que o professor começou a lecionar na instituição.

Tabela `professor`

ID	nome	dataNascimento	curso	dataInicio
990	Nanni Demchen	13/06/1973	Administração	04/11/2015
991	Dagny Canner	09/09/1985	Ciências Contábeis	14/06/2018
992	Rhea Farlam	05/10/1990	Análise e Desenvolvimento de Sistemas	23/05/2016
993	Rheta Brik	13/09/1983	Direito	05/03/2019
994	Caz Bracken	10/08/1982	Análise e Desenvolvimento de Sistemas	11/10/2015
995	Larissa Di Claudio	14/02/1989	Análise e Desenvolvimento de Sistemas	01/05/2018
996	Jinny Heady	08/03/1987	Direito	09/11/2018

ID	nome	dataNascimento	curso	dataInicio
997	Alysia Klimas	10/03/1979	Análise e Desenvolvimento de Sistemas	02/06/2015
998	Dalenna Verbeek	29/08/1973	Marketing	02/12/2017
999	Lorianna Aisthorpe	09/03/1969	Ciências Contábeis	18/07/2018

Já na tabela `coordenador`, a seguir, estão representados cinco atributos (`ID`, `nome`, `dataNascimento`, `curso` e `dataInicio`) e um total de quatro tuplas ou registros (linhas da tabela). Essa tabela visa a armazenar as informações dos professores que também são coordenadores, cujos atributos dizem respeito a:

- `ID`: atributo único e usado para identificar cada tupla;
- `nome`: nome do professor;
- `dataNascimento`: data de nascimento do professor;
- `curso`: curso que o professor coordena;
- `dataInicio`: data em que o professor começou a coordenar o curso.

Tabela `coordenador`

ID	nome	dataNascimento	curso	dataInicio
990	Nanni Demchen	13/06/1973	Administração	04/11/2015
997	Alysia Klimas	10/03/1979	Análise e Desenvolvimento de Sistemas	02/06/2015
998	Dalenna Verbeek	29/08/1973	Marketing	02/12/2017
999	Lorianna Aisthorpe	09/03/1969	Ciências Contábeis	18/07/2018

Ao reproduzirmos a intersecção de `professor` ∩ `coordenador`, o resultado será uma representação das tuplas existentes tanto na tabela `professor` quanto na tabela `coordenador`, conforme visto na tabela a seguir.

ID	nome	dataNascimento	curso	dataInicio
990	Nanni Demchen	13/06/1973	Administração	04/11/2015
997	Alysia Klimas	10/03/1979	Análise e Desenvolvimento de Sistemas	02/06/2015
998	Dalenna Verbeek	29/08/1973	Marketing	02/12/2017
999	Lorianna Aisthorpe	09/03/1969	Ciências Contábeis	18/07/2018

Na Figura 4, apresenta-se um exemplo abordado por Mannino (2008), em que são exibidas duas tabelas de alunos, chamadas `aluno1` e `aluno2`, com as mesmas quantidades de atributos e com os respectivos tipos de dados. Vale ressaltar que, para a realização de intersecção em dois conjuntos de dados, os dois conjuntos de dados devem ter a mesma representação referente à quantidade de atributos e aos seus respectivos tipos de dados.

Essa teoria também é defendida por Mannino (2008), que aponta, apesar da possibilidade de determinar se duas linhas são idênticas apenas comparando cada `CPFAluno`, todas as colunas são comparadas em virtude da maneira como os operadores são projetados.

CPFAluno	Sobrenome Aluno	Cidade Aluno	UFAluno	Especializacao	Turma	Media Aluno
123-45-6789	WELLS	SEATTLE	WA	SI	FR	3,00
124-56-7890	NORBERT	BOTHELL	WA	FINAN	JR	2,70
234-56-7890	KENDALL	TACOMA	WA	CONTB	JR	3,50

CPFAluno	Sobrenome Aluno	Cidade Aluno	UFAluno	Especializacao	Turma	Media Aluno
123-45-6789	WELLS	SEATTLE	WA	SI	FR	3,00
995-56-3490	BAGGINS	AUSTIN	TX	FINAN	JR	2,90
111-56-4490	WILLIAMS	SEATTLE	WA	CONTB	JR	3,40

Figura 4. Tabelas `aluno1` e `aluno2`.

Fonte: Adaptada de Mannino (2008).

Com base nas duas tabelas da Figura 4, Mannino (2008) expõe que a operação de intersecção recupera apenas as linhas em comum, cujo resultado é dado na Figura 5.

A linha exibida na tabela da Figura 5 diz respeito à única tupla que constava tanto na tabela `aluno1` quanto na tabela `aluno2`. Por isso, ao realizar **r1 ∩ r2**, obtém-se essa tabela com apenas uma tupla.

CPFAluno	Sobrenome Aluno	Cidade Aluno	UFAluno	Especializacao	Turma	Media Aluno
123-45-6789	WELLS	SEATTLE	WA	SI	FR	3,00

Figura 5. Resultado da operação `aluno1 INTERSECT aluno2`.
Fonte: Adaptada de Mannino (2008).

3 Implementação da operação de intersecção em SQL

Agora, você verá como realizar a aplicação do comando de intersecção em uma consulta de banco de dados. Para tanto, será necessário fazer uso do comando `SELECT` para consultar todos ou partes dos atributos de uma tabela. De acordo com Barbosa e Freitas (2018), a partir do comando `SELECT`, pode-se realizar consultas ao banco de dados, retornando registros, parâmetros, composições, bem como muitos outros tipos de consultas. Além desse comando, veremos como trabalhar em conjunto o comando `INTERSECT` com consultas usando também a cláusula `WHERE`.

Para tanto, considere as tabelas `professor` e `coordenador` citadas anteriormente, nas quais temos, respectivamente, os registros dos docentes e as informações dos professores que também exercem a coordenação de curso. A informação que precisamos levantar é a relação de todos os professores que também são coordenadores. Para isso, pode-se partir da representação **r1 ∩ r2** para alcançar o resultado desejado.

Tem-se, portanto, a seguinte *query* (consulta):

```
SELECT * FROM professor
INTERSECT
SELECT * FROM coordenador;
```

Devemos salientar que:

- SELECT: é o comando responsável por consultar as tuplas de uma tabela;
- * (asterisco): indica que todos os atributos da tabela serão apresentados na consulta;
- FROM: aponta qual será a tabela consultada;
- professor/coordenador: são as tabelas consultadas;
- INTERSECT: comando responsável por realizar justamente a intersecção das tuplas das duas tabelas.

Para o funcionamento do INTERSECT, os atributos da tabela devem apresentar as mesmas especificações, tanto da quantidade de atributos quanto da definição dos respectivos tipos de dados.

Além disso, podemos combinar as demais especificações de um comando SELECT em uma *query* usando o INTERSECT, como exibido no exemplo a seguir. Dessa maneira, somente as tuplas da tabela professor com o ID maior que 995 serão avaliadas na hora de realizar a intersecção com a tabela coordenador.

```
SELECT * FROM professor
    WHERE ID > 995
INTERSECT
SELECT * FROM coordenador;
```

Na tabela a seguir, ilustra-se o resultado apresentado pela consulta citada. Ao compararmos a tabela professor com a tabela coordenador, podemos perceber que somente a tupla de ID igual a 990 deixou de aparecer no resultado da intersecção das duas tabelas. Isso ocorreu justamente pela adição da cláusula WHERE ID > 995 ao primeiro SELECT, ou seja, essa condição invalida o retorno da tupla 990, já que 990 é inferior a 995.

ID	nome	dataNascimento	curso	dataInicio
997	Alysia Klimas	10/03/1979	Análise e Desenvolvimento de Sistemas	02/06/2015
998	Dalenna Verbeek	29/08/1973	Marketing	02/12/2017
999	Lorianna Aisthorpe	09/03/1969	Ciências Contábeis	18/07/2018

A cláusula `WHERE` pode também estar disposta tanto no primeiro quanto no segundo `SELECT`:

```
SELECT * FROM professor
    WHERE ID > 995
INTERSECT
SELECT * FROM coordenador
    WHERE ID <= 998;
```

Essa consulta apresenta tanto a cláusula do `WHERE` tanto no primeiro quanto no segundo `SELECT`, caso em que as tuplas consideradas na intersecção dessas duas tabelas serão somente as que atendem às condições do `WHERE` de ambos os `SELECT` e que estiverem nas duas tabelas.

A tabela a seguir ilustra a execução da consulta apresentada, que apresenta duas cláusulas `WHERE`. Como pode ser visto, com a adição da condição de `ID` menor ou igual a `998`, uma das tuplas deixou de ser exibida no resultado da intersecção das duas tabelas.

ID	nome	dataNascimento	curso	dataInicio
997	Alysia Klimas	10/03/1979	Análise e Desenvolvimento de Sistemas	02/06/2015
998	Dalenna Verbeek	29/08/1973	Marketing	02/12/2017

Por fim, e não menos importante, também podemos adicionar a ordenação baseada no resultado da intersecção de duas *query*, o que permite ordenar de modo crescente ou decrescente as tuplas resultadas de uma consulta.

Na consulta exibida a seguir, foi adicionada, após a última condição do `WHERE`, a função de ordenação `ORDER BY 3`, o que significa dizer que as tuplas exibidas após a intersecção das duas tabelas serão ordenadas em ordem crescente a partir da terceira coluna da tabela resultante.

```
SELECT * FROM professor
    WHERE ID > 995
INTERSECT
SELECT * FROM coordenador
    WHERE ID <= 998
ORDER BY 3;
```

ID	nome	dataNascimento	curso	dataInicio
998	Dalenna Verbeek	29/08/1973	Marketing	02/12/2017
997	Alysia Klimas	10/03/1979	Análise e Desenvolvimento de Sistemas	02/06/2015

O resultado da consulta fez com que as tuplas sofressem alteração na ordem de exibição. Afinal, na coluna `3` (`dataNascimento`), a tupla de `ID` igual a `998` nasceu em `29/08/1973`, enquanto a tupla de `ID 997` nasceu em `10/03/1979`, ou seja, ao ordenarmos de forma crescente (do menor para o maior), a tupla `ID 998` assumiu a posição antes da tupla `ID 997`.

Exemplo

Para inverter a ordem de ordenação de um `ORDER BY`, basta adicionar o modificador `DESC` logo após o número de índice do atributo que você deseja ordenar.

```
SELECT * FROM professor
    WHERE ID > 995
INTERSECT
SELECT * FROM coordenador
    WHERE ID <= 998
ORDER BY 3 DESC;
```

Ramakrishnan e Gehrke (2008) abordam um exemplo, representado na Figura 6, sobre o uso do comando `INTERSECT`, em que se pede para encontrar os nomes dos marinheiros que reservaram um barco vermelho e um barco verde. O exemplo demonstra como e para que se deve utilizar o comando `INTERSECT`.

id-marin	*nome-marin*	*avaliação*	*idade*
22	Dustin	7	45,0
31	Lubber	8	55,5
58	Rusty	10	35,0

id-marin	*nome-marin*	*avaliação*	*idade*
28	Yuppy	9	35,0
31	Lubber	8	55,5
44	Guppy	5	35,0
58	Rusty	10	35,0

id-marin	*id-barco*	*dia*
22	101	10/10/96
58	103	11/12/96

Figura 6. Representação das instâncias M1 e M2 de marinheiros, e R1 de reservas.
Fonte: Adaptada de Ramakrishnan e Gehrke (2008).

Para obtermos as informações referentes aos marinheiros, há uma alternativa para desenvolver essa consulta usando as tabelas envolvidas com o operador lógico `AND`, conforme visto no exemplo de consulta elaborado por Ramakrishnan e Gehrke (2008).

```
SELECT
    M.nome-marin
FROM
    Marinheiros M
    , Reservas R1
    , Barcos B1
    , Reservas R2
    , Barcos B2
WHERE
    M.id-marin = R1.id-marin
    AND R1.id-barco = B1.id-barco
    AND M.id-marin = R2.id-marin
    AND R2.id-barco = B2.id-barco
    AND B1.cor = 'vermelho'
    AND B2.cor = 'verde';
```

Essa consulta representa uma alternativa para as tuplas das mesmas tabelas que usam condições distintas no `WHERE`, ou seja, nessa *query*, existe a duplicação na inclusão das tabelas que estão sendo usadas para elaborar a consulta na base de dados. Observe que as tabelas `reservas` e `barcos` estão duplicadas duas vezes cada uma e, nas condições do `WHERE`, há a replicação dos filtros alterando para cada uma das duas tabelas duplicadas.

Essa forma de escrever a consulta acaba tornando a sua interpretação confusa e mais difícil de compreender (RAMAKRISHNAN; GEHRKE, 2008). Por isso, há justamente a alternativa de desmembrá-la em duas e realizar a `INTERSECT` dos dois conjuntos, embora isso ainda fará com que sejam retornados apenas os valores existentes em ambas as consultas. Para resolver essa questão, basta adicionar a cláusula `ALL` logo depois do `INTERSECT` para que as linhas duplicadas nas duas tabelas também sejam exibidas no terceiro conjunto de valores gerado a partir da intersecção.

Referências

BARBOSA, F. F. M.; FREITAS, P. H. C. *Modelagem e desenvolvimento de banco de dados*. Porto Alegre: SAGAH, 2018. 188 p.

ELMASRI, R.; NAVATHE, S. B. *Sistemas de bancos de dados*. 6. ed. São Paulo: Pearson Education do Brasil, 2011. 788 p.

MANNINO, M. V. *Projeto, desenvolvimento de aplicações e administração de banco de dados*. 3. ed. Porto Alegre: AMGH; 2008. 717 p.

RAMAKRISHNAN, R..; GEHRKE, J. *Sistemas de gerenciamento de banco de dados*. 3. ed. Porto Alegre: AMGH; Bookman, 2008. 905 p.

Leituras recomendadas

CASANOVA, M. A. *et al.* (org.). *Bancos de dados geográficos*. Curitiba: EspaçoGEO, 2005. 504 p. Disponível em: http://www.dpi.inpe.br/livros/bdados/. Acesso em: 9 maio 2020.

CIFERRI, C. D. A. *Álgebra relacional e SQL*. São Carlos: Instituto de Ciências Matemáticas e de Computação, 2013. 57 p. (Notas de aula). Disponível em: http://wiki.icmc.usp.br/images/2/2c/SCC578920131-algebraSQL.pdf. Acesso em: 9 maio 2020.

DATE, C. J. *SQL e teoria relacional*: como escrever códigos SQL precisos. São Paulo: Novatec, 2015. 536 p.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. *Sistema de banco de dados*. 3. ed. São Paulo: Makron Books, 1999. 778 p.

Fique atento

Os *links* para *sites* da *web* fornecidos neste capítulo foram todos testados, e seu funcionamento foi comprovado no momento da publicação do material. No entanto, a rede é extremamente dinâmica; suas páginas estão constantemente mudando de local e conteúdo. Assim, os editores declaram não ter qualquer responsabilidade sobre qualidade, precisão ou integralidade das informações referidas em tais *links*.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS