**7 Mais SQL: Consultas complexas, triggers, views e modificação de esquema**

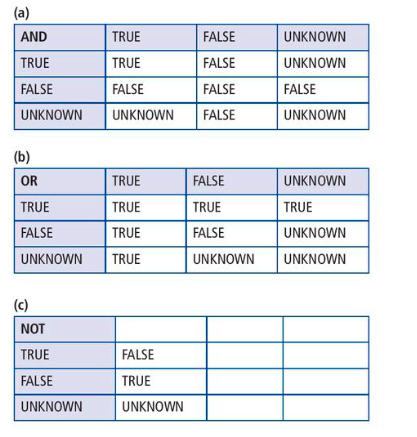
7.1 Consultas de recuperação SQL mais complexas

7.1.1 Comparaçõeos envolvendo NULL e lógica de três valores

A SQL tem diversas regras para lidar com valores NULL.

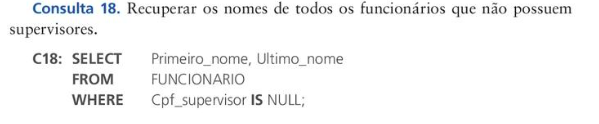
1. **Valor Desconhecido:** A data de nascimento d euma pessoa não é conhecida, e por isso é representada por NULL no BD.
2. **Valor indisponível ou retido:** Uma pessoa tem telefone residencial, mas não deseja que ele seja listado, por isso é representado por NULL
3. **Atributo não aplicável:** Um atributo TituloUniversitario seria NULL para uma pessoa que tivesse nível universitário.

O SQL utiliza ao inves da lógica booleana a lógica de 3 estados: TRUE. FALSE e UNKNOWN. Portanto, deve-se definir os resultados das expressões lógicas de 3 valores quando os concectivos lógicos forem utilizados.

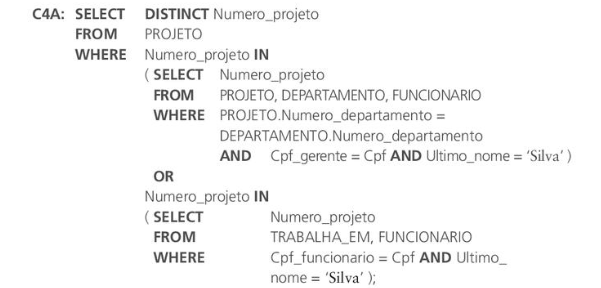


Na operação seleção-projeção-junção a somente são utilizadas as tuplas que na expressão WHERE resultam em TRUE.

A SQL permite identificar se um valor é NULL, para isso não se deve utilizar os operadores tradicionais, mas sim a operação IS NULL ou IS NOT NULL



7.1.2 Consultas aninhadas, tuplas, e comparações de conjunto/mulitconjunto

Alguma consultas necessitam que os valores existentes no banco de dados seja buscadas e depois usados em uma condição de comparação. Essas consultas podem ser formuladas convenientemente com o uso de consultas aninhadas, que são blocos select-from-where completos dentre de outra consulta SQL. Essa consulta é chamada de consulta externa. Essas consultas podem aparecer dentro da cláusula WHERE ou na FROM ou também na SELECT

Esta consulta introduz o operaodr IN, que compara um valor v, com um conjunto de valores V e avalia como TRUE se v for um dos elementos em V.

Nesta consulta, a primeira consulta aninhada seleciona os números dos projetos que possuem um funcionário com sobrenome ‘Silva’ envolvido como gerente, enquanto a segunda consulta aninhada seleciona os números dos projetos que possuem um funcionário com o sobrenome ‘Silva’ envolvido como trabalhador. Na consulta externa usamos o concetivo OR para recuperar uma tupla PROJETO se o NUMERO\_PROJETO dessa tupla estiver no resultado de qualquer das consultas aninhadas.

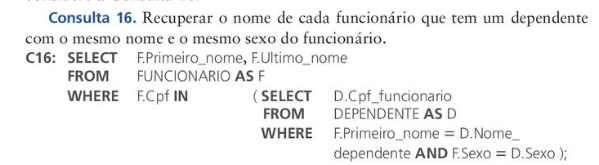
Se esta consulta retornar um único atributo e uma única tupla, o resultado da consulta será um único valor. Nesses casos, é permitido usar = em vez de IN para o operador de comparação.



Esta consulta selecionará os CPFs de todos os funcionários que trabalham na mesma combinação (projeto,horas) em algum projeto no qual o funcionário ‘Joao Silva’ trabalha.

O operador = ANY ou = SOME retorna TRUE se o valor v for igual a alguma valor no conjunto e portanto equivale a IN. A palavra chave ALL também pode ser combinada com esses operadores. Por exemplo, a condição v > ALL V retorna TRUE se o valor v for maior que todos os valores no conjunto.

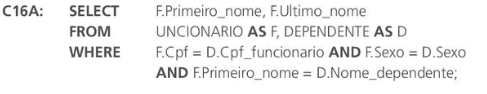
A referência a um atributo não qualificado refere-se a relação declarada na consulta aninhada mais interna. Por exemplo, nas cláusulas SELECT e WHERE da primeira consulta aninhada de C4A, uma referência a qualquer atributo não qualificado da relação PROJETO refere-se a relação PROJETO especificada na cláusula FROM da consulta aninhada. Para se referir a um atributo da relação PROJETO especificada na consulta externa, especificamos a nos referimos a um apelido para essa relação. ESsas regras são semelhantes às regras de escopo para variáveis de programa nas linguagens de programação.



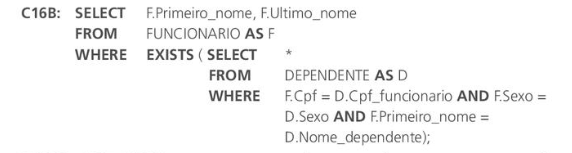
7.1.3 Consultas aninhadas correlacionadas

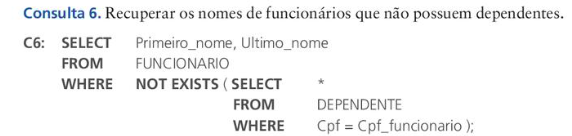
Sempre que uma condição na cláusula WHERE de uma consulta aninhada referencia algum atributo de uma relação declarada na consulta externa, as duas consultas são consideradas correlacionadas. Podemos entender melhor uma consulta correlacionada ao considerar que a consulta aninhada é avaliada uma vez para cada tupla.

Uma consulta escrita com blocos aninhados select-from-where e usando os operadores de comparação = ou IN sempre pode ser expressa com uma única consulta em bloco.

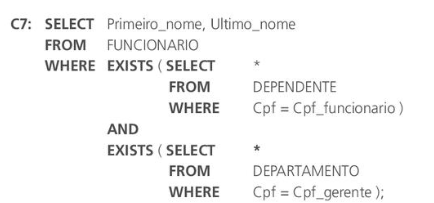


7.1.4 As funções EXISTS e UNIQUE em SQL

EXISTS e UNIQUE são funções booleanas que retornam TRUE ou FALSE, logo elas podem ser usadas em uma condição da cláusula WHERE. A função EXISTS em SQL é usada para verificar se o resultado de uma consulta aninhada correlacionada é vazio ou não. O resultado de EXISTS é um valor booleano TRUE se o resultado da consulta aninhada tiver pelo menos um tupla e FALSE se não houver nenhuma TUPLA.

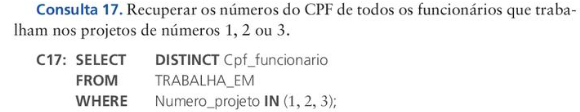
EXISTS e NOT EXISTS costuma ser usadas em conjunto com uma consulta aninhada correlacionada.



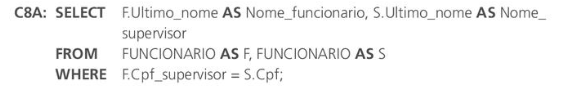


7.1.5 Conjunto explícitos e renomeção de atributos em SQL

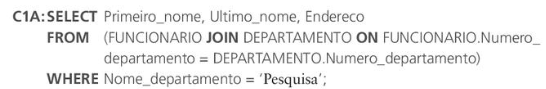
É possível usar um conjunto explícito de valores na cláusula WHERE, em vez de uma consulta aninhada. Esse conjunto é delimitado por parênteses em SQL.



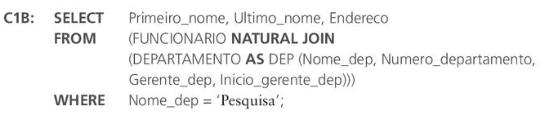
Em SQL é possível renomear qualuqer atributo que apareça no resultado de uma consulta acrescentando o qualificador AS, seguido pelo novo nome desejado.



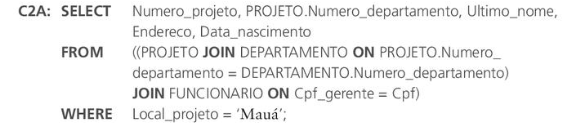
7.1.6 Tabelas de junção em SQL e junções externas

O conceito de uma tabela de junção existe na SQL para permitir aos usuários especificar uma tabela resultantes de uma operação de junção na cláusula FROM de uma consulta. Essa construção pode ser mais fácil de compreender que misturar todas as condições de seleção e junção na cláusula WHERE. 

Pode-se aplicar todos os tipos de junção sobre as relações do SQL. No caso da NATURAL JOIN, caso os atirbutos que representem os mesmo valores não tenham o mesmo nome, deve-se inicialmente renomea-los e depois performar a junção.

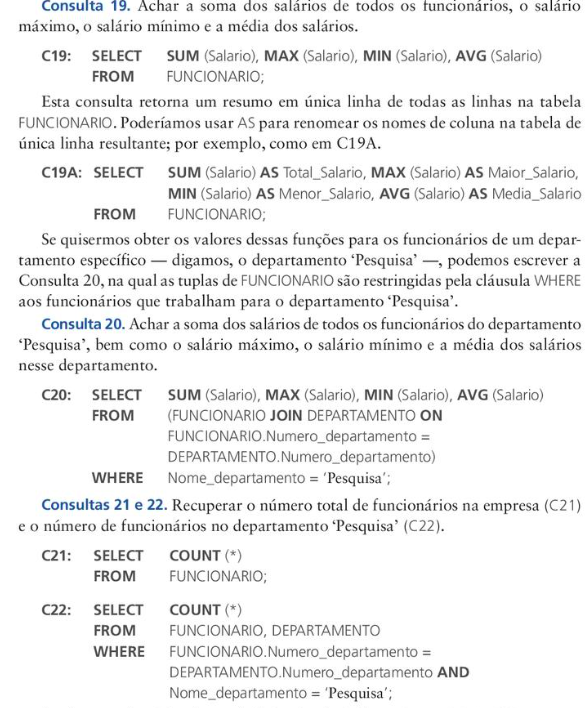


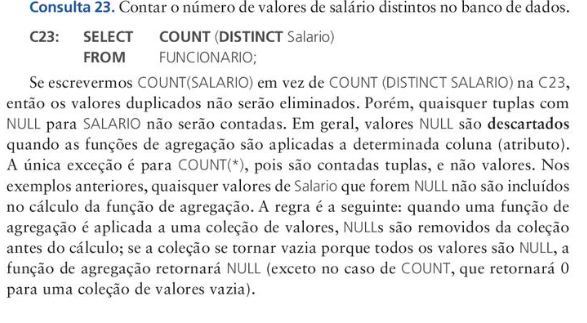
É possível aninhar junções, ou seja, realizar uma junção com mais de duas tableas.



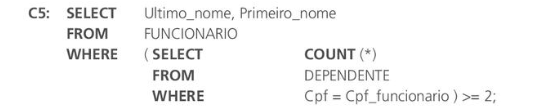
7.1.7 Funções de agregação em SQL

As funções de agregação são usadas para resumir informações de várias tuplas em uma síntese de tupla única. O agrupamento é usado para criar subgrupos de tuplas antes do resumo. Existem diversas funções de agregação embutidas: COUNT, SUM, MAX, MIN e AVG. A função COUNT retorna o número de tuplas ou valores, conforme especificado em uma consulta. As funções SUM,MAX,MIN e AVG podem ser aplicadas a um conjunto e retornam algum valor especifico. Essas funções podem ser usadas na cláusula SELECT ou em uma cláusula HAVING.





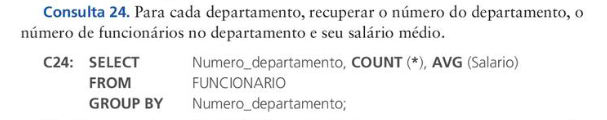
Podemos especificar uma consulta aninhada correlacionada com uma função de agregação, e depois usar a consulta aninhada na cláusula WHERE de uma consulta externa, por exemplo, para recuperar todos os funcionários que tem dois ou mais dependentes.

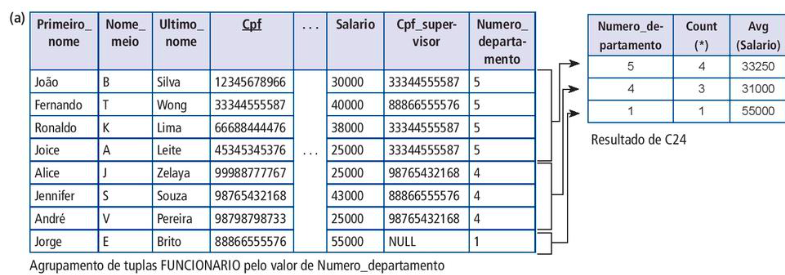


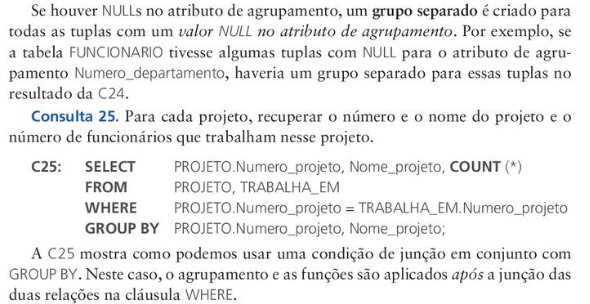
7.1.8 Agrupamento: as cláusulas GRUOP BY e HAVING

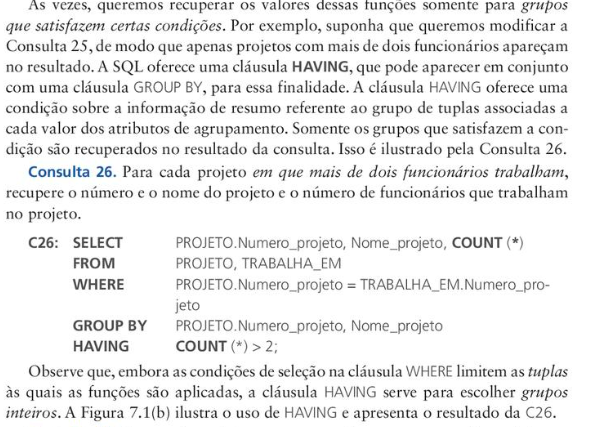
É comum querer aplicar funções de agregação a subgrupos de tuplas em uma relação, na qual os subgrupos são baseados em alguns valores de atributo. Para isso é necessário particionar a relação em subconjuntos de tuplas não sobrepostos. Cada partição consistirá nas tuplas que possuem o mesmo valor de algum atributo chamado atributo de agrupamento.

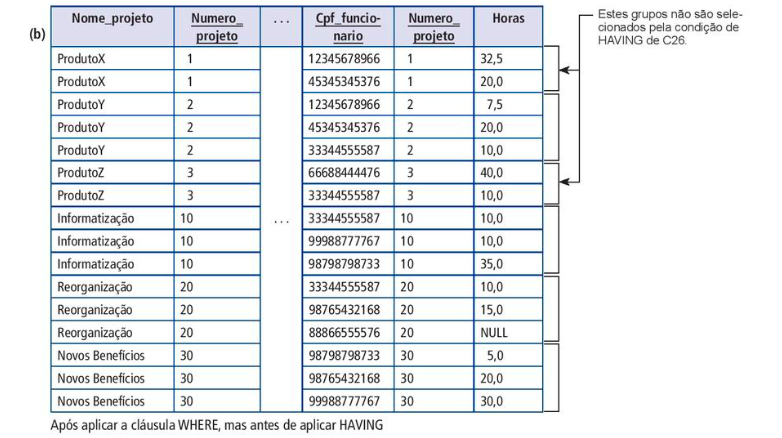
Podemos então aplciar a função a cada grupos desse tipo independentemente, para produzir informações de resumo sobre cada grupo. A SQL tem uma cláusula GROUP BY para essa finalidade. A cláusula GROUP BY especifica os atributos de agrupamento que também devem aparecer na cláusula SELECT, de modo que o valor resultante da aplicação de cada função de agrgação a um grupo de tuplas apareça com o valor dos atributos de agrupamento.

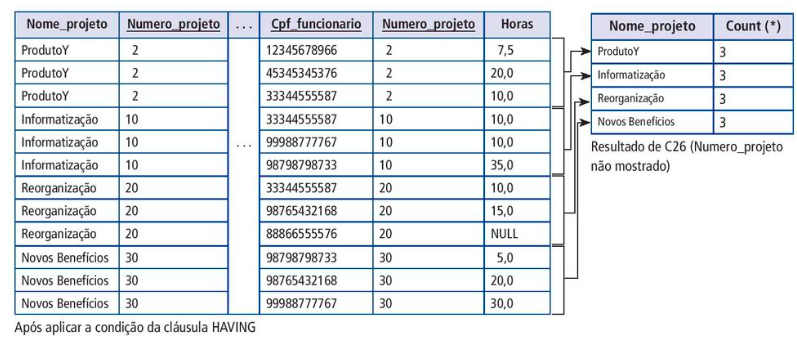


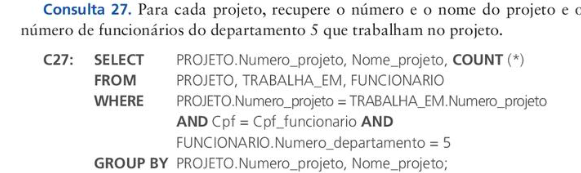




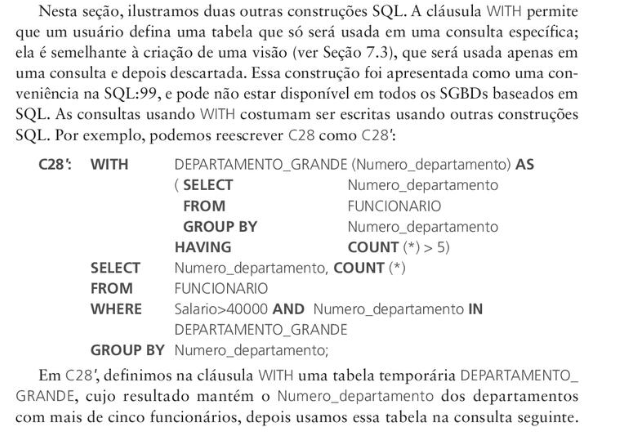


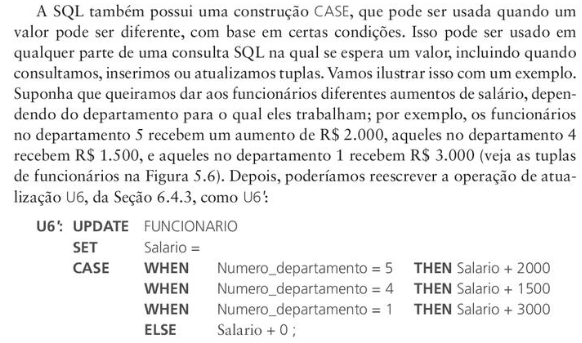






7.1.9 Outras construções SQL; WITH e CASE





7.1.10 Consultas recursivas em SQL

