8.4 CONSULTAS SQL BÁSICAS

8.4.1 A Estrutura SELECT-FROM-WHERE das Consultas Básicas

Consulta 0
Recupere o aniversário e o endereço do(s) empregado(s) cujo nome seja 'John B. Smith'.

QO: **SELECT** DATANASC, ENDEREÇO **FROM** EMPREGADO
WHERE PNOME='John' **AND** MINICIAL='B'AND UNOME='Smith';

Consulta 1
Recupere o nome **e** o endereço de todos os empregados que trabalham no departamento 'Pesquisa'.

Ql: **SELECT** PNOME, UNOME, ENDEREÇO **FROM** EMPREGADO, DEPARTAMENTO
WHERE DNOME='Pesquisa' **AND** DNUMERO=DNO;

Consulta 2
Para cada projeto localizado em 'Stafford', relacione o número do projeto, o número do departamento responsável e o último nome do gerente do departamento, seu endereço e sua data de aniversário.

Q2: SELECT PNUMERO, DNUM, UNOME, ENDEREÇO, DATANASC **FROM** PROJETO,
DEPARTAMENTO, EMPREGADO **WHERE** DNUM=DNUMERO **AND**
GERSSN=SSN **AND**
PLOCALIZACAO='Stafford';

8.4.2 Nomes de Atributos Ambíguos, Aliases (Pseudônimos) e Variáveis de Tuplas

Q1A: **SELECT** PNOME, EMPREGADO.NOME, ENDEREÇO **FROM** EMPREGADO, DEPARTAMENTO
WHERE DEPARTAMENTO.NOME='Pesquisa' **AND**

DEPARTAMENTO.DNUMERO=EMPREGADO.DNUMERO;

A ambiguidade também pode ser originada no caso de consultas que se referem duas vezes à mesma relação, como no exemplo seguinte.

Consulta 8
Para cada empregado, recupere o primeiro e o último nome do empregado e o primeiro e o último nome de seu superior imediato.

Q8: **SELECT** E.PNOME, E.UNOME, S.PNOME, S.UNOME **FROM** EMPREGADO **AS** E, EMPREGADO **AS** S **WHERE** E.SUPERSSN=S.SSN;

EMPREGADO **AS** E(PN, MI, UN, SSN, DT, END, SEX, SAL, SSSN, DNO)
> na cláusula FROM, PN se torna pseudônimo de PNOME, MI de MINICIAL, UN de UNOME, e assim por diante.

Q1B: SELECT E.PNOME, E.NOME, E.ENDEREÇO
FROM EMPREGADO E, DEPARTAMENTO D
WHERE D.NOME='Pesquisa' **AND** D.DNUMERO=E.DNUMERO;

8.4.3 Cláusula WHERE Ausente e Uso do Asterisco

Consultas 9 e 10

Selecione todos os SSNs de EMPREGADO (Q9) e todas as combinações dos SSN de EMPREGADO e dos DNOME de DEPARTAMENTO (Q10) do banco de dados.

Q9: **SELECT** SSN **FROM** EMPREGADO;

Q10: **SELECT** SSN, DNOME
FROM EMPREGADO, DEPARTAMENTO;

QIC: **SELECT ***
FROM EMPREGADO **WHERE** DNO=5;

Q1D: **SELECT ***
FROM EMPREGADO, DEPARTAMENTO **WHERE** DNOME='Pesquisa' **AND** DNO=DNUMERO;

Q10A: **SELECT**

FROM EMPREGADO, DEPARTAMENTO;

8.4.4 Tabelas como Conjuntos em SQL

Consulta 11
Recupere o salário de todos os empregados (Ql 1) e todos os diferentes valores dos salários (Q11A).

Qll: **SELECT ALL** SALÁRIO
FROM EMPREGADO;

Ql IA: **SELECT DISTINCT** SALÁRIO
FROM EMPREGADO;

A SQL incorporou diretamente algumas operações de conjuntos da álgebra relacional: as operações de união de conjuntos (UNION), de diferença de conjuntos (EXCEPT) e de interseção de conjuntos (1NTERSECT).

Consulta 4
Faça uma lista com todos os números de projetos nos quais esteja envolvido algum empregado cujo último nome seja 'Smith'; ou como empregado, ou como gerente do departamento que controle o projeto.

Q4: (SELECT DISTINCT PNUMERO
FROM PROJETO, DEPARTAMENTO, EMPREGADO
WHERE DNUM=DNUMERO **AND** GERSSN=SSN **AND** UNOME='Smith')
UNION
(SELECT DISTINCT PNUMERO
FROM PROJETO, TRABALHA_EM, EMPREGADO
WHERE PNUMERO=PNO **AND** ESSN=SSN **AND** UNOME='Smith');

8.4.5 Comparações entre Substrings e Operadores Aritméticos

Consulta 12
Recupere todos os empregados cujos endereços sejam em Houston, Texas.

Q12: **SELECT** PNOME, LNOME **FROM EMPREGADO WHERE** ENDEREÇO **LIKE** '%Houston,TX%';

Para recuperar todos os empregados que nasceram na década de 50, podemos usar a Consulta 12A. Aqui, '5' precisa ser o terceiro caractere da cadeia (de acordo com nosso formato de data), assim, usaremos o valor '__5_____', com cada underscore servindo de substituto para um caractere qualquer.

Consulta 12A
Encontre todos os empregados que nasceram durante a década de 50.

Q12A: **SELECT** PNOME, UNOME **FROM** EMPREGADO
WHERE DATANASC **LIKE** '__5_____';

Se um sinal underscore ou % for necessário como um caractere literal em uma cadeia, ele deve ser precedido por um caractere de escape, o qual é especificado depois do scring usando-se a palavra-chave ESCAPE. Por exemplo, 'AB_CD\%EF' ESCAPE 'V representa a cadeia literal 'AB_CD%EF', porque \ é especificado como um caractere de escape.

Mostre o resultado dos salários caso fosse dado a todos os empregados que trabalham no 'ProductX' 10% de aumento.

Q13: **SELECT** PNOME, UNOME, 1.1*SALÁRIO **AS** AUMENTO_SAL
FROM EMPREGADO, TRABALHA_EM, PROJETO
WHERE SSN=ESSN **AND** PNO=PNUMERO **AND** PNOME='ProdutoX';

Consulta 14
Recupere todos os empregados do departamento 5 que ganham entre 30 mil e 40 mil dólares.

Q14: **SELECT ***
FROM EMPREGADO
WHERE (SALÁRIO **BETWEEN** 30000 **AND** 40000) **AND** DNO = 5;

A condição (SALÁRIO **BETWEEN** 30000 **AND** 40000) em Q14 é equivalente à condição ((SALÁRIO >= 30000) **AND** (SALÁRIO <= 40000)).

8.4.6 Ordenando o Resultado das Consultas

A SQL permite que o usuário ordene as tuplas do resultado de uma consulta, pelos valores de um ou mais atributos, usando-se a cláusula ORDER BY. Isso é ilustrado na Consulta 15.

Consulta 15
Recupere a lista dos empregados e os respectivos projetos nos quais eles trabalham, ordenada por departamento e, dentro de cada departamento, também por ordem alfabética do último nome, e depois, pelo primeiro nome do empregado.

Q15: **SELECT** DNOME, UNOME, PNOME, PJNOME
FROM DEPARTAMENTO, EMPREGADO, TRABALHAREM, PROJETO
WHERE
DNUMERO=DNO **AND** SSN=ESSN **AND** PNO=PNUMERO **ORDER BY**
DNOME, UNOME, PNOME;

A ordenação default é ascendente. Podemos especificar a palavra-chave DESC se quisermos que os valores do resultado apareçam na ordem descendente. A palavra-chave ASC pode ser usada para explicitar a ordenação ascendente. Por exemplo, se quisermos DNOME em ordem descendente e UNOME e PNOME em ordem ascendente, a cláusula ORDER BY de Q15 pode ser escrita como:
ORDER BY DNOME **DESC,** UNOME **ASC,** PNOME ASC

8.5 CONSULTAS SQL MAIS COMPLEXAS

8.5.1 Comparações Envolvendo NULL e os Três Valores Lógicos

Consulta 18
Recupere os nomes de todos os empregados que não têm supervisor.

Q18: **SELECT** PNOME, UNOME **FROM** EMPREGADO
WHERE SUPERSSN **IS** NULL;

8.5.2 Consultas Aninhadas, Comparações de Tuplas e de Conjuntos/Multiconjuntos

Q4A: **SELECT DISTINCT** PNUMERO **FROM** PROJETO
WHERE PNUMERO **IN**
(SELECT PNUMERO

FROM PROJETO, DEPARTAMENTO, EMPREGADO
WHERE DNUM=DNUMERO AND GERSSN=SSN **AND** UNOME='Smith')
OR PNUMERO **IN**
(SELECT PNO
FROM TRABALHA_EM, EMPREGADO **WHERE** ESSN=SSN **AND**
UNOME='Smith');

A primeira consulta aninhada seleciona os números de projetos que tiverem um 'Smith' como supervisor, enquanto a segunda seleciona os números de projetos que tiverem um empregado 'Smith' envolvido. Na consulta externa, usamos o conectivo lógico OR (ou) para recuperar uma tupla de PROJETO cujo valor PNUMERO esteja contido no resultado de uma das consultas aninhadas.
Se a consulta aninhada devolve um atributo único e uma tupla única, o resultado da consulta será um valor único (escalar). Nesses casos, é possível usar = em vez de IN como operador de comparação. De modo geral, a consulta aninhada vai devolver uma **tabela** (relação), que é um conjunto ou um multiconjunto de tuplas. A SQL permite o uso de **tuplas** de valores em comparações colocando-as entre parênteses. Para ilustração, considere a seguinte consulta:

SELECT DISTINCT ESSN **FROM** TRABALHA_EM
WHERE (PNO, HORAS) **IN (SELECT** PNO, HORAS **FROM** TRABALHA_EM
WHERE SSN="123456789');

Essa consulta vai selecionar o número do seguro social de todos os empregados que trabalham com a mesma combinação (projeto, horas) em algum dos projetos em que o empregado'John Smith' (SSN= '123456789') trabalhe. Nesse exemplo, o operador IN compara a subtupla de valores entre parênteses (PNO, HORAS) de cada tupla de TRABALHA_EM com p conjunto de tuplas compatíveis-por-união produzido pela consulta aninhada.

O exemplo a seguir devolve os nomes dos empregados cujos salários são maiores que os salários de todos os empregados do departamento 5:

SELECT UNOME, PNOME **FROM** EMPREGADO
WHERE SALÁRIO > **ALL (SELECT** SALÁRIO **FROM** EMPREGADO
WHERE DNO=5);

Consulta 16

Recupere o nome de cada um dos empregados que tenham um dependente cujo primeiro nome e sexo sejam o mesmo do empregado em questão.

Q16: **SELECT** E.PNOME, E.UNOME **FROM** EMPREGADO **AS** E **WHERE** E.SSN IN
(SELECT ESSN
FROM DEPENDENTE **WHERE** E.PNOME=DEPENDENTE_NOME **AND** E.SEXO=SEXO);

8.5.3 Consultas Aninhadas Correlacionadas

Sempre que uma condição na cláusula WHERE de uma consulta aninhada se referir a algum atributo da relação declarada na consulta externa, as duas consultas são chamadas **correlacionadas.**

Q16A: **SELECT** E.PNOME, E.UNOME
FROM EMPREGADO **AS** E, DEPENDENTE **AS** D **WHERE** E.SSN=D.ESSN **AND** E.SEXO=D.SEXO **AND** E.PNOME=D.DEPENDENTE_NOME;

Consulta 3
Recupere o nome de cada um dos empregados que trabalha em todos os projetos controlados pelo departamento 5.

Q3: **SELECT** PNOME, UNOME **FROM** EMPREGADO
WHERE
(**(SELECT** PNO **FROM** TRABALHA_EM **WHERE** SSN=ESSN)
CONTAINS
(SELECT PNUMERO **FROM** PROJETO **WHERE** DNUM=5));

8.5.4 As Funções EXISTS e UNIQUE da SQL

A função EXISTS da SQL é usada para verificar se o resultado de uma consulta aninhada correlacionada é vazio (não contém nenhuma tupla) ou não. Ilustramos o uso do EXISTS — e do NOT EXISTS — com alguns exemplos. Primeiro, formulamos um modo alternativo para a Consulta 16 usando EXISTS. A Q16B mostra essa opção:

Q16B: **SELECT** E.PNOME, E.UNOME **FROM** EMPREGADO **AS** E **WHERE**
EXISTS (SELECT *
FROM DEPENDENTE
WHERE E.SSN=ESSN **AND** E.SEXO=SEXO
AND E.PNOME=DEPENDENTE_NOME);

EXISTS e NOT EXISTS são empregados, normalmente, em conjunto com as consultas aninhadas correlacionadas.

Consulta 6
Recupere os nomes dos empregados que não possuem nenhum dependente.
Q6: **SELECT** PNOME, UNOME
FROM EMPREGADO **WHERE**
NOT EXISTS (SELECT * FROM DEPENDENTE **WHERE**
SSN=ESSN);

Em Q6, a consulta aninhada correlacionada devolve todas as tuplas de DEPENDENTE relacionadas a uma tupla EMPREGADO em particular. Se não existir nenhuma, a tupla EMPREGADO será selecionada.

Consulta 7
Relacione os nomes dos gerentes que possuam ao menos um dependente.

Q7: **SELECT** PNOME, UNOME
FROM EMPREGADO **WHERE**
EXISTS (SELECT *
FROM DEPENDENTE **WHERE** SSN=ESSN)
AND EXISTS (SELECT *
FROM DEPARTAMENTO **WHERE** SSN=GERSSN);

Q3A:
SELECT PNOME, UNOME **FROM** EMPREGADO **WHERE**
NOT EXISTS ((SELECT PNUMERO **FROM** PROJETO
WHERE DNUM=5)
EXCEPT (SELECT PNO **FROM** TRABALHA_EM
WHERE SSN=ESSN));

Em Q3A, a primeira subconsulta (que não está correlacionada) seleciona todos os projetos controlados pelo departamento 5, e a segunda (que está correlacionada) seleciona todos os projetos nos quais o empregado em particular, que está sendo considerado, trabalhe. Se a diferença dos

conjuntos, o da primeira subconsulta MINUS (EXCEPT) e o da segunda, for vazio, isso significa que o empregado trabalha em todos os projetos e então é selecionado. Lembre-se de que EXCEPT é o operador da diferença entre os conjuntos.

Q3B:

```
SELECT UNOME, PNOME FROM EMPREGADO WHERE
NOT EXISTS
        (SELECT * FROM TRABALHA_EM E
                WHERE (B.PNO IN (SELECT PNUMERO
                                FROM PROJETO WHERE DNUM=5) )
                        AND
                        NOT EXISTS (SELECT * FROM TRABALHA_EM C
                                WHERE C.ESSN=SSN AND C.PNO=B.PNO) );
```

Em Q3B, a consulta aninhada mais externa seleciona qualquer tupla TRABALHA_EM (B) cujos PNO sejam de projetos controlados pelo departamento 5, se não existir nenhuma tupla TRABALHA_EM (C) com o mesmo PNO e o mesmo SSN da tupla EMPREGADO que está sendo considerada na consulta externa. Se não existir nenhuma dessas tuplas, selecionaremos a tupla EMPREGADO.

8.5.5 Conjuntos Explícitos e Nomes Alternativos de Atributos em SQL

Vimos diversas consultas aninhadas na cláusula WHERE. E também possível usar um **conjunto de valores explícitos** na cláusula WHERE em vez de consultas aninhadas. Esse conjunto, em SQL, é colocado entre parênteses.

Consulta 17
Recupere os números dos seguros sociais de todos os empregados que trabalham nos projetos 1, 2 ou 3. DISTINCT ESSN TRABALHA_EM PNO **IN** (1,2, 3);

Q17: **SELECT FROM WHERE**

Em SQL é possível dar nomes alternativos a qualquer atributo pela adição do qualificador AS seguido do novo nome desejado — esses nomes aparecerão no resultado da consulta

Q8A: **SELECT** E.UNOME AS NOME_EMPREGADO, S.UNOME AS

NOME_SUPERVISOR **FROM** EMPREGADO AS E, EMPREGADO **AS** S

WHERE

E.SUPERSSN=S.SSN;

8.5.6 Junção (JoiN) de Tabelas em SQL

Q1A: **SELECT** PNOME, UNOME, ENDEREÇO
FROM (EMPREGADO **JOIN** DEPARTAMENTO **ON** DNO=DNUMERO)
WHERE DNOME='Pesquisa';

A cláusula FROM em Ql A contém uma única tabela juntada (joined table). Os atributos dessa tabela são todos aqueles da primeira tabela, EMPREGADO, seguidos de todos os atributos da segunda tabela, DEPARTAMENTO.

QIB: **SELECT** PNOME, UNOME, ENDEREÇO **FROM** (EMPREGADO **NATURAL JOIN**
(DEPARTAMENTO **AS** DEPT (DNOME, DNO, GERSSN, GERDATA IN))) **WHERE** DNOME='Pesquisa';

O tipo default de junção de uma tabela é a **junção interna** (innerjoin), na qual uma tupla é inclusa no resultado somente se existir uma tupla que combine na outra relação. Por exemplo, na consulta Q8A, somente os empregados que possuírem um supervisor serão inclusos no resultado; uma tupla EMPREGADO, cujo valor de SUPERSSN for NULL, será excluída.

Q2A: **SELECT** PNUMERO, DNUM, UNOME, ENDEREÇO, DATANASC
FROM ((PROJETO **JOIN** DEPARTAMENTO **ON** DNUM=DNUMERO)
JOIN EMPREGADO **ON** GERSSN=SSN) **WHERE**
PLOCALIZACAO='Stafford';

8.5.7 Funções Agregadas em SQL

Há diversas funções pré-construídas para funções agregadas como funções de operação relacional: COUNT, SUM, MAX, MIN e AVG.
A função COUNT devolve o número de tuplas ou valores especificado em uma consulta. As funções SUM, MAX, MIN e AVG são aplicadas em um conjunto ou multi-conjunto de valores e devolvem, respectivamente, a soma, o valor máximo, o valor mínimo e a média desses valores.

Consulta 19
Encontre a soma dos salários, o maior salário, o menor salário e a média salarial de todos os empregados.

Q19: **SELECT SUM** (SALÁRIO), **MAX** (SALÁRIO), **MIN** (SALÁRIO), **AVG** (SALÁRIO) **FROM** EMPREGADO;

Consulta 20
Encontre a soma dos salários de todos os empregados do departamento 'Pesquisa', bem como o maior salário, o menor salário e a média salarial desse departamento.

Q20: **SELECT SUM** (SALÁRIO), **MAX** (SALÁRIO), **MIN** (SALÁRIO), **AVG** (SALÁRIO) **FROM** (EMPREGADO **JOIN** DEPARTAMENTO **ON** DNO=DNUMERO) **WHERE** DNOME='Pesquisa';

Consultas 21 e 22
Recupere o número total de empregados da empresa (Q21) e o número de empregados do departamento 'Pesquisa' (Q22).

Q21: **SELECT COUNT (*)**
FROM EMPREGADO;
Q22: **SELECT COUNT (*)**
FROM EMPREGADO, DEPARTAMENTO **WHERE** DNO=DNUMERO **AND** DNOME='Pesquisa';

Aqui, o asterisco (*) refere-se às linhas (tuplas), logo, COUNT (*) devolverá, para o resultado da consulta, o número de linhas. Também podemos usar a função COUNT para contar o número de valores em uma coluna em vez o número de tuplas, como veremos no próximo exemplo.

Consulta 23
Conte o número dos diferentes valores de salário contidos no banco de dados.

Q23: **SELECT COUNT (DISTINCT** SALÁRIO) **FROM** EMPREGADO;

Se escrevermos COUNT(SALARIO) em vez de COUNT(DISTINCT SALÁRIO) na Q23, então os valores repetidos não serão eliminados. Entretanto, uma tupla com NULL em SALÁRIO não será contabilizada. Em geral, os valores NULL são **descartados** quando se aplicam as funções agregadas em uma coluna (atributo) em particular.

Q5: SELECT UNOME, PNOME **FROM** EMPREGADO **WHERE (SELECT COUNT (*)**
FROM DEPENDENTE **WHERE** SSN=ESSN) >= 2;

A tupla aninhada correlacionada conta o número de dependentes que cada um dos empregados tem; se esse número for maior ou igual a dois, a tupla desse empregado será selecionada.

8.5.8 Agrupamento: as Cláusulas GROUP BY e HAVING

Consulta 24
Para cada departamento, recupere seu número, o número de empregados que nele trabalham e a média de seus
salários.

Q24: SELECT DNO, COUNT (*), **AVG** (SALÁRIO) **FROM** EMPREGADO **GROUP BY DNO;**

Em Q24, as tuplas EMPREGADO serão particionadas em grupos — cada grupo tendo o mesmo valor no atributo de agrupamento DNO. As funções COUNT e AVG serão aplicadas em cada grupo de tuplas.

Consulta 25
Para cada projeto, recupere seu número, seu nome e o número de empregados que nele trabalham.

Q25: **SELECT** PNUMERO, PJNOME, COUNT (*) **FROM** PROJETO, TRABALHA_EM
WHERE PNUMERO=PNO **GROUP BY** PNUMERO, PJNOME;

Q25 mostra como podemos usar uma condição de junção com a GROUP BY. Nesse caso, o agrupamento e as funções serão aplicados depois da junção das duas relações.

Consulta 26
Para cada projeto em que trabalhem mais de dois empregados, recupere o número do projeto, seu nome e o
número de empregados.

Q26: **SELECT** PNUMERO, PJNOME, **COUNT (*)**
FROM PROJETO, TRABALHA_EM

WHERE PNUMERO=PNO
GROUP BY PNUMERO, PJNOME
HAVING COUNT (*) > 2;

Observe que, enquanto as condições de seleção da cláusula WHERE limitam as tuplas nas quais as funções serão aplicadas, a cláusula HAVING serve para escolher grupos inteiros. A Figura 8.6b ilustra o uso da HAVING e mostra o resultado da Q26.

Consulta 27
Para cada projeto, recupere seu número, seu nome e o número de empregados do departamento 5 que nele trabalhem.

Q27: **SELECT** PNUMERO, PNOME, **COUNT (*)**
FROM PROJETO, TRABALHA_EM, EMPREGADO **WHERE** PNUMERO=PNO **AND**
SSN=ESSN **AND** DNO=5 **GROUP BY** PNUMERO, PNOME;

Aqui restringimos as tuplas da relação (e daí as tuplas de cada grupo) àquelas que satisfazem a condição especificada na cláusula WHERE — a saber, aquelas dos que trabalham no departamento 5.

Consulta 28
Para cada departamento que tenha mais de cinco empregados, recupere o número do departamento e o número dos
empregados que recebem mais de 40 mil dólares.

Q28: **SELECT** DNUMERO, **COUNT (*)**
FROM DEPARTAMENTO, EMPREGADO **WHERE** DNUMERO=DNO **AND**
SALARIO>40000 **AND** DNO **IN (SELECT** DNO
FROM EMPREGADO **GROUP BY** DNO **HAVING COUNT (*) > 5)**
GROUP BY DNUMERO;

8.6 COMANDO INSERT (INSERÇÃO), DELETE (EXCLUSÃO) E UPDATE (ATUALIZAÇÃO) EM SQL

8.6.1 O comando INSERT

Exemplo 1: Adicionar uma nova tupla 'a relação EMPREGADO.

U1: **INSERT INTO** EMPREGADO
VALUES ('Richard', 'K', 'Marini', '653298653', '1962-12-30', '98'

Oak Forest,Katy,TX', 'M', 37000, '987654321', 4);

Adicionar uma tupla de um novo EMPREGADO para o qual sabemos apenas os atributos PNOME, UNOME, DNO e SSN.

U1A: **INSERT INTO** EMPREGADO (PNOME, UNOME, DNO, SSN)
VALUES ('Richard', 'Marini', 4, '653298653');

Exemplo 2: integridade referencial

U2: **INSERT INTO** EMPREGADO (PNOME, UNOME, SSN, DNO)
VALUES ('Robert', 'Hatcher', 4, '980760540', 2);

(* U2 seria rejeitado se o SGBD implementasse a integridade referencial, caso não exista nenhuma tupla DEPARTAMENTO no banco de dados com DNUMERO = 2 *).

U2A: **INSERT INTO** EMPREGADO (PNOME, UNOME, DNO)
VALUES ('Robert', 'Hatcher', 5);

(* U2A seria rejeitado se houvesse a verificação para impedir o registro de valores *null* pelo SGBD *).

Uma variação do comando INSERT é a inserção de diversas tuplas em uma relação por meio da conjunção da inserção do resultado de uma consulta em uma nova relação que será criada. Por exemplo, para criar uma tabela temporária que tenha o nome de 'númeor de empregados' e que contenha o total de salários de cada departamento, podemos formular o comando U3A e U3B:

```
U3A: CREATE TABLE   DEPTS_INFO
     (DEPT_NOME     VARCHAR(15).
     NO_DE_EMPS     INTEGER,
     TOTAL_SAL      INTEGER);
U3b: INSERT INTO    DEPTS_INFO (DEPT_NOME, NO_DE_EMPS,
                    TOTAL_SAL)
     SELECT         DNOME, COUNT (*), SUM (SALARIO)
     FROM           (DEPARTAMENTO JOIN EMPREGADO ON
                    DNUMERO+DNO)
     GROUP BY       DNOME;
```

8.6.2 O Comando DELETE

```
U4A:  DELETE FROM
WHERE U4B: DELETE FROM
WHERE U4C: DELETE FROM
WHERE
U4D: DELETE FROM
EMPREGADO UNOME='Brown'; EMPREGADO SSN='123456789';
EMPREGADO DNO IN (SELECT FROM WHERE EMPREGADO;
DNUMERO
DEPARTAMENTO
DNOME='Pesquisa');
```

8.6.3 O Comando UPDATE

```
U5: UPDATE    PROJETO
SET           PLOCALIZACAO = 'Bellaire', DNUM = 5
WHERE     PNUMERO=10;

U6: UPDATE    EMPREGADO
SET           SALÁRIO = SALÁRIO *1.1
WHERE     DNO IN (SELECT     DNUMERO
FROM       DEPARTAMENTO WHERE     DNOME='Pesquisa');
```

Referência Bibliográfica:

ELSMARI, R.; NAVATHE, S.B. Sistemas de Banco de Dados. 4ª. Edição, Editora Person Addison Wesley, São Paulo, 2005.