

BANCO DE DADOS CONSULTAS AVANÇADAS E VISÕES (VIEWS)

Autor: Me. Otacílio José Pereira

INICIAR

introdução

Introdução

Quando estamos estudando banco de dados, ao iniciar com consultas exploramos os recursos do comando SELECT para os primeiros resultados que em geral já são interessantes.

Esta unidade visa evoluir a construção de consultas com recursos mais avançados associados a este comando de SELECT. Serão vistos tópicos como junções, comandos para ordenação e agrupamento dos dados, uso de funções de agregação como somatórios e contabilizações e por fim como criar visões (views) para tornar mais fácil e prático a manipulação de consultas.

Todos estes recursos são bem úteis em tarefas cotidianas de produzir relatórios e atender demandas de usuários que precisam extrair informação e conhecimento mais apurado a partir do banco de dados.

Vamos em frente!

Junções

Em um primeiro momento no aprendizado com comandos de DML como o `SELECT` o foco é selecionar os dados de uma tabela, por exemplo, quais os clientes do sexo feminino, entretanto é mais comum precisarmos de dados que estão presentes em diversas tabelas. Se a estruturação dos dados divide cada registro em sua respectiva tabela, Clientes, Compras, Cidades, Produtos, ao recuperar estes dados é preciso de alguma forma para juntá-los.

Vamos analisar um exemplo que envolve uma loja virtual, com seus dados de clientes, cidades e de compras. Parte do modelo está exposto a seguir na Figura 4.1. Imagine que haja a necessidade de saber quais clientes (Nome e Contato) com suas respectivas Cidades (Dcr_Cidade) e que realizaram compras no mês de abril (campo Data_Compra em Compras), talvez para fazer um contato e saber se o cliente está satisfeito com a compra. Perceba que Nome e Contato estão na tabela Cliente, a descrição da cidade está em Cidade e Data e Valor estão na tabela de Compras. Fica então a questão, como colocar estes dados juntos?



Figura 4.1 - Parte de modelo de exemplo do cenário de Loja Virtual

Fonte: Elaborado pelo autor (2019).

Um primeiro raciocínio é partirmos da ideia da chave estrangeira, ao estudar sobre a modelagem dos dados elas surgem como forma de representar os relacionamentos entre entidades (no Modelo de Entidade e Relacionamento) e tabelas (no Modelo Relacional), servem como elo entre as duas tabelas em que há um relacionamento 1:N por exemplo. Portanto, as chaves estrangeiras desempenham um papel chave para fazer as junções dos dados.

Primeiro exemplo de junção (Inner Join)

Para compreender melhor este uso das chaves estrangeiras, vamos para um exemplo. Observe a Figura 4.2, do lado esquerdo temos um conjunto de resultados provenientes da tabela Cliente. À direita encontram-se as cidades cadastradas na tabela Cidade. Como transformar, na linha com nome 'Adriana Araújo', o valor de código da cidade 1 na descrição da Cidade 'Salvador'? É necessário um mecanismo que faça com que a chave estrangeira 1 seja consultada (via junção) na tabela de cidade. Aproveite esta lista de clientes para perceber que existem valores que não possuem a cidade associada (valores NULL), no caso os clientes 'Rodrigo Vieira' e 'Fabiana Moreira', eles não devem conseguir participar da junção pois não terão valores correspondentes na outra tabela.

TABELA: CLIENTE				TABELA: CIDADE	
COD_CLIENTE	NOME	CONTATO	COD_CIDADE	COD_CIDADE	DCR_CIDADE
1	Adriana Araújo	71.982213455	3	1	São Paulo
2	Renato Hoqueira	11.933321999	1	2	Rio de Janeiro
3	Viviane Sales	11.987712022	1	3	Salvador
4	Marcelo Campos	71.973514468	3	4	Manaus
5	Rodrigo Gonçalves	21.986121942	2	5	Porto Alegre
6	Jorge Maranhão	11.995439812	1	6	Campo Grande
7	Rodrigo Vieira	71.972318872	NULL		
8	Vanessa Aquino	21.933211346	2		
9	Fabiana Moreira	11.933216758	NULL		
10	Helena Conceição	21.994419257	1		

Figura 4.2 - Exemplo de duas tabelas dissociadas

Fonte: Elaborado pelo autor (2019).

O mecanismo para resolver esta necessidade é a junção entre tabelas, e um primeiro tipo é a junção interna ou "inner join" cuja sentença está escrita no

Quadro 4.1. A ideia é para cada código de cidade em Cliente buscar na tabela de Cidade a linha que contém o mesmo código, isto é, o campo Cod_Cidade serve como elo entre as duas tabelas.



Quadro 4.1 - Exemplo de Inner Join

Fonte: Elaborado pelo autor.

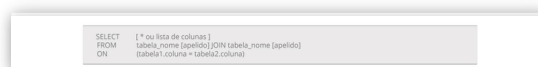
O resultado deste comando de junção pode ser visualizado a seguir na Figura 4.3, compare com as tabelas Cliente e Cidade dissociadas da Figura 4.2, cada registro tem agora a descrição da cidade equivalente ao código que foi usado na junção. O registro de 'Adriana Araújo' com código de cidade 3, foi associado ou juntado com a descrição 'Salvador' vinda da tabela de Cidade. Perceba também que apenas 8 linhas foram recuperadas e existem 10 clientes, o que ocorre é que os clientes sem valores no campo cidade (NULL) não conseguem fazer a junção.

NOME	CONTATO	COD_CIDADE	DCR_CIDADE
Renato Nogueira	11 933321999	1	São Paulo
Viviane Sales	11 987712022	1	São Paulo
Jorge Marinho	11 995439812	1	São Paulo
Maria Conceição	21 954419257	1	São Paulo
Rodrigo Gonçalves	21 986121942	2	Rio de Janeiro
Vanessa Aquino	21 933211346	2	Rio de Janeiro
Adriana Araújo	71 982213455	3	Salvador
Marcela Campos	71 973514998	3	Salvador

Figura 4.3 - Resultado da junção entre as duas tabelas

Fonte: Elaborado pelo autor (2019).

A sintaxe do comando para junção está mostrada a seguir conforme descrito em (PUGA, 2013, p226).



Quadro 4.2 - Sintaxe da junção entre tabelas

Fonte: PUGA (2013, p. 229).

Os termos em '[' indicam que apresentam opções ou podem ser suprimidos. Perceba que na sintaxe não foi explicitado a palavra chave 'INNER', em alguns casos ela é considerada como padrão ao fazer a junção.

Tipos de Junções

Nosso primeiro caso de junção visto anteriormente é denominada de junção interna (inner join) e no caso os campos usados para fazer a junção precisam ter valores nas duas tabelas para que a junção ocorra. Existem outros tipos de junções, por exemplo as junções externas em que não necessariamente devem existir valores correspondente nas duas tabelas.

Para compreender esta questão, vamos reconsiderar nosso exemplo, observe a Figura 4.4 com os dados. Perceba que na esquerda, na tabela Cliente, as linhas com código 7 e 9 não apresentam valores de cidade, por isso não foram consideradas na junção anterior. Por outro lado, na tabela de Cidade à direita, existem as cidades de Manaus, Porto Alegre e Campo Grande (códigos 4, 5 e 6 respectivamente) que não possuem nenhum cliente associado na tabela de clientes, por isso não apareceriam em uma junção como foi feita (e de fato, confira os resultados na Figura 4.3, só aparecem as cidades São Paulo, Rio de Janeiro e Salvador).

TABELA: CLIENTE				TABELA: CIDADE	
COD_CLIENTE	NOME	CONTATO	COD_CIDADE	COD_CIDADE	DCR_CIDADE
1	Adriana Araújo	71 982212455	3	1	São Paulo
2	Rafaela Vasquez	11 933212999	1	2	Rio de Janeiro
3	Viviane Sales	11 987712022	1	3	Salvador
4	Marcelo Campos	71 973514988	3	4	Manaus
5	Rodrigo Gonçalves	21 986112942	2	5	Porto Alegre
6	Jorge Marinho	11 995438612	1	6	Campo Grande
7	Rodrigo Vieira	71 972318872	0000		
8	Vanessa Aquino	21 933211346	2		
9	Pabiana Moreira	11 933216758	0000		
10	Maria Conceição	21 934419257	1		

Figura 4.4 - Exemplo de duas tabelas dissociadas

Fonte: Elaborado pelo autor (2019).

As junções externas (outer) podem ser de dois tipos principais, pela esquerda (left outer join) e pela direita (right outer join). No caso da esquerda, os registros na tabela da esquerda que não possuem correspondentes aparecem mesmo assim no resultado final. No caso da direita, os registros da tabela da direita que não possuem correspondentes são mostrados nos resultados. Usar figuras com a ideia de conjuntos pode ajudar a entender estas junções. Vale destacar que esquerda e direita são definidos pela ordem com que as tabelas aparecem na junção, a primeira que aparece é considerada esquerda e a segunda é direita.

TIPO DE JUNÇÃO	FIGURA ILUSTRATIVA
Interna (inner join) - Apenas os registros que possuem valores correspondentes nas duas tabelas serão considerados.	
Externa - esquerda (left outer join) - São considerados, à esquerda, tanto registros com valores correspondentes na tabela da esquerda como também os que não possuem correspondência.	
Externa - Direita (right outer join) - São considerados, à direita, tanto registros com valores correspondentes na tabela da esquerda como também os que não possuem correspondência.	
Full join - Consideram todos os registros de ambas as tabelas.	

Quadro 4.3 - Tipos de junções

Fonte: Elaborado pelo autor (2019).

saiba mais
Saiba mais

Para aprofundar sobre o uso de junções, vale mencionar duas dicas de leitura, um link para um artigo sobre os diversos tipos de junção, simples, direto e prático. A outra dica é do livro de Puga (2013), interessante também..

Fonte: PUGA, S. **Banco de Dados:** implementação em SQL, PL/SQL e Oracle 11g. São Paulo: Pearson Education do Brasil, 2013.

ACESSAR

Bom, já que entendemos o funcionamento dos 'outer joins' vamos aos exemplos. Antes de entender o exemplo, relembre os dados de clientes e cidades, quais clientes não possuem cidade associada e quais cidades não possuem clientes associados.

Para a junção pela esquerda (left join) o comando e os resultados seriam os mostrados no Quadro 4.4 a seguir, perceba que neste caso as linhas antes suprimidas aparece mas os campos da tabela Cidade são preenchidos com NULL.

EXEMPLO DE COMANDO: JUNCÃO EXTERNA - ESQUERDA (LEFT OUTER JOIN)									
SELECT	C.NOME	C.CONTATO							
	C.COD_CIDADE	at	CL.CLIENTE_COD_CIDADE						
	CID.COD_CIDADE			at	CID.COD_CIDADE				
	C.COD_C								
FROM									
	CL.CLIENTE AS C	LEFT OUTER JOIN		CIDADE AS CID					
ON									
	C.COD_CIDADE = CID.COD_CIDADE								

Comentário:

- Nas linhas sem correspondência, os campos da tabela da direita (CIDADE) ficam sem valor (NULL)

Fonte: Elaborado pelo autor (2019).

Já para a junção pela direita (right outer join) o exemplo está mostrado no Quadro 4.5 a seguir.

EXEMPLO DE COMANDO: JUNCÃO EXTERNA - DIREITA (RIGHT OUTER JOIN)

```

SELECT
  C.NOME, C.CONTATO,
  C.COD_CIDADE as 'CLIENTE_COD_CIDADE',
  CO.COD_CIDADE as 'CIDADE_COD_CIDADE',
  CO.DIX_CIDADE

FROM
  CLIENTES as C RIGHT OUTER JOIN CIDADES as CO

ON
  C.COD_CIDADE = CO.COD_CIDADE
  
```

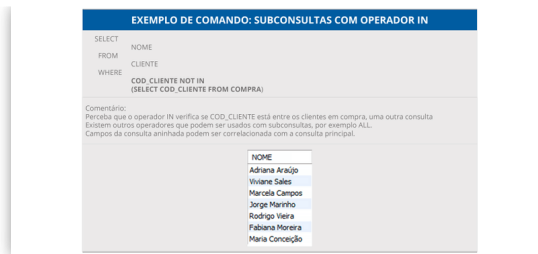
Comentário:
 Nos linhas sem correspondência, os campos da tabela da esquerda (CLIENTES) ficam sem valor (NULL)

Fonte: Elaborado pelo autor (2019).

Neste caso as cidades (tabela da direita) que não possuem correspondentes em clientes aparecem nos resultados e as colunas da tabela da esquerda (Cliente) ficam sem valores (null).

Subconsultas

No comando SELECT, em uma certa consulta é possível usar como parte de suas cláusulas uma outra consulta com um comando completo de SELECT. Elmasri (2011), as chama de consultas aninhadas e também são comumente chamadas de subconsultas. Existem várias formas e com vários outros operadores, o exemplo a seguir no Quadro 4.6 apresenta um destes exemplos.



Quadro 4.6 - Exemplo de subconsultas

Fonte: Elaborado pelo autor (2019).

No exemplo é recuperado os clientes que não possuem nenhuma compra na loja, que pode ser útil para fazer contato e verificar porque os clientes não finalizaram nenhuma compra.

saiba mais
Saiba mais

Existem várias formas de se criar consultas, com alguns outros operadores. Para saber mais sobre subconsultas, o livro de Elmasri, capítulo 5 sobre consultas complexas pode ser uma boa fonte e para uma leitura mais prática, vale olhar também o link sugerido.

Fonte: ELMASRI, R.; NAVATHE, S. **Sistemas de Banco de Dados**. São Paulo: Pearson Addison Wesley, 2011.

ACESSAR

praticar
Vamos Praticar

As junções permitem que os dados em tabelas diferentes possam compor um mesmo conjunto de resultados. Dentre os tipos de junções algumas são: as junções

internas e as junções externas, estas últimas pela esquerda ou pela direita.

Em uma consulta foi feita uma junção (join) entre as tabelas Categoria e Produto onde categoria visa classificar os produtos de uma certa loja e na consulta SELECT, categoria e produto estão escritas nesta ordem (categoria primeiro, antes de produto). O resultado da consulta está expresso na tabela a seguir:

Cod_Categoria	Dcr_Categoria	Cod_Produto	Dcr_Produto
1	Tecnologia	2	Notebook
1	Tecnologia	10	Monitor de Vídeo
2	Móveis	NULL	NULL
3	Eletrodomésticos	21	Liquidificador
3	Eletrodomésticos	23	Televisão

Quadro 4.7 - Resultado de consulta com junção entre Categoria e Produto

Fonte: Elaborado pelo autor (2019).

Qual foi a junção realizada nesta consulta?

- ☐ **a)** Inner Join
- ☐ **b)** Left Outer Join
- ☐ **c)** Right Outer Join
- ☐ **d)** Full Join
- ☐ **e)** Exponential Join

Ordenação

Enquanto um comando bastante versátil, o SELECT permite ainda que os registros sejam organizados para melhor visualização ordenada dos dados ou ainda permite contabilizações e agrupamentos. Para entendermos essas necessidades, no nosso exemplo de loja Virtual, em alguma hora podemos querer ver os Clientes ordenados por nome, para achar mais fácil um cliente na lista pelo seu nome. Mas considere que o objetivo é processar ligações para os clientes e organizados por cidade, neste caso pode ser melhor ordenar por Cidade. Estes dois exemplos estão organizados a seguir (Quadros 4.8 e 4.9).

EXEMPLO DE COMANDO: ORDENAÇÃO (ORDER BY)			
<pre> SELECT C.NOME, C.CONTATO, CID.DCR, CIDADE FROM CLIENTE AS C LEFT OUTER JOIN CIDADE AS CID ON C.COD_CIDADE = CID.COD_CIDADE ORDER BY C.NOME </pre>			
Comentários: - Perceba que a lista está ordenada por nome			
NOME	CONTATO	DCR	CIDADE
Adriana Araújo	71.93232485	Salvador	
Paulina Noronha	11.93232768	SP	
Jorge Marinho	11.99449812	São Paulo	
Henrique Campos	71.97511490	Salvador	
Henri Conceição	21.95449327	São Paulo	
Renato Figueira	11.93232990	São Paulo	
Rodrigo Gonçalves	11.98121892	Rio de Janeiro	
Rodrigo Viera	71.97231892	SP	
Viviane Aquino	11.93231196	Rio de Janeiro	
Viviane Sales	11.98772022	São Paulo	

Quadro 4.8 - Ordenação por ordem crescente de nome do cliente

Fonte: Elaborado pelo autor (2019).

Nome	Sexo	DATA_NASC
Viviane Sales	F	02/11/1995
Marcela Campos	F	19/01/1980
Adriana Araújo	F	03/02/1987
Rodrigo Gonçalves	M	10/05/1992
Renato Nogueira	M	09/07/1977
Jorge Marinho	M	07/06/1990

Quadro 4.10 - Dados de clientes.

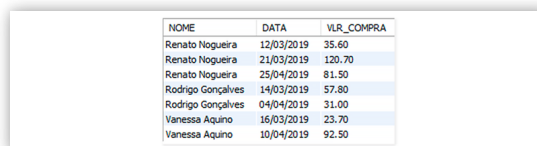
Fonte: Elaborado pelo autor (2019).

Qual a cláusula de ordenação deve ter sido escrita na consulta?

- ☐ **a)** ORDER BY SEXO DESC, NOME ASC
- ☐ **b)** ORDER BY SEXO ASC, NOME ASC
- ☐ **c)** ORDER BY SEXO DESC, DATA_NASC ASC
- ☐ **d)** ORDER BY SEXO ASC, DATA_NASC ASC
- ☐ **e)** ORDER BY SEXO ASC, NOME DESC

Funções (de Agregação) e Agrupamento de Linhas

Uma vez com os dados armazenados em um banco de dados, existe um grande potencial de transformar estes dados em informação e conhecimento. Um primeiro passo pode vir por contabilizações, por exemplo, qual o total de vendas de certo mês, quantos produtos temos em um determinado estoque por produto, além de outros. Para compreender como faremos estas consultas vamos agora trabalhar com um conjunto de dados diferente, os registros são de clientes e suas compras e estão mostrados a seguir (Figura 4.6).



NOME	DATA	VLR_COMPRA
Renato Nogueira	12/03/2019	35.60
Renato Nogueira	21/03/2019	120.70
Renato Nogueira	25/04/2019	81.50
Rodrigo Gonçalves	14/03/2019	57.80
Rodrigo Gonçalves	04/04/2019	31.00
Vanessa Aquino	16/03/2019	23.70
Vanessa Aquino	10/04/2019	92.50

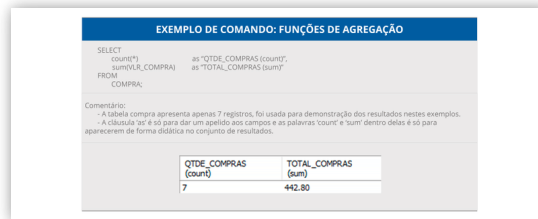
Figura 4.5 - Exemplo de dados com Clientes e Compras

Fonte: Elaborado pelo autor (2019).

Analisando os dados, imagine como calcular o total de compras? E como avaliar a quantidade de compras por cliente? Questões assim podem ser expressas pelas funções de agregação que veremos adiante.

Funções de Agregação

As funções de agregação são funções que permitem calcular o total (sum), contar (count), avaliar o maior (max) e assim por diante. Imagine que se deseja obter o valor total de compras de nosso exemplo e quantas compras foram realizadas. O exemplo a seguir (Quadro 4.10) mostra como elas podem ser utilizadas.



Quadro 4.11 - Exemplo com função de agregação

Fonte: Elaborado pelo Autor (2019).

O Quadro 4.11 mostra o conjunto de funções de agregação que podem ser utilizadas nas consultas.

Função

Descrição

count Conta quantos registros no resultado da consulta

sum Calcula o total dos valores no campo

max Avalia qual o valor máximo no campo

min Avalia qual o valor mínimo no campo

avg Calcula a média dos valores no campo

Quadro 4.12 - Conjunto de Funções de Agregação

Fonte: (PUGA,2013)

As funções usadas até agora foram aplicadas a todo o conjunto de resultado do SELECT mas o mais comum é elas servirem para cálculos em agrupamentos, por exemplo, em vez de calcular o total de todas as compras, ao agrupar o cálculo seria do total de todas as compras por cliente.

Agrupamentos

As contabilizações e totalizações são comuns de ocorrerem junto com agrupamentos. Um agrupamento ocorre quando um conjunto de linhas são combinadas com base em um mesmo grupo de valores. Por exemplo, ao se calcular o total de vendas por Cidade, os registros são agrupados conforme o valor de Cidade. As cidades são base para agrupar os registros que possuem o mesmo valor de cidade.

O exemplo exposto no Quadro 4.12 permite entender como a cláusula “group by” responsável pelos agrupamentos é empregada. No exemplo é possível agrupar as compras por Nome do cliente e assim saber o total de compras, quantas compras cada um fez e qual o valor da maior compra. Perceba que o campo que aparece na cláusula group by deve aparecer na lista de campos de SELECT e ainda, que os campos que não constam na cláusula group by precisam estar com uma função de agregação para se ter um valor resumo. Isso é necessário justamente para saber qual operação (total, contagem, maior) será feita ao agrupar o conjunto de valores de um mesmo grupo.

EXEMPLO DE COMANDO: AGRUPAMENTO (GROUP BY)				
<pre> SELECT C.NOME, C.COUNT(*) AS QTDE, SUM(COMP.VLR_COMPRA) AS TOTAL, MAX(COMP.VLR_COMPRA) AS MAIOR FROM CLIENTE AS C INNER JOIN COMPRA AS COMP ON C.COD_CLIENTE = COMP.COD_CLIENTE GROUP BY C.NOME; </pre>				
Comentários: - O campo da cláusula “group by” aparece na lista de colunas no SELECT e os outros campos são processados via uma função de agregação.				
NOME	QTDE	TOTAL	MAIOR	
Renato Hogueira	3	237,80	120,70	
Rodrigo Gonçalves	2	88,80	57,80	
Vanessa Aquino	2	116,20	92,50	

Quadro 4.13 - Ordenação com várias colunas e uso de ASC e DESC

Fonte: Elaborado pelo autor (2019).

Bom, para fixar a ideia, fica o convite para olhar os dados originais (Figura 4.6) e conferir se os valores conferem. Perceba que lá, “Rodrigo” realizou duas compras nos valores de R\$57,80 e R\$31,00 o que confere com os resultados de 2 compras, num total de R\$88,80 e com o maior valor de R\$57,80.

Filtros em Agrupamentos (having)

Quando se aprende o SELECT, na cláusula WHERE é possível filtrar apenas algumas tuplas que satisfazem certa condição. O comando WHERE continua podendo filtrar tuplas para o agrupamento porém para filtrar os campos com as funções agregadas é necessário usar o comando “having”. A partir disso, as condições são aplicadas sobre as totalizações e contabilizações. Veja o exemplo no Quadro 4.13.

EXEMPLO DE COMANDO: AGRUPAMENTO (GROUP BY) COM HAVING

```

SELECT
  C.NOME,
  COUNT(*) AS QTDE,
  SUM(COMP.VLR_COMPRA) AS TOTAL,
  MAX(COMP.VLR_COMPRA) AS MAIOR
FROM
  CLIENTE AS C
  INNER JOIN COMPRA AS COMP
ON
  C.COD_CLIENTE = COMP.COD_CLIENTE
GROUP BY
  C.NOME
HAVING SUM(COMP.VLR_COMPRA) > 100

```

Comentário:
- Observe que apenas os registros com total maior que 100 foram recuperados.

NOME	QTDE	TOTAL	MAIOR
Renato Nogueira	3	237.80	120.70
Vanessa Aquino	2	116.20	92.50

Quadro 4.14 - Exemplo de filtros em agrupamentos com o “HAVING”

Fonte: Elaborado pelo autor (2019).

Na consulta anterior foi escrita uma condição apenas para o somatório de compras, mas filtros com estes campos calculados combinados com os operadores lógicos de AND e OR por exemplo podem ser expressos, de forma análoga a forma como os filtros da cláusula WHERE são feitos.

praticar

Vamos Praticar

Em uma consulta SELECT, ao se utilizar uma cláusula “group by” é possível fazer o agrupamento dos registros com base em alguns campos e ela é usada em conjunto com funções de agregação, por exemplo um somatório ou média.

Em um sistema de escolas, existe uma tabela TURMA_ALUNO que contém o registro de turmas juntamente com os seus alunos e para cada aluno é registrada a nota. A

especificação está exposta a seguir.

TURMA_ALUNO (COD_TURMA, COD_ALUNO, NOTA)

Para recuperar a lista das turmas e para cada turma a média das notas dos alunos, a consulta é

- ☐ **a)** SELECT COD_TURMA, AVG(NOTA) FROM TURMA_ALUNO GROUP BY COD_TURMA
- ☐ **b)** SELECT COD_ALUNO, AVG(NOTA) FROM TURMA_ALUNO GROUP BY COD_ALUNO
- ☐ **c)** SELECT COD_TURMA, SUM(NOTA) FROM TURMA_ALUNO GROUP BY COD_TURMA
- ☐ **d)** SELECT COD_ALUNO, SUM(NOTA) FROM TURMA_ALUNO GROUP BY COD_TURMA
- ☐ **e)** SELECT COD_TURMA, COUNT(COD_ALUNO) FROM TURMA_ALUNO GROUP BY COD_TURMA

Visões (Views)

Um banco de dados é formado por diversos outros componentes além das tabelas, que indiscutivelmente é a “protagonista” já que abriga nosso principal insumo, o dado. Entretanto existem ainda as visões, os procedimentos armazenados, os gatilhos e outros que em um estudo mais extenso, além do escopo desta unidade, podem ser explorados.

Neste tópico vamos explorar o conceito de visões que tem forte relação com o que vimos até agora sobre como sofisticar nossas consultas. Conforme Elmasri (2011)

“Uma view em terminologia SQL é uma única tabela que é derivada de outras tabelas. Essas outras tabelas podem ser tabelas da base ou views previamente definidas. Uma view não necessariamente existe em forma física, ela é considerada uma tabela virtual, ao contrário das tabelas de base, cujas tuplas sempre estão armazenadas fisicamente no banco de dados. Isso limita possíveis operações de atualização que podem ser aplicadas às views, mas não oferece quaisquer limitações sobre a consulta de uma view.” (ELMASRI, 2011, p.88).

Portanto uma view não apresenta dados em si, como mencionado por Elmasri, ela é uma “tabela virtual”, ela apenas se utiliza das outras tabelas e outras views. Isso traz algumas vantagens em seu uso mas primeiro vamos entender como este mecanismo funciona. Imagine o nosso exemplo de modelo que possui Cidades, Clientes e Compras. Para realizarmos uma consulta que tenha a descrição da cidade, dados do cliente e dados de compra, precisaríamos fazer uma consulta mais complexa que envolve três joins e tudo o mais. Imagine também que esta consulta é bastante utilizada, várias vezes são requisitados relatórios com base nestes dados. Para facilitar, neste cenário pode ser criada uma view conforme o Quadro 4.14 a seguir.

EXEMPLO DE COMANDO: CRIAÇÃO E USO DE VISÕES (VIEWS)							
CREATE VIEW `COMPRAS_CLIENTES_VW` AS							
SELECT							
CL1.NOME, CL1.BAIRRO, CL1.DATA_NASC, CL1.SEXO,							
CID.COD_CIDADE, CID.DCR_CIDADE,							
CMP.DATA_COMPRA, CMP.VLR_COMPRA							
FROM							
CLIENTE CL1 INNER JOIN COMPRA CMP							
ON CL1.COD_CLIENTE = CMP.COD_CLIENTE							
INNER JOIN CIDADE AS CID							
ON CL1.COD_CIDADE = CID.COD_CIDADE							
Comentários:							
- O cabeçalho é um comando de DDL para a criação da view							
- Perceba que a consulta faz o join de três tabelas (Cliente, Compra e Cidade)							
Para recuperar os resultados, basta consultar a view como se fosse uma tabela							
SELECT * FROM COMPRAS_CLIENTES_VW							
NOME	BAIRRO	DATA_NASC	SEXO	COD_CIDADE	DCR_CIDADE	DATA_COMPRA	VLR_COMPRA
Renato Inagata	Marandu	1977-07-09 00:00:00	M	1	São Paulo	2019-03-12 00:00:00	25.60
Rodrigo Gonçalves	Centro	1982-05-10 00:00:00	M	2	Rio de Janeiro	2019-03-04 00:00:00	17.80
Vanessa Aquino	Centro	1972-03-15 00:00:00	F	2	Rio de Janeiro	2019-03-08 00:00:00	23.70
Renato Inagata	Marandu	1977-07-09 00:00:00	M	1	São Paulo	2019-03-21 00:00:00	130.70
Rodrigo Gonçalves	Centro	1982-05-10 00:00:00	M	2	Rio de Janeiro	2019-04-04 00:00:00	31.00
Vanessa Aquino	Centro	1972-03-15 00:00:00	F	2	Rio de Janeiro	2019-04-10 00:00:00	182.30
Renato Inagata	Marandu	1977-07-09 00:00:00	M	1	São Paulo	2019-04-20 00:00:00	81.30

Quadro 4.15 - Criação e exemplo de uso de uma view

Fonte: Elaborado pelo autor (2019).

Existem variações para o comando anterior, por exemplo, no cabeçalho podem ser adicionados os nomes dos campos que inclusive pode ter nomes diferentes, mais intuitivos, do que os usados nas tabelas de base.

Exemplo de comando: Definição dos nomes no cabeçalho da visão (view)

```
CREATE VIEW `COMPRAS_CLIENTES_VW2`  
  
(NOME,          BAIRRO,          DATA_NASC,      SEXO,  
  
COD_CIDADE,     DCR_CIDADE,     DATA_COMPRA,    VLR_COMPRA) AS ...
```

Quadro 4.16 - Uso de nomes dos campos no cabeçalho de criação da View

Fonte: Elaborado pelo autor (2019).

Vale destacar que uma vez criada, a visão (view) funciona como uma tabela e daí pode ser usada com todos os comandos vistos até aqui, por exemplo, ela pode participar de visões, sofrer funções de agregação, group_by, order by e outros comandos usados para consultas via Select. Em alguns bancos são permitidas algumas operações de manipulação (INSERT, UPDATE e DELETE) sobre views porém com limitações.

Para compreender as vantagens e o uso de views, observe a consulta usada na criação da view (com os três joins) e a consulta usada depois já com a view criada, no caso o "select * from compras_clientes_vw". Qual delas é mais simples de usar? É fácil perceber que a segunda opção é mais fácil, e esta é uma das grandes vantagens no uso de visões. Elas reduzem a complexidade de compreender e manipular os dados "mais internos" das tabelas de base. Não é necessário entender todos os relacionamentos, teor de todos os campos, como realizar os diversos joins e criar uma consulta grande e complexa. Tudo fica sintetizado pela view.

Isso pode ser útil em alguns cenários. Por exemplo, para os próprios desenvolvedores ao criar relatórios e precisar de consultar o banco de dados, usar uma view é mais prático. Um usuário final mais ambientado com consultas a dados via ferramentas de relatórios ou pelo próprio SQL pode usar as views para suas próprias consultas. E ainda, quando é necessário integrar dois sistemas, uma view pode ser criada para que o fornecedor de outro sistema tenha acesso aos dados com maior facilidade sem precisar entender o modelo interno do sistema. Além de diminuir a complexidade para consultar dados, as views são úteis ainda no gerenciamento de segurança de acesso aos dados e outras, mas as principais vantagens dessa redução na complexidade no trato com dados.

reflita
Reflita

Nesta unidade tivemos contato com recursos que nos permitem realizar consultas mais complexas sobre os dados, para extrair o dado e compor informações mais refinadas

para que estas ofereçam melhor suporte à tomada de decisões nos negócios.

Isso tem sido cada vez mais usado, por exemplo, reflita sobre um caso. Em seu dia a dia por exemplo, ao pesquisar por um livro em uma livraria, como o site consegue sugerir outros livros relacionados com sua pesquisa?

O mais curioso é que este estudo de consultas mais avançadas em SQL é apenas um ponto de partida para esta compreensão de fenômenos que pode ser extraída a partir dos dados. Áreas inteiras foram desenvolvidas a partir neste contexto, por exemplo, a área de Business Intelligence, KDD (Knowledge Discovery On Data), Data Mining, Data Science e Aprendizagem de Máquina.

Daí vale a reflexão, qual a essência base nestas áreas? Parece ser obter conhecimento sobre dados. E como estas áreas estão hoje e para onde elas podem crescer! É uma boa visão para onde evoluir o conhecimento desta unidade.

praticar

Vamos Praticar

Um banco de dados é formado por diversos componentes além de tabelas. Por exemplo, existem os procedimentos armazenados (stored procedures), as visões (views) e gatilhos (triggers) por exemplo. Cada um destes componentes quando usado agregam alguma vantagem no projeto de banco de dados. Sobre as visões, qual das sentenças a seguir representa uma das vantagens de usar estes componentes em um banco de dados?

- ☐ **a)** Permite programar instruções para lógicas de regras de negócios.
- ☐ **b)** Permite disparar instruções em operações de inserção por exemplo.
- ☐ **c)** Armazena os dados mais simples de uma determinada entidade.

- ☐ **d)** Simplificar a complexidade dos dados em consultas e relatórios.
 - ☐ **e)** Estabelece conjuntos de valores válidos em campos de tabela.
-

indicações

Material Complementar



LIVRO

Sistemas de Banco de Dados

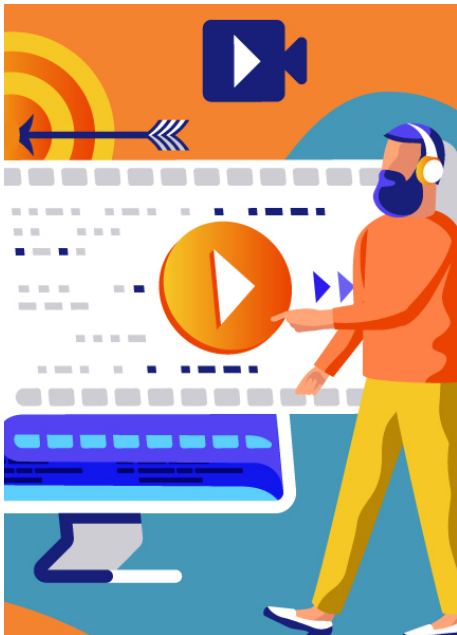
Abraham Silberschatz, S. Sundarshan, Henry Korth

Editora: Elsevier

ISBN: 9788535245356

Comentário: Em banco de dados existem alguns livros que vale a pena o estudante ter como referência. Durante os textos o (Elmasri, 2011) foi muito usado por ser comum a sua disponibilidade na plataforma digital de livros. Mas existem outros dois autores que merecem referência, o que está citado nesta dica, o (Silberchatz, 2012) e vale ainda citar o (Date, 2004). Estes são livros bastantes densos e que compõem um conhecimento bastante sólido e abrangente para a área

de banco de dados. São excelentes para estabelecer uma fundamentação sobre o assunto. E a partir deles, caso o estudante queira uma visão mais prática o (PUGA, 2013) também muito usado em nossas referências e um manual do servidor de banco de dados, e claro com consultas na internet e o estudante terá boas fontes para os mais diversos fins, desde a fundamentação até a prática do dia a dia.



FILME

O homem que mudou o jogo / Moneyball

Ano: 2011

Comentário: Nesta unidade o foco foi em recursos mais avançados para extrair informações a partir dos dados básicos armazenados nas tabelas. Daí foram usadas dentre outros recursos, o uso de funções como somas (sum), médias (avg), máximos e mínimos (max e min). Isso permite por exemplo a análise de dados e estatísticas no banco de dados.

Bom, esta perspectiva é uma das facetas deste filme indicado, nele um treinador faz uma parceria com um estatístico e eles analisam dados de jogadores de beisebol para avaliar como montar times com boa eficiência e ao mesmo tempo com menor recurso financeiro. Estatísticas sobre índice de acerto de jogadas, de erros e outras são analisados pelos personagens através de relatórios. Muito provavelmente os recursos de pegar o dado bruto e realizar agrupamentos, filtros, contabilizações e totalizações foram empregados nesta análise.



conclusão

Conclusão

Vimos nesta unidade como sofisticar as consultas feitas com o comando SELECT. Estes recursos junto com os mais básicos de filtros nas cláusulas WHERE e outros oferecem um leque bastante interessante de comandos que atendem às mais variadas demandas por informação a partir dos dados de um negócio.

Em se tratando de uma conclusão, vale comentar que estes comandos podem ser um ponto de partida para o que hoje é chamada a área de Ciência de Dados. Os resultados dos comandos aprendidos aqui já permitem boas primeiras análises e podem também servir de insumos para análises mais sofisticadas, servir de entrada para algoritmos de aprendizagem de máquina e inteligência artificial. Enfim, são os dados operacionais do negócio podendo sofrer análise em diversos níveis de sofisticação para prover informações e conhecimento ao seus gestores.

São áreas bem interessantes caso pretenda evoluir neste assunto, espero que a dica seja útil!

referências

Referências Bibliográficas

DATE, C.J. **Introdução a Banco de Dados**. Rio de Janeiro: Elsevier, 2004.

ELMASRI, R.; NAVATHE, S. **Sistemas de Banco de Dados**. São Paulo: Pearson Addison Wesley, 2011.

PUGA, S. **Banco de Dados** : implementação em SQL, PL/SQL e Oracle 11g. São Paulo: Pearson Education do Brasil, 2013.

SILBERCHATZ, A. SUNDARSHAN, S. KORTH, H. **Sistema de Banco de Dados** . Rio de Janeiro: Elsevier, 2012.

