DB2

I n d i c e

1 Introdução

Devido à carência de literatura destinada ao ensino de DB2 (Banco de Dados e SQL) para estudantes, a FUTURE SCHOOL elaborou a presente apostila, com o intuito de estabelecer um mínimo de conhecimentos destinados a introduzir o estudante no mundo dos Gerenciadores de Banco de Dados e da Linguagem SQL.

Esperamos passar a vocês um pouco de nosso conhecimento e se por ventura houver alguma falha de nossa parte, podem ter a certeza que não foi intencional, e assim sendo aceitaremos qualquer material que venha acrescentar o conteúdo desta apostila, aliás a FUTURE SCHOOL está aberta a toda e qualquer crítica. “Não estamos aqui, somente para ganhar, e sim para tentar passar um pouco de nossa experiência”.

Muito obrigado.

2 Modelo relacional

Suas principais características são;

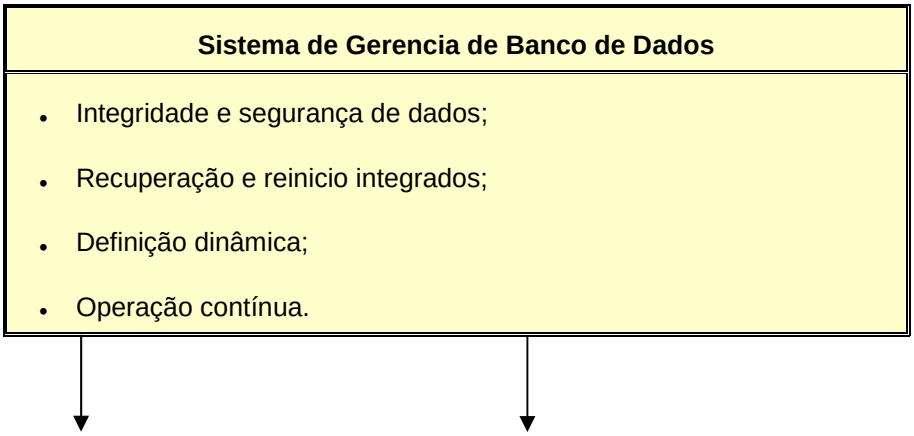
- Independência dos dados;
- Acesso fácil aos dados através de uma linguagem específica; • Acesso interativo;s que conseguem manipular dados em tabelas.
- Acesso via programas de aplicação;
- Permite o processamento com conjunto de dados;
- Permite a utilização de conceitos matemáticos de conjuntos (união, itersecção, diferença etc);
- O resultado de um acesso é uma tabela de dados;
- Fácil manutenção das bases de dados;
- Maior autonomia ao usuário;
- Aumento de produtividade;
- Maior flexibilidade no tratamento dos dados; e;
- Aumenta a segurança e integridade das informações.

2.1 Tabela Relacional

Todos os dados e relacionamentos entre dados são representados por valores de campo Pointers físicos não utilizados.

Cada ocorrência da tabela é chamada de linha. Os atributos são chamados de colunas.

As tabelas DB2 são definidas e manipuladas usando-se a linguagem SQL .



Modelo de dados relacional

- Os dados vistos como tabelas;
- Projeto mais fácil

Linguagem SQL

- Definição, manipulação e controle de dados

Exemplo de Tabela Relacional: Vide item 6.1.1 Tabelas

3 SGBD

Um SGBD (Sistema de Gerenciamento de Banco de Dados) ou DBMS (Data Base Management System) é o software responsável pelo gerenciamento (armazenamento e recuperação) dos dados no Banco de Dados, em outras palavras é uma coleção de programas que permitem ao usuário definir, construir e manipular Bases de Dados para as mais diversas finalidades.

Um DBMS relacional exige:

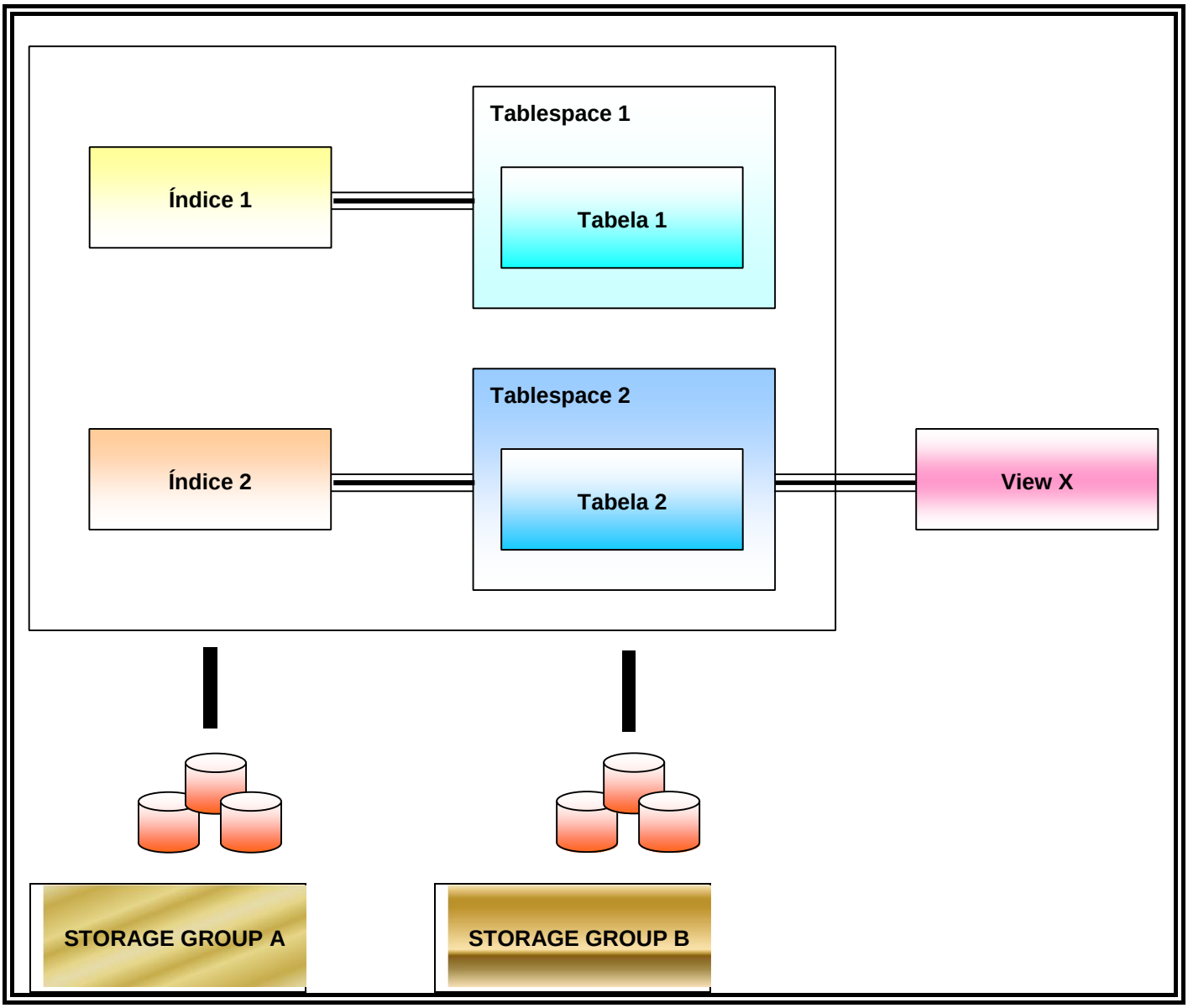
- **Estrutura tabular de dados** – Os dados devem ser colocados em tabelas;
- **Operadores de tabelas (SQL, DDl, DML e DCL)** – Deve ter operadores que consigam manipular dados em uma tabela;
- **Uniformidade e integridade dos dados** – Possui mecanismos que mantém a integridade dos dados que são adicionados ou modificados. O DBMS deve também ter mecanismos capazes de forçar uma uniformização dos dados que nela são armazenados. Por exemplo, podemos forçar o armazenamento dos dados sempre no formato numérico; e;
- **Independência dos dados** – A lógica das aplicações não devem se preocupar com a estrutura de armazenamento e a teoria de acesso aos dados.

Suas principais características são:

- Integridade / Consistência;
- Restrições;
- Segurança / Privacidade;
- Restauração; • Reorganização; e;
- Eficiência.

4 Banco de dados (Data Base – DB2)

Representa o arquivo físico de dados, armazenado em dispositivos periféricos, onde estão armazenados os dados de diversos sistemas, possibilitando um armazenamento e manipulação de dados mais eficientes e não redundantes, índices e tablespaces agrupados sob um critério administrativo.



4.1 Tabelas

É a estrutura fundamental de um banco de dados relacional, na qual são armazenadas linhas (registros) e colunas (campos). Os dados são normalmente sobre uma determinada categoria de assuntos, como por exemplo, clientes,

faturamento, verbas, aluno, colaboradores, peças, etc. Ela é constituída de um número fixo de colunas e um número variável de linhas.

Exemplo de Tabela de Alunos

Colunas (campos)

Linhas (registros)

Código	Nome	Curso	Turma
00010	José da Silva	Cobol	0001
00020	João de Souza	Cics	0002
00030	Maria dos Santos	Cics	0002
00040	Joaquim da Silva	Cics	0003
00050	Manoel dos Santos	Cobol	0002

4.2 Tablespace

Conjunto de datasets VSAM LDS (linear data sets) que contém dados de uma ou mais tabelas. Suas páginas podem ser de 4K ou 32K. Existem 3 (três) tipos de tablespaces:

- **Simples**
  - As páginas de seus data sets podem conter dados de uma ou mais tabelas;
- **Segmentado**
  - É dividido em segmentos que são constituídos de 4 a 64 páginas e sempre múltiplos de 4. Cada segmento só pode conter dados de uma tabela. Sua utilização é recomendada pois administra melhor o espaço;
- **Particionado**
  - Recomendado para tabelas gigantes. Pode ser dividido em até 254 partições;
  - Requer um índice *cluster* que serve de filtro para determinar a distribuição das linhas entre as partições.

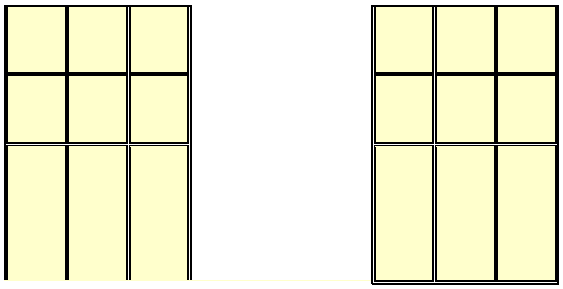
4.3 View (visões)

É uma visão lógica, permitindo a representação de dados de uma ou mais tabelas, como sendo uma outra tabela. Nela podemos demonstrar determinadas linhas e colunas de uma ou mais tabelas. Sua utilização, nada difere com a de uma tabela. Conforme o caso, uma VIEW, pode ser somente para leitura.

• Visão Idêntica;

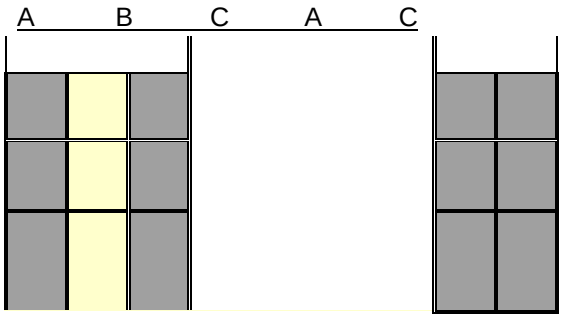
TABELA VISÃO

A B C A B C



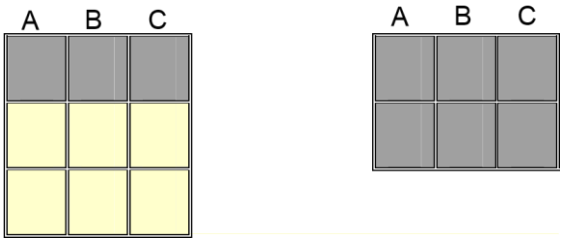
• Seleção por seleção de colunas;

TABELA VISÃO



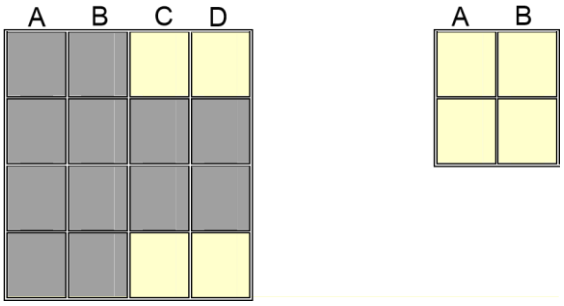
• Seleção por seleção de linhas;

TABELA VISÃO



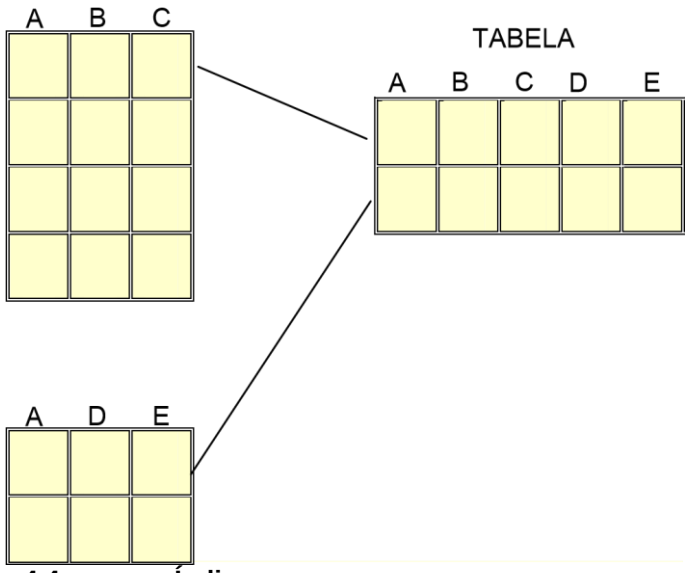
• Seleção por seleção de linhas;

TABELA VISÃO



• Seleção por junção de tabelas;

TABELA



4.4 Índice

É um conjunto ordenado de ponteiros para dados de uma tabela. Pode ser formado por mais de uma coluna. Sua criação é basicamente por questão de performance. Estes são armazenados separadamente das tabelas, o usuário não tem acesso aos mesmos.

É dividido em 3 três tipos:

- **Unique:** Criado a partir da chave primária, não permitindo a inclusão de linhas duplicadas, ou seja, força a inicidade dos dados através dos valores de uma coluna especificada como chave única;
- **Cluster**: A seqüência física das linhas procura obedecer à seqüência das entradas do índice. Existem utilitários que reorganizam um tablespace através do índice cluster de suas tabelas; e;
- **Normal**

4.5 Storage Group

É um conjunto de volumes (discos) onde estarão os tablespaces e índices. Sua utilização nos poupa da codificação do AMS de VSAM para definirmos data sets de um tablespace.

Os discos de um storage group não precisam ser dedicados ao DB2.

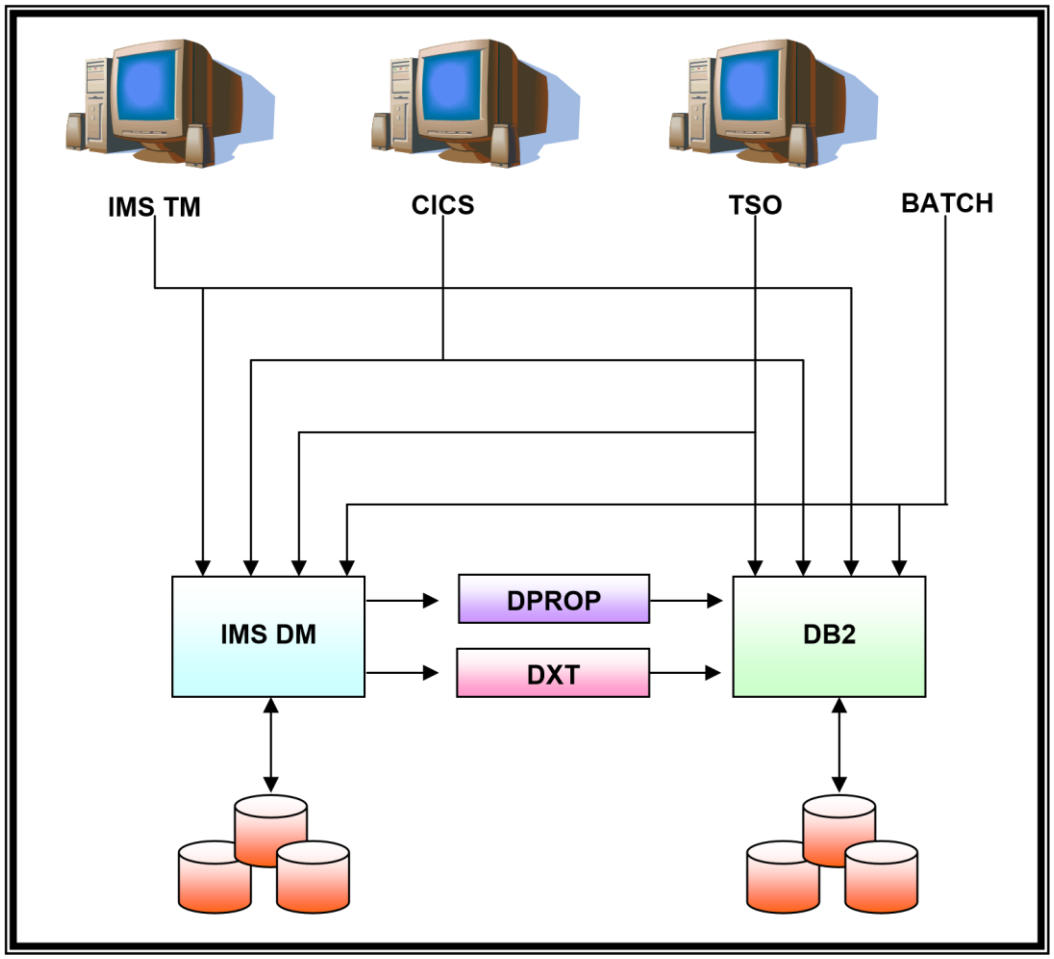
4.6 Synonym (sinônimo)

É um nome alternativo ou apelido, para uma tabela ou uma view. Uma vez criado, a sua utilização é restrita a um auth-id e válido em nível de subsistema DB2.

4.7 Alias

É um nome alternativo ou apelido, para uma tabela ou view. Uma alias é de uso compartilhado ao contrário do sinônimo. É utilizado mais como apelido para uma tabela remota e é valido em nível de subsistema DB2.

4.8 Ambiente DB2



Observação:

O DB2 pode ser acessado através do IMS, CICS, TSO e jobs BATCH simultaneamente. Dados que estiverem em IMS/DM (antigo IMS/DB) podem ser transferidos para o DB2 através do DPROD (Data Propagator) ou DXT (Data Extract).

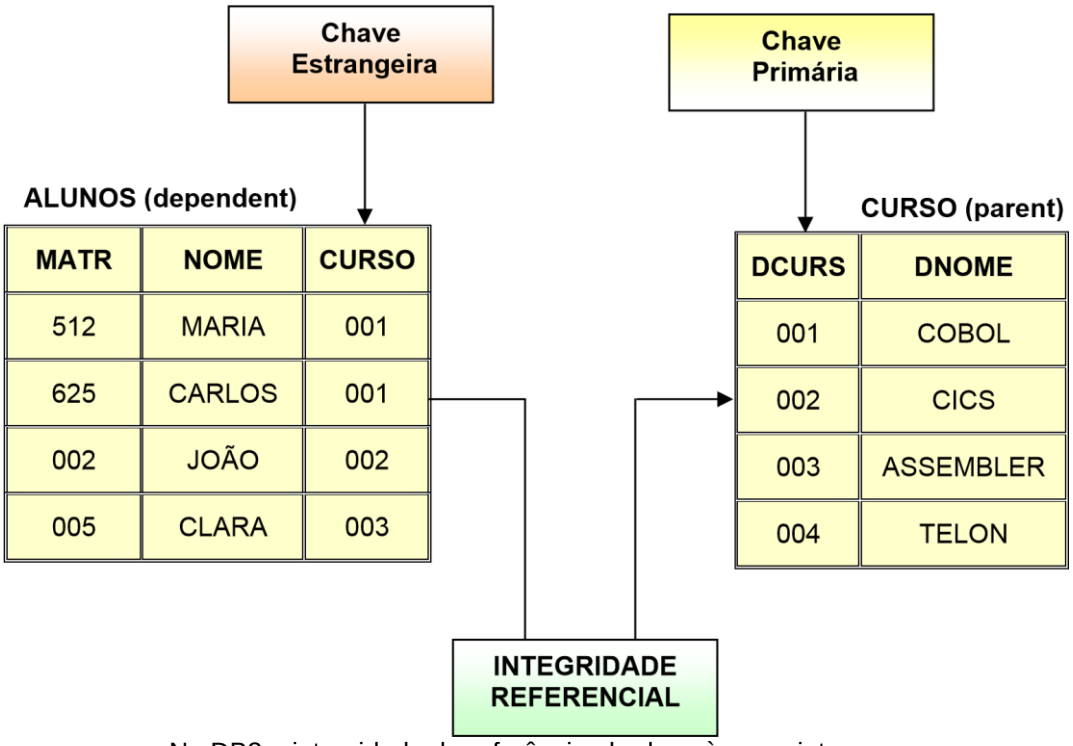
4.9 Projetos com DB2

4.9.1 Integridade de referência

Garante que qualquer valor apontado pela chave estrangeira tenha uma valor correspondente na chave primária à qual está associada.

Esta integridade de referência é implementada no DB2 via DLL (instruções de definição de objetos na linguagem SQL).

Modelo:



MATR	NOME	CURSO
512	MARIA	001
625	CARLOS	001
002	JOÃO	002
005	CLARA	003

DCURS	DNOME
001	COBOL
002	CICS
003	ASSEMBLER
004	TELON

No DB2 a integridade de referência obedece às seguintes regras:

- Uma chave primária tem valor único e não nulo;
- Uma chave estrangeira é nula ou tem valor correspondente na chave primária; tanto na inserção como na atualização de linhas, o DB2 força a observação destas regras;
- Todas as tabelas associadas entre si, via regra de integridade de referência, formam um conjunto chamado de *estrutura referencial* ou *referencial structure*;
- Uma tabela com chave primária definida é chamada de *parent table* ou *tabela mãe*;
- Uma tabela com chave estrangeira definida é chamada de *dependent table* ou *tabela dependente*;
- Uma linha de uma *parent table* pe chamada de *parent row* ou *linha mãe* se existir pelo menos uma linha na tabela dependente com chave estrangeira correspondente à sua chave primária;
- Uma linha de uma *dependent table* pe chamada de *dependent row* ou *linha dependente* se existir uma linha mãe com valor de chave primária correspondente a da sua chave estrangeira, ou seja, a chave estrangeira não deve ter valor nulo para ser uma linha dependente;
- Na eliminação de uma linha da tabela mãe, podem ocorre situações conforme a regra de deleção estabelecida pela chave estrangeira;
  - **CASCADE**: regra onde todas as linhas dependentes são eliminadas juntamente com a mãe;
  - **SET NULL**: regra onde as chaves estrangeiras das linhas dependentes são atualizadas com valor nulo após eliminação da mãe; e;
  - **RESTRICT**: regra onde a eliminação da linha mãe é impedida. Neste caso somente as linhas que não são mães podem ser eliminadas.

4.10 Acesso ao DB2

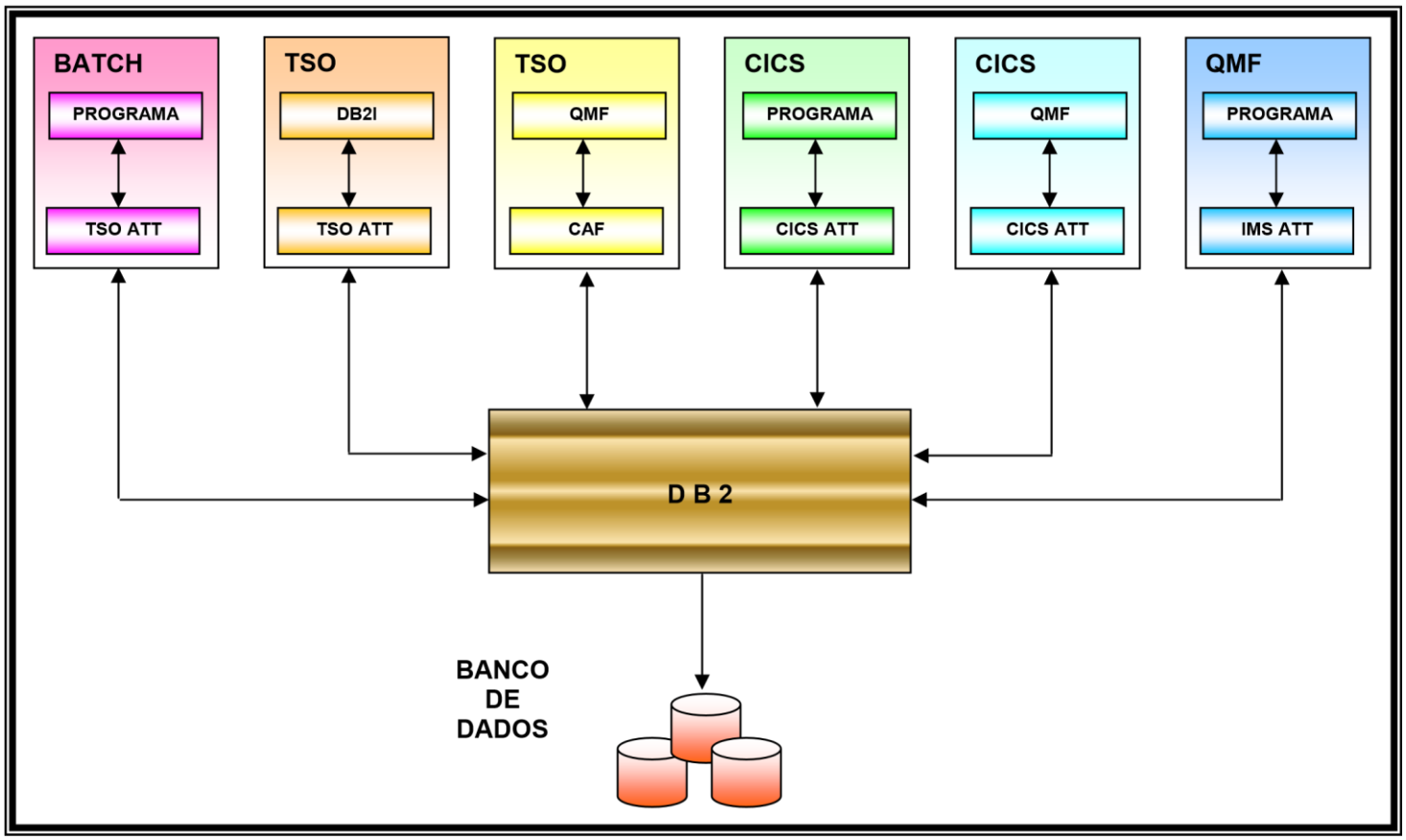
Os Attachment Facilities são interfaces que permitem aos programas a emissão de CALLs para o DB2.

Para cada monitor de transação (CICS, IMS, TSO etc) temos um attachment específico. Para aqueles que não vão trabalhar com estes monitores existe a alternativa de se utilizar o CAF (Call Attachment Facility).

O DB2 é um programa que é instalado com o DB2 e utiliza o TSO Attachment. O DB2 oferece uma série de recursos e é basicamente uma ferramenta para os que atuam na área de suporte.

O QMF é um programa produto que pode ser executado sob o TSO ou sob o CICS, lançando mão do TSO Attachment ou do CICS Attachment.

O QMF é um gerador de relatórios que acessa tabelas relacionais. É voltado para o usuário final, mas é comum a sua utilização pelo pessoal de suporte.



4.11 DB2 Interative

- Ferramenta interativa
- Recursos

- Execução de instrução SQL (SPUFI);
- Execução de comando s DB2;
- Preparação de programas de aplicação; e; Execução de utilitários.

4.11.1 SPUFI

SQL PROCESSING USING FILE INPUT.

É uma ferramenta para executar instruções SQL.

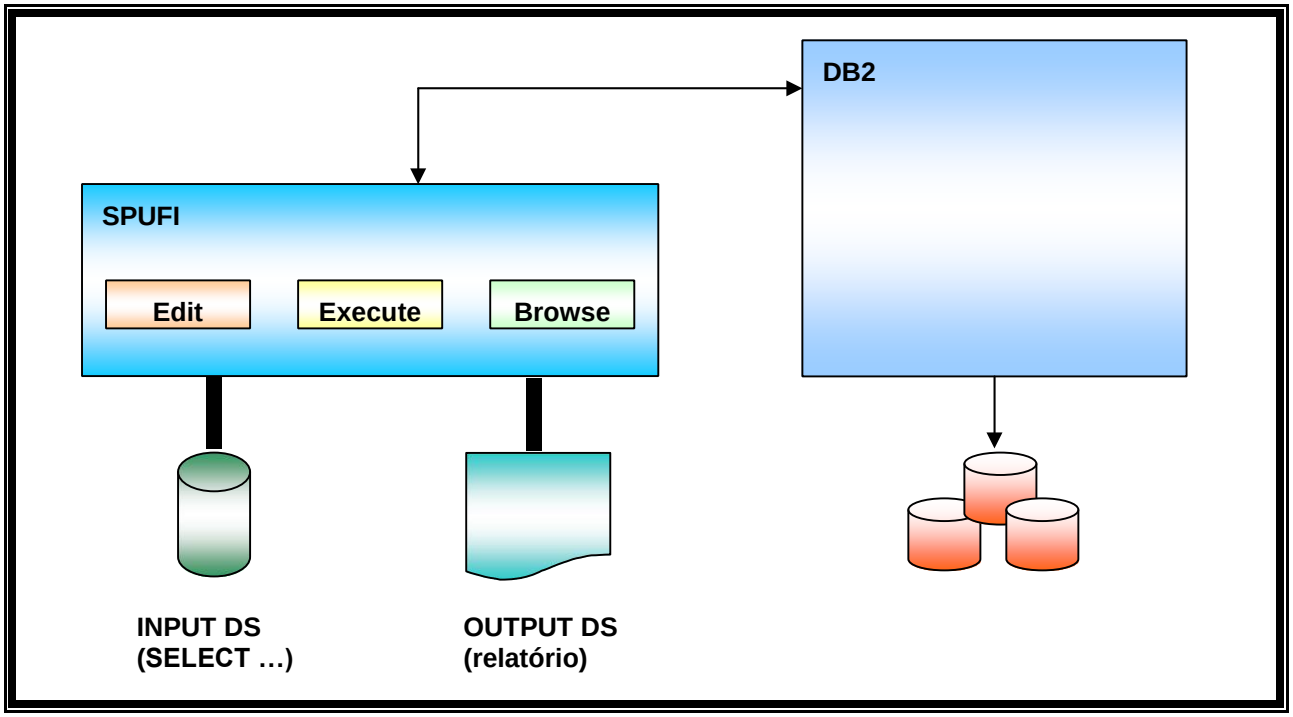
Trabalha com um arquivo de entrada e um arquivo de saída.

O SPUFI dirige o seu usuário para um ciclo de trabalho que consiste basicamente de três fases:

• *Edit*
  - Edição de arquivo de entrada onde é codificada uma ou mais instruções SQL;

• *Execute*
  - O DB2 interpreta a instrução SQL do arquivo de entrada e a executa;

• *Browse*
  - Mostra o arquivo de saída com um relatório do resultado da execução;



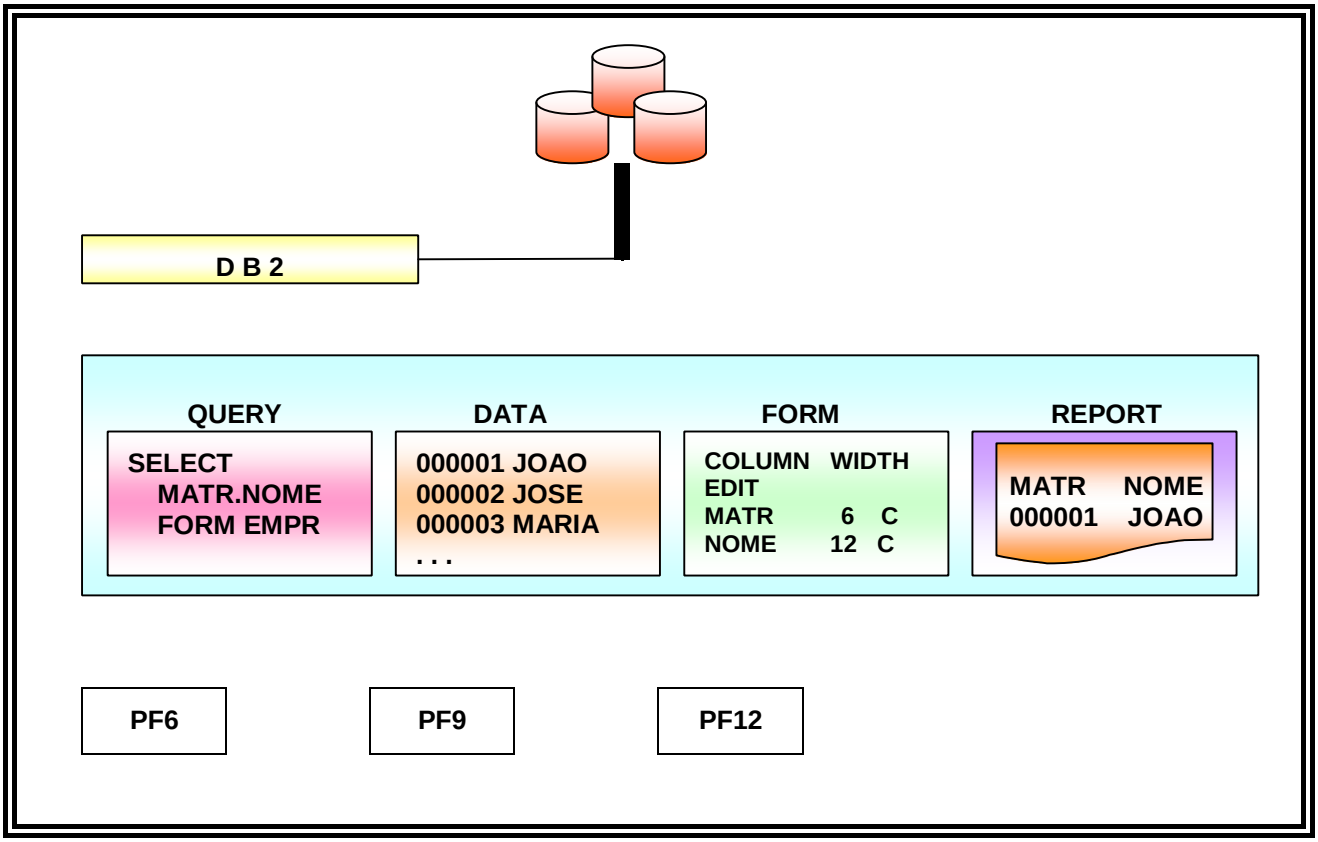
4.12 QMF

O **Q**uery **M**anagement **F**acility é um gerador de relatórios que pesquisa tabelas relacionais.

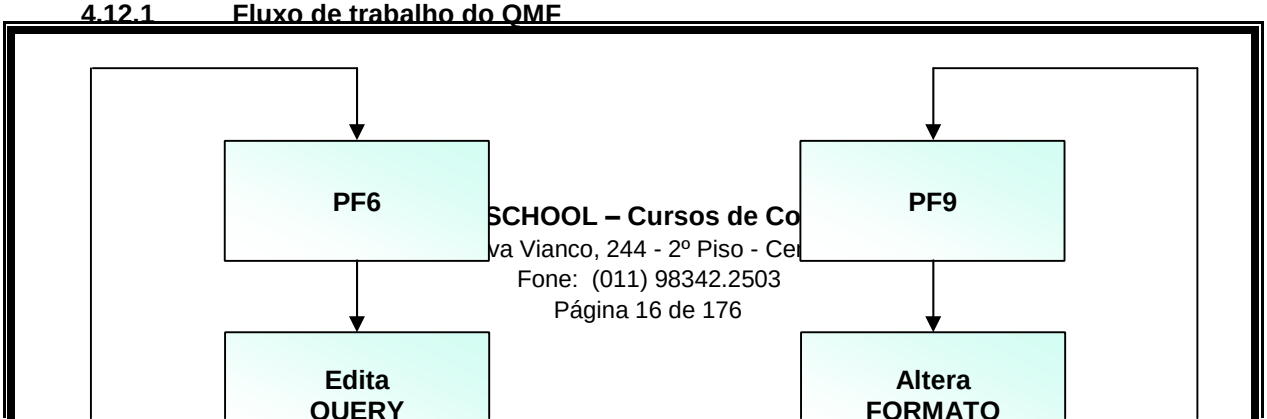
A memória do QMF é dividida em várias área, sendo as mais utilizadas:

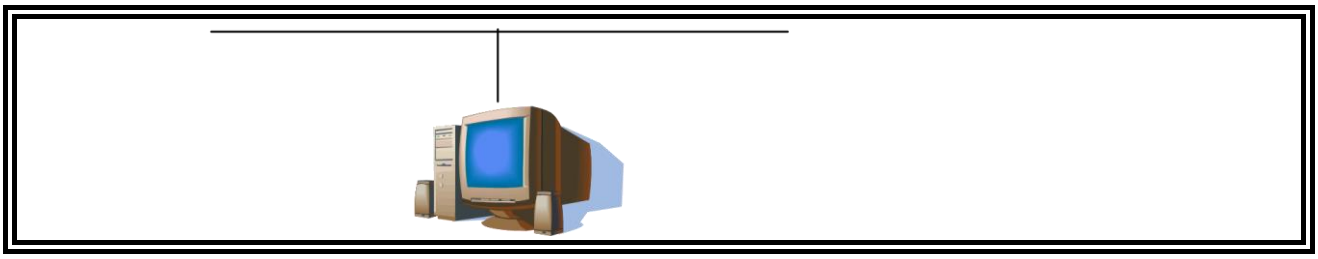
- *QUERY*
  - Entrada da instrução SQL;
- *DATA*
  - Resultado devolvido pelo DB2;
- *FORM*
  - Formato do relatório de saída;
- *REPORT*
  - Relatório montado com base nos dados da área DATA e no formato especificado em FORM.

O QMF trabalha também com relatórios gráficos usando para isto a interface com GDDM.



4.12.1 Fluxo de trabalho do QMF





O QMF dirige o usuário a um ciclo de trabalho tam como no SPUFI.

A diferença é que no QMF temos a possibilidade de trabalhar o relatório, enquanto que no SPUFI o relatório pe padronizado e com poucas alternativas de alteração.

O fluxo de trabalho do QMF tem 2 (dois) ciclos:

1. Obtenção dos dados
   No ciclo de obtenção dos dados, trabalhamos interessados somente no conteúdo do resultado de uma instrução. O ciclo é percorrido através de algumas teclas PFs:
   - **PF6** Mostra a área da QUERY para editarmos uma instrução SQL; - **PF2** Executa a instrução e mostra o relatório resultante.
2. Refinamento do relatório
   Neste ciclo trabalhamos interessados na apresentação final do relatório que resultou do ciclo anterior.
   O ciclo é percorrido através de algumas teclas PFs:

- **PF9** Mostra a área de FORM para alterarmos a especificação do formato do relatório. É nesta etapa que fornecemos o cabeçalho, quebra de paginas, rodapés etc
- **PF12** Mostra o relatório editado a partir de informações contidas na área de DATA e na área de FORM.

4.12.2 Editando relatório

Uma das características principais do QMF está na sua capacidade de permitir a elaboração de relatórios complexos.

Para se editar um relatório, a área de dados deve estar carregada via execução de uma query.

Exemplo:

- Digitada a tecla PF9, o QMF exibe a área de FORM, que vem inicialmente carregado com um formato default;
- No exemplo serão mostrados quatro alterações no formato default:
  1. Acréscimo do cabeçalho "**RELATORIO RH**";
  2. Alteração do **NOME** do cabeçalho da coluna **NOME** para **FUNCIONÁRIO**;
  3. Alteração do **NOME** do cabeçalho da coluna **SAL** para **SALARIO**; e;
  4. Alteração do código de edição da coluna SALARIO para que seja exibida o caractere "$".
- Outras alterações possíveis no formato default:
  a) Acréscimo de rodapé;
  b) Quebra de página por valor de coluna;
  c) Cabeçalho e rodapé para cada quebra de página;
  d) Quebras encadeadas em múltiplos níveis;
  e) Cabeçalhos e rodapés com texto dependente dos doados;
  f) Expansão dos dados de uma linha em múltiplas linhas; e;
  g) Quebra e relatórios grandes (mais de 132 caracteres) em páginas físicas menores;

```
FORM MAIN
  Columns:

 NUM   COLUMN  HEADING   EDIT
    1   NOME                C
2   SAL                 L2


PAGE HEADING:
```

```
FORM MAIN
  Columns:

 NUM   COLUMN  HEADING   EDIT
1   FUNCIONARIO         C
2   SALARIO             L2

PAGE HEADING: RELATORIO RH
```

REPORT

NOME	SAL
JOAO	4500,00
JOSE	2600,00
MARIA	4800,00
MANOEL	3200,00

REPORT

RELATORIO RH

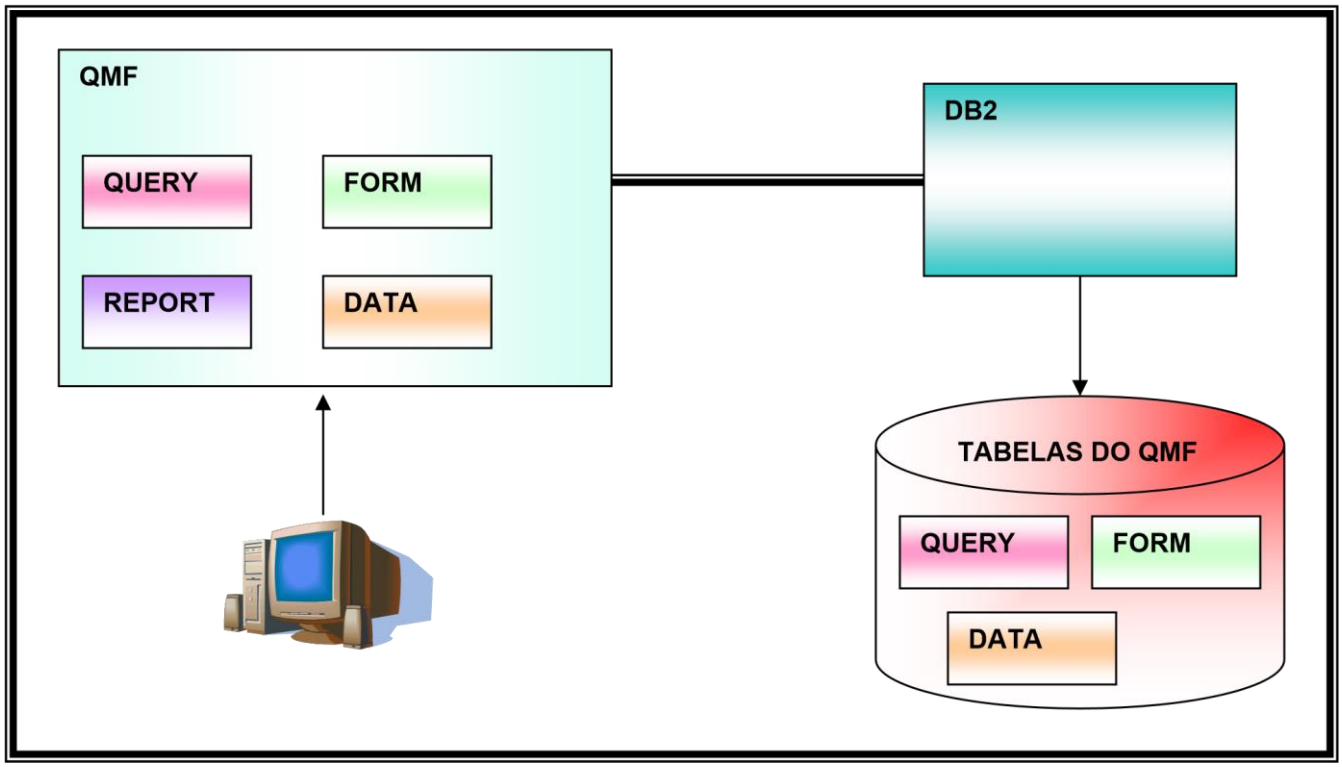
FUNCIONARIO	SALARIO
JOAO	$ 4500,00
JOSE	$ 2600,00
MARIA	$ 4800,00
MANOEL	$ 3200,00

4.12.3 Salvando a memória

As áreas de memória do QMF podem ser salvas para posterior utilização.

As informações são salvas em tabela de controle do QMF, que ficam sob controle do DB2.

ser utilizada para cria aplicações de uso geral.



As informações podem ser salvas para uso compartilhado. Esta capacidade pode

5 Tabelas a serem usadas nos exemplos

SAL_COMIS (tabela de salários / comissões)

COD	NOME	DEPTO	CARGO	ANOS	SALARIO	COMIS

10	JOAO	20	GER	7	18357.50	?
20	JOSE	20	VENDAS	8	18171.50	612.45
30	PAULO	38	GER	5	17506.75	?
40	MESSIAS	38	VENDAS	6	18006.00	846.55
50	CLEIDE	15	GER	10	20659.80	?
60	CARLOS	38	VENDAS	0	16808.30	650.25
70	MARIA	15	VENDAS	7	16502.83	1152.00
80	NELSON	20	ATEND	0	13504.60	128.20
90	LUIS	42	VENDAS	6	18001.75	1386.70
100	RAFAEL	42	GER	7	18352.80	?
110	TEREZA	15	ATEND	5	12508.20	206.60
120	ROBERTA	38	ATEND	0	12954.75	180.00
130	ADRIANA	42	ATEND	6	10505.90	75.60
140	CELSO	51	GER	6	21150.00	?
150	CAMILA	51	VENDAS	6	19456.50	637.65
160	CARLA	10	GER	7	22959.20	?
170	EDNA	15	ATEND	4	12258.50	110.10
180	CASSIA	38	ATEND	3	12009.75	236.50
190	LIGIA	20	ATEND	8	14252.75	126.50
200	PAULA	42	ATEND	0	11508.60	84.20
210	FERNANDA	10	GER	10	20010.00	?
220	LETICIA	51	VENDAS	7	17656.50	992.80
230	SERGIO	51	ATEND	3	13369.80	189.65
240	RICARDO	10	GER	5	19260.25	?
250	MAURO	51	ATEND	6	14460.00	513.30
260	OSWALDO	10	GER	12	21234.00	?
270	IVETE	66	GER	9	18555.50	?
280	SIDNEY	66	VENDAS	9	18674.50	811.50
290	MARLI	84	GER	10	19818.00	?

300	WILSON	84	VENDAS	5	15454.50	806.10
310	HELIO	66	VENDAS	13	21000.00	200.30
320	ELZA	66	VENDAS	4	16858.20	844.00
330	AUREA	66	ATEND	1	10988.00	55.50
340	MARCIA	84	VENDAS	7	17844.00	1285.00
350	LEA	84	ATEND	5	13030.50	188.00

COD_GEREN (tabela de código de departamentos)

DEPTCOD	DEPTNOME	GERENTE	DIVISAO	LOCAL
10	MATRIZ	160	CENTRO	SAO PAULO
10	MATRIZ	210	CENTRO	SAO PAULO
10	MATRIZ	240	CENTRO	SAO PAULO
10	MATRIZ	260	CENTRO	SAO PAULO
15	CAMPINAS	50	SUDESTE	CAMPINAS
20	RIO	10	SUDESTE	RIO DE JANEIRO
38	MINAS	30	SUDESTE	BELO HORIZONTE
42	BAHIA	100	NORDESTE	SALVADOR
51	RECIFE	140	NORDESTE	RECIFE
66	RIO GRANDE	270	SUL	PORTO ALEGRE
84	PARANA	290	SUL	CURITIBA

TAB_PEDIDOS (tabela de pedido de vendas)

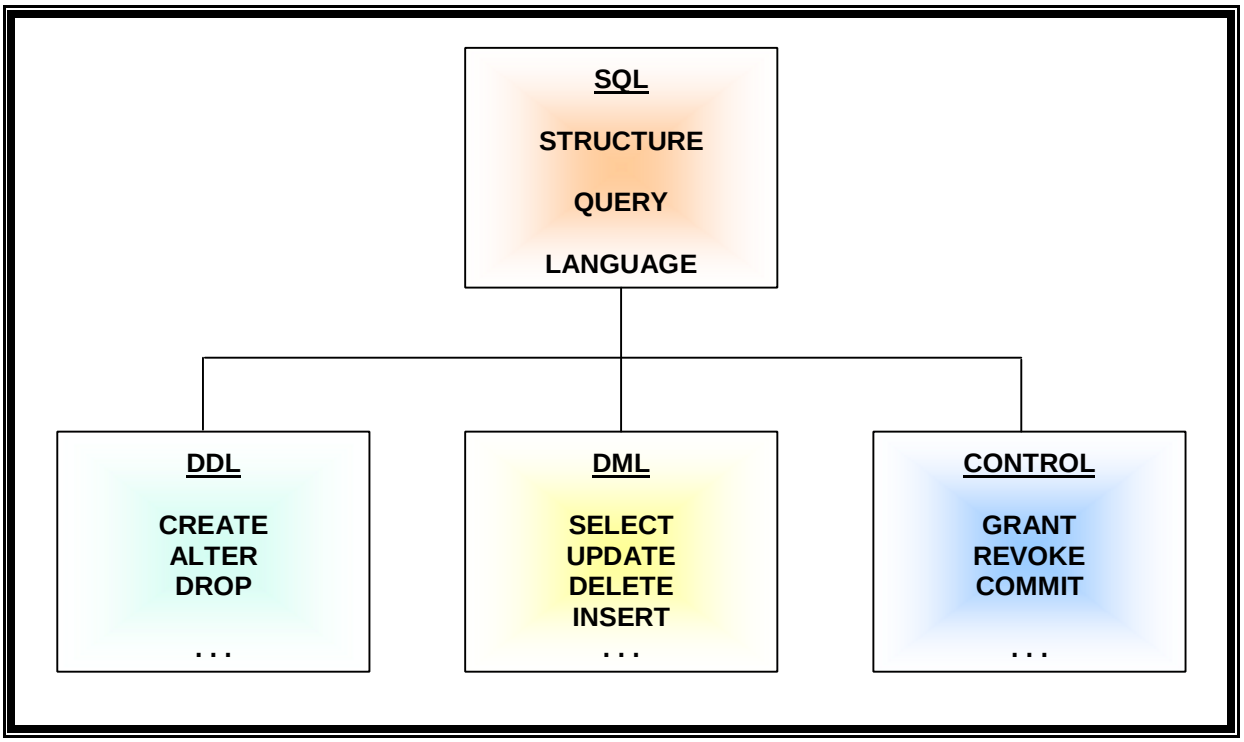
COD_VENDEDOR	COD_PEDIDO
10	1234
20	3456
20	6667
20	3580
20	7010
30	1478
40	5687
40	6597
40	6540
70	9875
80	9963
90	3252
100	3654
110	1548
120	9854
130	3654
140	4664
150	4466
160	4335

COD_VENDEDOR	COD_PEDIDO
170	5589
180	1269
190	6932
200	3265
210	5459
220	9597
230	9893
240	3536
250	2021
260	2326
270	6861
280	3637
290	3839
300	4041
310	4243
320	4445
330	4658
340	6580
350	2568

6 SQL

Structured Query Language (SQL) representa um conjunto de comandos responsáveis pela definição das tabelas, comandos e atualizações dos dados em um SGBD, está separada em 3 (três) grupos:

- **Comandos DDL** (Data Definition Language);
- **Comandos DML** (Data Manipulation Language); e;
- **Comandos CONTROL** – Utilizado para administrar a segurança dos recursos DB2.



6.1 Comandos DDL

Conjunto de comandos responsáveis pela criação, alteração e deleção das estrutura das tabela e índices de um sistema.

A execução de DDL é permitido somente à ***auth-id's*** devidamente autorizados. O criador (auth-id) do objeto é normalmente o proprietário (owner) do mesmo, porém o DB2 permite que sejam criados objetos oara outros auth-id, ou seja, posso criar objetos dos quais não serei propritário.

O proprietário de um objeto tem todos os privilégios sobre o mesmo. A equipe de suporte (SYSADM, SYSCTRL, DBADM) ou proprietário do objeto é o responsável pela concessão de autorizações para acesso e/ou uso dos objetos.

A execução das instruções ALTER ou DROP stão reservadas à equipe de suporte ou ao proprietário dos objetos.

6.1.1 Storage Group (create e alter)

O nome de um storage group não deve ultrapassar 8 (oito) caracteres. Podem ser especificados até 133 volumes do mesmo periférico no parâmetro VOLUMES.

VCTA aponta para o catálogo ICF que qualificará os datasets dos tablespaces e dos indexspaces.

Exemplo:

```
CREATE STOGROUP FUTUVOLS VOLUMES (FUTUDB2X, FUTUDB2Y) VCAT DB2CATLOG

ALTER STOGROUP FUTUVOLS ADD   VOLUMES (FUTUIMSX)
                        REMOVE (FUTUDB2Y)
```

6.1.2 Data Base (create)

O exemplo que mostraremos, é a criação de um Data Base de nome FUTUDB01, os parâmetros como Storage Group e Bufferpool podem ser fornecidos. Os valores especificados serão utilizados em case de omissão durante a definição de objetos que pertençam a este Data Base.

Exemplo:

```
CREATE DATABASE FUTUDB01
```

6.1.3 Tablespace (create e alter)

O exemplo que mostraremos, mostra a definição de um tablespace segmentado.

Exemplo:

```
CREATE TABLESPACE FUTUTS01
IN FUTUDB01
USING STOGROUP FUTUVOLS
PRIQTY 200
SECQTY 20
ERASE NO
LOCKSIZE ANY
BUFFERPOOL BP0
CLOSE YES
FREEPAGE 4
PCTFREE 25
SEGSIZE 8

ALTER TABLESPACE FUTUTS01
PRIQTY 400
```

onde:

Conteúdo	Descrição
IN	Especifica o data base que vai conter o tablespaces. Default é **DSNB04**.
USING STOGROUP	Especifica o storage group que conterá fisicamente o dataset do tablespace. O valor default é obtido do data base ou do sistema.
PRIQTY	Alocação primária em Kbytes para o dataset do tablespace. O número debe ser inteiro. A alocação real será o valor dividido por 4 (32 se páginas de 32K) e arredondado para cima. O valor default é **3 (três)**.
SECQTY	Alocação secundária em Kbytes para o dataset do tablespace. O número debe ser inteiro.
ERASE	Especifica se o espaço ocupado pelos datasets devem ser preenchidos com zero quando forem eliminados (DROP). O default é **NO**.
LOCKSIZE	Especifica a granularidade do locking, portanto é um parâmetro de performance, pode ser: **ROW / PAGE** (locking em nível de linha ou página), **ANY** (é o valor default para p parâmetro LOCKSIZE, com este valor a decisão pela granularidade fica por conta do DB2), **TABLE** (locking em nível de tabela) e **TABLESPACE** (locking em nível de tablespace)
BUFFERPOOL	Importante parâmetro que determina o tamanho das páginas do tablespace. Este valor aponta para o conjunto de buffers que será associado com o tablespace. As opções são **BP0** / **BP1** (4K) ou **BP32K** (32K)
CLOSE	Especifica se o dataset do tablespace debe ser fechado quando não está sendo utilizado. O default é **YES**.
FREEPAGE	Especifica o intervalo de páginas que devem existir entre as páginas livres quando um tablespace é carregado ou reoprganizado. O intervalo pode variar de o(sem página livre) à 255. Se o tablespace é segmentado, o número debe ser menor que o tamanho do segmentado. Se for especificado um intervalo maior que o tamanho do segmento, o DB2 adotará o tamanho do segmento m,enos 1. O valor default é **0 (zero)**.
PCTFREE	Especifica qual é a fração percentual das páginas que devem ser mantidas livres quando o tablespace é carregado ou reorganizado. O valor pode variar de 0 à 99. O default é **5 (cinco).**
SEGSIZE	Se especificado, determina que o tablespace será segmentado com um certo tamanho. O tamanho de um segmento é dado em qualquer quantidade de páginas, deve ser inteiro múltiplo de 4, variando de 4 à 64.
NUMPARTS	Se especificado no lugar de SEGSIZE, determina que o tablespace será particionado. É acompanhado de um número que indica a quantidade de partições que deve ter um tablespace. O número pode variar de 1 à 64.

Observações:

- Se sem SEGZISE e nem NUMPARTS forem especificados, o tablespace será simples;
- Muitos do parâmetros do tablespace podem ser alterados;

- Não podem ser alterados: o data base que contém o seu tipo simples, segmentado ou particionado.

6.1.4 Tabela

6.1.4.1 Create

Criar a estrutura de uma tabela (arquivo) definindo as colunas e as chaves (primárias e estrangeiras) existentes.

```
CREATE TABLE <nome tabela>
(<nome coluna>, <tipo dado>, [NOT NULL]
                             [NOT NULL WITH DEFAULT]) PRIMARY
KEY (nome coluna chave)
FOREIGN KEY (nome coluna chave estrangeira) REFERENCES
            (nome tabela mãe) ON DELETE [RESTRICT]
                                        [CASCADE]
                                        [SET NULL]
```

onde:

Conteúdo	Descrição
nome tabela	Representa o nome da tabela que será criada
nome coluna	Representa o nome da coluna que será criada. A definição das colunas de uma tabela é feita relacionando-as uma após a outra.
tipo dado ***	Cláusula que define o tipo e tamanho dos campos definidos para a tabela.
NOT NULL	Exige o preenchimento do campo, ou seja, no momento da inclusão é obrigatório que possua um conteúdo.
NOT NULL WITH DEFAULT	Preenche o campo com valore pré definidos, de acordo com o tipo do campo, caso não seja especificado o seu conteúdo no momento da inclusão do registro. Os valores pré definidos são: **ZEROS** para campos **numéricos**; **BRANCOS** para campos **alfanuméricos**, **DATA CORRENTE** para campos **date** e **HORÁRIO DO MOMENTO DA OPERAÇÃO** para campos **time**.
PRIMARY KEY	Define para o banco de dados a coluna que será a chave primária da tabela (nome da coluna chave). Caso ela tenha mais de uma coluna como chave, elas deverão ser relacionadas entre parênteses.
FOREIGN KEY... REFERENCES	Define para o banco de dados as colunas que são chaves estrangeiras, ou seja, os campos que são chaves primárias de outras tabelas, na opção REFERENCE deve ser especificado a tabela na qual a coluna é a chave primária.
ON DELETE	Esta opção especifica os procedimentos que devem ser feitos pelo SGBD quando houver uma exclusão de um registro na tabela mãe quando existe um registro correspondente nas tabelas filhas. As opções disponíveis são: **RESTRICT**: opção default, permite a exclusão na tabela mãe de um registro cuja chave primária exista em alguma filha; **CASCADE**: realiza a exclusão em todas as tabelas filhas que possua o valor da chave que será excluída da tabela mãe; **SET NULL**: atribui o valor NULO nas colunas das tabelas folhas que contenha o valor da chave que será excluída na tabela mãe.

*** Tipos da dados mais comuns:

1. Numéricos:
   - **SMALLINT** – Armazena valores em dois bytes binários, compreendidos entre o intervalo -32768 a +32767;
   - **INTEGER** – Armazena valores em quatro bytes binários, compreendidos entre o intervalo -2147483648 a +2141423647;
   - **DECIMAL(n,m)** – Armazena valores com no máximo 15 dígitos. Nesta opção deve ser definida a quantidade de dígitos inteiros (n) e casa decimais (m) existentes no campo.

2. Alfanuméricos:
   - **VARCHAR(n)** – Define um campo de até n caracteres, onde n deve ser menor ou igual a 254;
   - **CHAR(n)** – Define um campo de n caracteres, onde n der ser menor ou igual a 254;
   - **LONG VARCHAR** – Define um campo de comprimento maior que 254 caracteres.

3. Date – Define um campo que irá armazenar DATAS.

4. Time – Define um campo que irá armazenar o horário do momento da operação.

6.1.4.2 Alter

Alterar a estrutura de uma tabela (arquivo) acrescentando, alternado, retirando e alterando nomes, formatos de colunas e a integridade referencial definidas em uma determinada tabela.

```
ALTER  TABLE <nome tabela>
 DROP   <nome coluna>
 ADD    <nome coluna> <tipo dado>, [NOT NULL]
                                   [NOT NULL WITH DEFAULT]
 RENAME <nome coluna> <novo nome coluna>
 RENAME TABLE <novo nome tabela>
 MODIFY <nome coluna> <tipo dado>, [NULL]
                                   [NOT NULL]
                                   [NOT NULL WITH DEFAULT]
 ADD  PRIMARY KEY (nome coluna)
 DROP PRIMARY KEY (nome coluna)
 ADD  FOREIGN KEY (nome coluna chave estrangeira) REFERENCES
                  (nome tabela mãe) ON DELETE [RESTRICT]
                                              [CASCADE]
                                              [SET NULL]
 DROP FOREIGN KEY (nome coluna chave estrangeira) REFERENCES
                  (nome tabela pai) onde:
```

Conteúdo	Descrição
nome tabela	Representa o nome da tabela que será atualizada
nome coluna	Representa o nome da coluna que será criada.
tipo dado	Cláusula que define o tipo e tamanho dos campos definidos para a tabela.
DROP <nome coluna>	Realiza a retirada da coluna especificada na estrutura da tabela.
ADD <nome coluna> <tipo do dado>	Realiza a inclusão da coluna especificada na estrutura da tabela. Na coluna correspondente a este campo nos registros já existentes será preenchido o valor NULL (nulo). As definições NOT NULL e NOT NULL WITH DEFAULT são semelhantes à do comando CREATE TABLE.
RENAME <nome coluna> <novo nome coluna>	Realiza a troca do nome da coluna especificada.
RENAME TABLE <novo nome tabela>	Realiza a troca de nome da tabela especificada.
MODIFY <nome coluna> <tipo do dado>	Permite a alteração na característica da coluna especificada, utilizando-se das definições descritas na opção ADD e também da opção NULL que altera a característica do campo passando a permitir o preenchimento com o valor nulo.
ADD PRIMARY KEY <nome coluna>	Esta opção é utilizada quando é acrescido um novo campo como chave primária da tabela.

DROP PRIMARY KEY <nome coluna>	Esta opção é utilizada quando é retirado um campo como chave primária da tabela.
ADD FOREIGN KEY <nome coluna>	Esta opção é utilizada quando é acrescido um novo campo, sendo ele, uma chave estrangeira.
DROP FOREIGN KEY <nome coluna>	Esta opção é utilizada quando é retirado uma chave estrangeira da estrutura da tabela.

6.1.4.3 Create Index

Criar uma estrutura de índice de acesso para uma determinada coluna em uma tabela. Um índice de acesso permite um acesso mais rápido aos dados em uma operação de seleção. Os índices poder sem criados a partir de um ou mais campos de uma tabela.

```
CREATE [UNIQUE] INDEX <nome índice>
  ON <nome tabela) (<nome coluna> [ASC ],[<nome coluna> [ASC ]])
                                  [DESC]                [DESC]
USING STOGROUP FUTUVOLS
PRIQTY 200
SECQTY 20
ERASE NO
BUFFERPOOL BP0
CLOSE NO
FREEPAGE 4
PCTFREE 25
```

Onde:

Conteúdo	Descrição
nome índice	Representa o nome da estrutura de índice que será criada.
nome tabela	Representa o nome da tabela que contém a coluna na qual será criada o índice de acesso.
nome coluna	Representa o nome da coluna que será criada.
ASC / DESC	Representa a forma ordenada de criação do índice, onde ASC é ascendente e DESC é decrescente.

Observação:

Demais conteúdos, já descritos no item 6.1.3 Tablespace

6.1.4.4 Alter Index

Alterar uma estrutura de índice de acesso para uma determinada coluna em uma tabela..

```
ALTER  INDEX <nome índice>
   USING STOGROUP FUTUVOL1
```
Onde:

Conteúdo	Descrição
nome índice	Representa o nome da estrutura de índice que será deletada.

Observação:

Demais conteúdos, já descritos no item 6.1.3 Tablespace.

6.1.5 Synonym e Alias (create)

Exemplo:

```
CREATE SYNONYM COLABORADOR
       FOR FUTUDB01.COLAB

CREATE ALIAS   CONSULSP
       FOR SP.FUTUDB01.CONS
```

6.1.6 View (create)

Exemplo:

```
CREATE VIEW SALARIO
       (DEPTO, SALARIO)
       AS
         SELECT DEPTO, SALARIO
FROM SAL_COMI
         WHERE SALARIO < 10000
```

Observação:

A View acima não permite que o usuário atualize a coluna SALARIO de qualquer linha para um valor maior oi igual a 10000.

6.1.7 Drop

6.1.7.1 Database

Deletar a estrutura e os dados existentes em um Data Base.

```
DROP   DATABASE  <nome database>
```

6.1.7.2 Tablespace

Deletar a estrutura e os dados existentes em um Tablespace.

```
DROP   TABLESPACE  <nome tablespace>
```

6.1.7.3 Table

Deletar a estrutura e os dados existentes em uma Tabela.

```
DROP   TABLE <nome tabela>
```

6.1.7.4 View

Deletar a estrutura e os dados existentes em uma View.

```
DROP   TABLE <nome view>
```

6.1.7.5 Synonym

Deletar a estrutura e os dados existentes em uma Synonym.

```
DROP   SYNONYM <nome synonym>
```

6.1.7.6 Index

Deletar uma estrutura de índice de acesso para uma determinada coluna em uma tabela..

```
DROP   INDEX <nome índice>
```

6.1.7.7 Stogroup

Deletar a estrutura e os dados existentes em uma Stogroup.

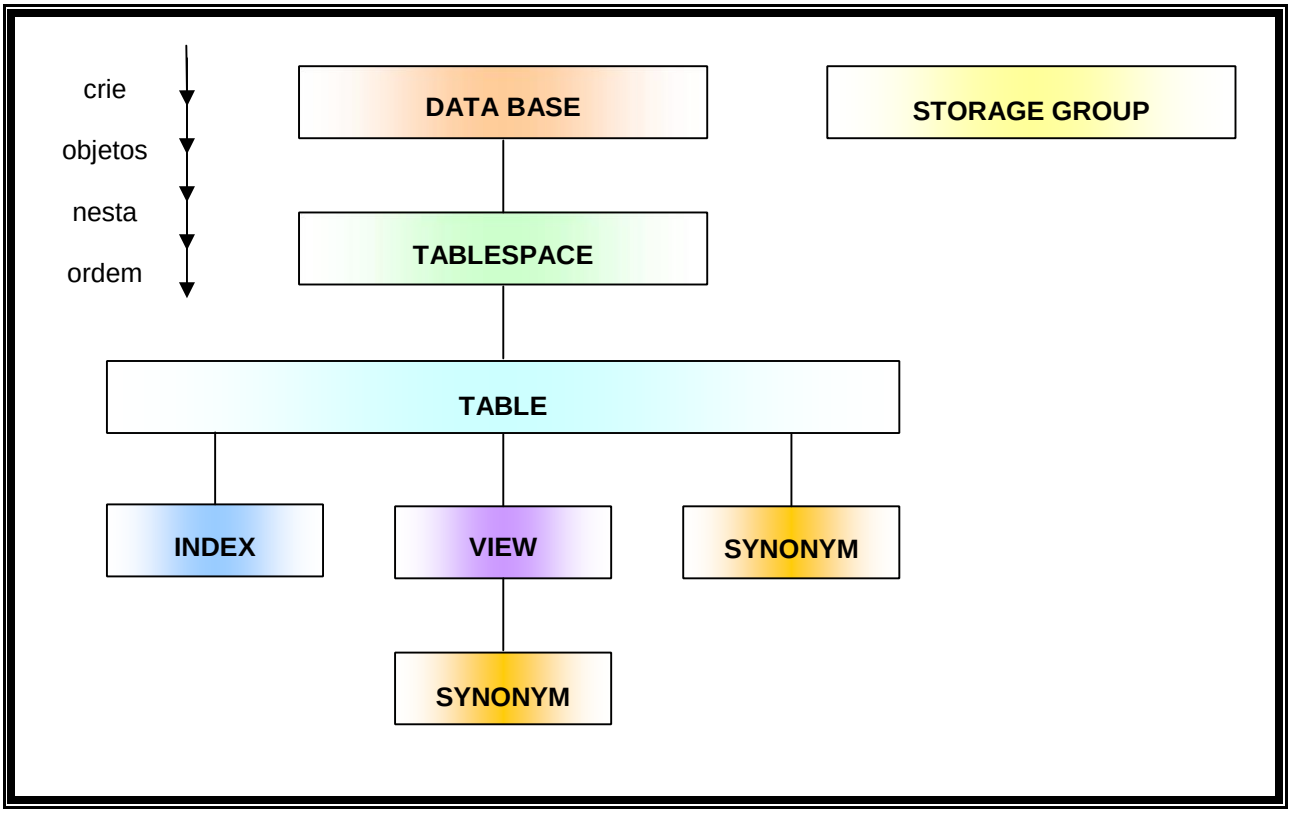
```
DROP   SYNONYM <nome stogroup
```

6.1.8 Dependência entre objetos

Os objetos DB2 são quase todos dependentes entre si, ou seja, ao definirmos um índice é preciso associá-lo à uma tabela, se esta tabela for eliminada, o índice perde a razão de sua existência e também será eliminado.

Um storage group pode ser eliminado sem problemas, a partir de sua eliminação os datasets não serão mais criados nos seus volumes, os eventuais dados que estiverem residindo em um storage group permanecerão lá até que seja aplicado um utilitário de recuperação ou reorganização.

A eliminação de uma tabela não implica em eliminação do ALIAS que o aponta, pois esse passa a ser inválido.



6.2 Comandos DML

Conjunto de comandos responsáveis pela consulta e atualização dos dados armazenados em um banco de dados.

6.2.1 Select

É utilizado para selecionar um conjunto de registros em uma ou mais tabelas que atenda a uma determinada condição definida pelo comando.

A instrução SELECT não é procedural, ou seja, não precisam fornecer todos os procedimentos necessários para fazer uma pesquisa como:

1. Zere o contador;
2. Abra o arquivo;
3. Leia um registro;
4. Verifique se satisfaz o critério de pesquisa; e
5. Incremente o contador…

Com a instrução SELECT, basta dizer:
- Quero os dados que satisfaçam a condição X.

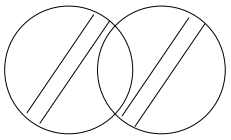
O SELECT trabalha com conjuntos e permite a execução de operadores elementares que nos são familiares.

Quem nunca ouvir falar dos operadores união ou intersecção?

Com uma instrução SQL podemos implementar os seguintes operadores:

- **União**

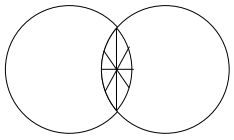
A ∪ B



A B

- **Intersecção**

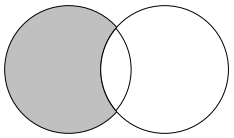
A ∩ B



A B

- **Diferença**

A - B



- **Produto cartesiano**

A X B = { (x,y) : x ∈ A e y ∈ B }

6.2.1.1 Query

As cláusulas de uma instrução SELECT devem ser especificadas obedecendo a uma seqüência. As cláusulas serão escritas em negrito, e não podem ser abreviadas.

No exemplo abaixo, será mostrada a seqüência obrigatória, estaremos fazendo uma pesquisa na tabela SAL_COMIS para gerar um relatório com o NOME, TEMPO DE ADMISSÃO e SALÁRIO de todos os colaboradores que estão alocados no DEPARTAMENTO 38, sendo que, este relatório deverá estar classificado (ordenado) pelo NOME dos funcionários:

```
     SELECT       NOME,
ANOS,
                  SALARIO
     FROM         SAL_COMIS
     WHERE        DEPTO      =       38
     ORDER BY     NOME
```

6.2.1.1.1 Formato livre

Uma instrução SELECT pode ser escrita em formato livre desde que as cláusulas estejam obedecendo à seqüência obrigatória.

Exemplo:

```
     SELECT NOME, ANOS, SALARIO FROM SAL_COMIS
     WHERE  DEPTO = 38 ORDER BY NOME
```

6.2.1.2 Select . . . From (determinadas colunas)

- **Select**
  - Listar as colunas desejadas na seqüência esquerda para a direita; e Usar vírgulas para separar a colunas desejadas.

- **From**
  - **Simples** EMP;
  - **Qualificado** AUTHID.EMP.

Exemplo:

```
     SELECT       DEPTONOME, DEPTOCOD
FROM         COD_GEREN
```

Resultado:

DEPTONOME	DEPTOCOD
MATRIZ	10
MATRIZ	10
MATRIZ	10
MATRIZ	10
CAMPINAS	15
RIO	20
MINAS	38
BAHIA	42
RECIFE	51
RIO GRANDE	66
PARANA	84

6.2.1.3 Select . . . From (todas colunas)

Para selecionar todas as colunas de uma tabela na seqüência em que foram definidas, basta utilizarmos o * (asterisco).

Exemplo:

```
SELECT * FROM COD_GEREN
```

Resultado:

DEPTCOD	DEPTNOME	GERENTE	DIVISAO	LOCAL
10	MATRIZ	160	CENTRO	SAO PAULO
10	MATRIZ	210	CENTRO	SAO PAULO
10	MATRIZ	240	CENTRO	SAO PAULO

10	MATRIZ	260	CENTRO	SAO PAULO
15	CAMPINAS	50	SUDESTE	CAMPINAS
20	RIO	10	SUDESTE	RIO DE JANEIRO
38	MINAS	30	SUDESTE	BELO HORIZONTE
42	BAHIA	100	NORDESTE	SALVADOR
51	RECIFE	140	NORDESTE	RECIFE
66	RIO GRANDE	270	SUL	PORTO ALEGRE
84	PARANA	290	SUL	CURITIBA

Observação:

Evitar o uso do * (asterisco).

6.2.1.4 Where

Utilizado para controle de linhas, ou seja, é a cláusula em que se estabelece critérios para seleção de linhas.

No exemplo abaixo, iremos listar somente EMPREGADOS do DEPARTAMENTO 20:

Exemplo:

```
     SELECT      DEPTO,
NOME,
                 CARGO,
                 COMIS
     FROM        SAL_COMIS
     WHERE       DEPTO = 20
```

Resultado:

DEPTO	NOME	CARGO	COMIS
20	JOAO	GER	?
20	JOSE	VENDAS	612.45
20	NELSON	ATEND	128.20
20	LIGIA	ATEND	126.50

Exemplo de WHERE através de dado numérico

```
     SELECT      DEPTO,
NOME,
                 CARGO,
                 COMIS
     FROM        SAL_COMIS
     WHERE       DEPTO = 84
```

Exemplo de WHERE através de dado alfanumérico

```
     SELECT      DEPTO,
                 NOME,
                 CARGO,
                 COMIS
     FROM        SAL_COMIS
     WHERE       NOME = 'JOAO'
```

6.2.1.5 Operadores de comparação

Operadores	Significado
=	Igual
<> ou ¬ =	Diferente
>	Maior
>=	Maior ou Igual
<	Menor
<=	Menor ou Igual
¬ >	Não Maior
¬ <	Não Menor

Exemplos:

```
     SELECT      COD, COMIS
     FROM        SAL_COMIS
     WHERE       COMIS >= 1000

     SELECT      NOME, DEPTO, ANOS
     FROM        SAL_COMIS
     WHERE       CARGO <> 'GER'
```

6.2.1.6 Nulidade

Na inserção de uma determinada linha, uma coluna pode ser NOT NULL ou NULL.

- **NOT NULL**
  - O valor deve ser fornecido.
- **(ALLOW) NULL**
  - O valor pode ser omitido.

Observação:
Um valor NULO não é zero nem branco, ele é desconhecido.

6.2.1.7 Not Null with default

Uma coluna pode ser 'NOT NULL WITH DEFAULT'. Neste caso o sistema fornece um valor, à coluna, se omitido durante a inserção, ou seja, **zero** se coluna numérica, **brancos** se coluna definida como caractere, ou comprimento zero se coluna definida como caractere de tamanho variável.

6.2.1.8 Seleção de nulos

- Valor nulo como critério de SELECT

```
SELECT      NOME, CARGO, COMIS
FROM        SAL_COMIS
WHERE       COMIS IS NULL
```

Resultado:

NOME	CARGO	COMIS
JOAO	GER	?
PAULO	GER	?
CLEIDE	GER	?
RAFAEL	GER	?
CELSO	GER	?
CARLA	GER	?
FERNANDA	GER	?
RICARDO	GER	?
OSWALDO	GER	?
IVETE	GER	?
MARLI	GER	?

- Valor não nulo como critério de SELECT

```
SELECT      NOME, CARGO, COMIS
FROM        SAL_COMIS
WHERE       COMIS IS NOT NULL
```

Resultado:

NOME	CARGO	COMIS
JOSE	VENDAS	612.45
MESSIAS	VENDAS	846.55
CARLOS	VENDAS	650.25
MARIA	VENDAS	1152.00
NELSON	ATEND	128.20
LUIS	VENDAS	1386.70
TEREZA	ATEND	206.60
ROBERTA	ATEND	180.00
ADRIANA	ATEND	75.60
CAMILA	VENDAS	637.65
EDNA	ATEND	110.10
CASSIA	ATEND	236.50
LIGIA	ATEND	126.50
PAULA	ATEND	84.20
LETICIA	VENDAS	992.80
SERGIO	ATEND	189.65
MAURO	ATEND	513.30
SIDNEY	VENDAS	811.50
WILSON	VENDAS	806.10
HELIO	VENDAS	200.30

ELZA	VENDAS	844.00
AUREA	ATEND	55.50
MARCIA	VENDAS	1285.00
LEA	ATEND	188.00

6.2.1.9 Múltiplas condições

Os operadores utilizados são **AND** **OR**

Dadas as condições CARGO = 'VENDAS' SALARIO < 17000, temos:

- Ambas condições satisfeitas:

```
SELECT      NOME, CARGO, SALARIO
FROM        SAL_COMIS
WHERE       CARGO = 'VENDAS' AND SALARIO < 17000
```

Resultado:

NOME	CARGO	SALARIO
CARLOS	VENDAS	16808.30
MARIA	VENDAS	16502.83
WILSON	VENDAS	15454.50
ELZA	VENDAS	16858.20

- Qualquer condição satisfeita:

```
SELECT      NOME, CARGO, SALARIO
FROM        SAL_COMIS
WHERE       CARGO = 'VENDAS' OR SALARIO < 17000
```

Resultado:

NOME	CARGO	SALARIO
JOSE	VENDAS	18171.50
MESSIAS	VENDAS	18006.00
CARLOS	VENDAS	16808.30
MARIA	VENDAS	16502.83
NELSON	ATEND	13504.60
LUIS	VENDAS	18001.75
TEREZA	ATEND	12508.20
ROBERTA	ATEND	12954.75
ADRIANA	ATEND	10505.90
CAMILA	VENDAS	19456.50
EDNA	ATEND	12258.50
CASSIA	ATEND	12009.75
LIGIA	ATEND	14252.75
PAULA	ATEND	11508.60
LETICIA	VENDAS	17656.50
SERGIO	ATEND	13369.80
MAURO	ATEND	14460.00
SIDNEY	VENDAS	18674.50
MARLI	GER	19818.00
WILSON	VENDAS	15454.50
HELIO	VENDAS	21000.00

ELZA	VENDAS	16858.20
AUREA	ATEND	10988.00
MARCIA	VENDAS	17844.00
LEA	ATEND	13030.50

- Trabalhando com AND e OR:

Exemplo 01:

```
SELECT      NOME, CARGO, ANOS, SALARIO,COMIS
FROM        SAL_COMIS
WHERE       (CARGO = 'VENDAS' AND COMIS > 1200) OR ANOS > 10
```

Resultado:

NOME	CARGO	ANOS	SALARIO	COMIS
LUIS	VENDAS	6	18001.75	1386.70
OSWALDO	GER	12	21234.00	?
HELIO	VENDAS	13	21000.00	200.30
MARCIA	VENDAS	7	17844.00	1285.00

Exemplo 02:

```
SELECT      NOME, CARGO, ANOS, COMIS
FROM        SAL_COMIS
WHERE       CARGO = 'VENDAS' AND (COMIS > 1200 OR ANOS > 10)
```

Resultado:

NOME	CARGO	ANOS	COMIS
LUIS	VENDAS	6	1386.70
HELIO	VENDAS	13	200.30
MARCIA	VENDAS	7	1285.00

6.2.1.10 In

Utilizado quando o valor deve coincidir com algum elemento de uma lista. Equivale a múltiplos ORs para a mesma coluna.

Exemplo:

```
SELECT      NOME, DEPTO
FROM        SAL_COMIS
WHERE       DEPTO IN (38,20,42) ou
            DEPTO = 38 OR DEPTO = 20 OR DEPTO = 42
```

Resultado:

COD	NOME	DEPTO	CARGO	ANOS	SALARIO	COMIS
10	JOAO	20	GER	7	18357.00	?
20	JOSE	20	VENDAS	8	18181.50	612.45
30	PAULO	38	GER	5	17506.75	?
40	MESSIAS	38	VENDAS	6	18006.00	846.55
60	CARLOS	38	VENDAS	0	16808.30	650.25
80	NELSON	20	ATEND	0	13504.60	128.20
90	LUIS	42	VENDAS	6	18001.75	1386.70
100	RAFAEL	42	GER	7	18352.80	?
120	ROBERTA	38	ATEND	0	12954.75	180.00
130	ADRIANA	42	ATEND	6	10505.90	75.60
180	CASSIA	38	ATEND	3	12009.75	236.50
190	LIGIA	20	ATEND	8	14252.75	126.50
200	PAULA	42	ATEND	0	11508.60	84.20

6.2.1.11 Between

Utilizado para selecionar um intervalo fechado de valores.

Exemplo:

```
SELECT      NOME, CARGO, ANOS
```

```
FROM        SAL_COMIS
WHERE       ANOS BETWEEN 9 AND 11 ou
            Anos >= 9 and anos <= 11
```

Observação:

Neste caso a seleção será para anos = 9, 10, e 11.

Resultado:

NOME	CARGO	ANOS
CLEIDE	GER	10
FERNANDA	GER	10
IVETE	GER	9
SIDNEY	VENDAS	9
MARLI	GER	10

6.2.1.12 Pesquisas parciais (LIKE)

Utilizada para pesquisar um subconjunto de caracteres: **LIKE**

Os caracteres utilizados juntamente com o **LIKE** são:
- % Conjunto de caracteres qualquer; e;
- _ Um caractere qualquer **Exemplos:**

1. `WHERE NOME LIKE 'J%'`
   Inclui: JOAO, JOSE
2. `WHERE NOME LIKE '%NA'` Inclui: ADRIANA, EDNA
3. `WHERE NOME LIKE '%E%S%'` Inclui: MESSIAS, NELSON, CELSO
4. `WHERE NOME LIKE '_ _ _'`
   Inclui: LEA
5. `WHERE NOME LIKE '_I%'`
   Inclui: LIGIA, RICARDO, WILSON

O operador ESCAPE permite que seja utilizado outro caractere no lugar do '%'.

Exemplo:
WHERE COLUNA LIKE '100+%%' ESCAPE '+'

No exemplo acima estamos procurando strings que comecem por 100%.

O primeiro caractere % é interpretado como um caractere comum, pois é precedido pelo caractere +, que foi definido através do operador ESCAPE.

O segundo caractere % é interpretado como um conjunto de caracteres quaisquer, pois não é precedido por nenhum caractere ESCAPE.

Neste exemplo, quaisquer ocorrência do caractere + que não sejam ++ ou +% ou +_ provoca erro.

6.2.1.13 Negação

Utilizada para eliminar da pesquisa um subconjunto de caracteres

Exemplos:

1. **`WHERE NOME NOT LIKE 'J%'`** Elimina:
   JOAO, JOSE
2. **`WHERE ANOD NOT BETWEEN 9 AND 11`** Elimina ANOS DE 9 a 11

6.2.1.14 User

Registro especial, que apresenta:
- Authorization ID do tempo de execução;
- Comprimento fixo de 8; e;
- Útil para query dependente de executor.

Exemplo:

```
SELECT * SYSIBM.SYSTABLES WHERE OWNER = USER
```

6.2.1.15 Manipulando tabela resultante

Para manipulação de tabelas usamos:
- ORDER BY; e
- SELECT DISTINCT

6.2.1.16 Order By

Utilizado para classificar uma tabela em uma seqüência especifica.

Observação:
Por default a classificação é em ordem ascendente.

No exemplo abaixo, estaremos fazendo uma pesquisa na tabela SAL_COMIS para gerar um relatório com o NOME, DEPARTAMENTO, CARGO e ANOS de todos os colaboradores que estão alocados no DEPARTAMENTO 84, sendo que, este relatório deverá estar classificado (ordenado) pelo NOME dos funcionários:

```
SELECT      NOME, DEPTO, CARGO, ANOS
```

```
     FROM        SAL_COMIS
     WHERE       DEPTO = 84
ORDER BY    NOME
```

Resultado:

NOME	DEPTO	CARGO	ANOS
LEA	84	ATEND	5
MARCIA	84	VENDAS	7
MARLI	84	GER	10
WILSON	84	VENDAS	5

Para gerar um relatório com os mesmos campos, porém com uma classificação em ordem DESCENDENTE de NOME, deveríamos fazer a pesquisa conforme abaixo:

```
SELECT      NOME, DEPTO, CARGO, ANOS
FROM        SAL_COMIS
WHERE       DEPTO = 84
ORDER BY    NOME DESC
```

NOME	DEPTO	CARGO	ANOS
WILSON	84	VENDAS	5
MARLI	84	GER	10
MARCIA	84	VENDAS	7
LEA	84	ATEND	5

Podemos também efetuar uma classificação utilizando-se de campos ASCENDENTES e DESCENDENTES juntos, por exemplo, ao gerar um relatório com os mesmos campos dos relatórios acima, porém classificado em ordem ASCENDENTE de CARGO e simultaneamente em ordem DESCENDENTE de ANOS, devemos fazê-lo conforme abaixo:

```
SELECT      NOME, DEPTO, CARGO, ANOS
FROM        SAL_COMIS
WHERE       DEPTO = 84
ORDER BY    CARGO, ANOS DESC
```

Resultado:

NOME	DEPTO	CARGO	ANOS
LEA	84	ATEND	5
MARLI	84	GER	10
MARCIA	84	VENDAS	7
WILSON	84	VENDAS	5

Notas importantes:

a) Colunas classificadas devem ser referenciadas pelo SELECT;
b) Nulos são considerados como valor mais alto;
c) Podemos especificar colunas para o ORDER BY fornecendo um número que indica a posição da mesma na cláusula SELECT. O ORDER BY é o único que permite isto; e;
d) A cláusula ORDER BY classifica uma tabela resultante, logo dever ser sempre a única e a última a aparecer em uma instrução SELECT.

6.2.1.17 Select Distinct

É utilizado para eliminar linhas duplicadas. É efetuado uma classificação, automática, pelas colunas selecionadas.

Comparação entre SELECT e SELECT DISTINCT:

Uso do SELECT:

```
SELECT      DEPTO
FROM        SAL_COMIS
```

Resultado:

DEPTO
20
20
38
38
15
38
15
20
42
42
15
38
42

51
51
10
15
38
20
42
10
51
51
10
51
10
66
66
84
84
66
66
66
84
84

6.2.1.17.1 Coluna única

```
SELECT      DISTINCT DEPTO
FROM        SAL_COMIS
```

Resultado:

DEPTO
10
15
20
38
42
51
66
84

6.2.1.17.2 Múltiplas colunas

```
     SELECT      DISTINCT DEPTO, CARGO
FROM        SAL_COMIS
```

Resultado:

DEPTO	CARGO
10	GER
15	ATEND
15	GER
15	VENDAS
20	ATEND
20	GER
20	VENDAS
38	ATEND
38	GER
38	VENDAS
42	ATEND
42	GER
42	VENDAS
51	ATEND
51	GER
51	VENDAS
66	ATEND
66	GER
66	VENDAS
84	ATEND
84	GER

84	VENDAS

6.2.1.18 Order by

Utilizado para controlar a classificação

```
     SELECT      DISTINCT DEPTO, CARGO
     FROM        SAL_COMIS
ORDER BY    CARGO
```

Resultado:

DEPTO	CARGO
15	ATEND
20	ATEND
38	ATEND
42	ATEND
51	ATEND
66	ATEND
84	ATEND
10	GER
15	GER
20	GER
38	GER
42	GER
51	GER
66	GER
84	GER
15	VENDAS
20	VENDAS
38	VENDAS
38	VENDAS

42	VENDAS
51	VENDAS
66	VENDAS
84	VENDAS

6.2.1.19 Valores calculados

É efetuado através dos comandos aritméticos: + (adição), - (subtração), * (multiplicação) e / (divisão).

Gerar relatório com o ganho de cada colaborador do departamento 20.

```
SELECT      COD, SALARIO, COMIS, SALARIO + COMIS
FROM        SAL_COMIS
WHERE       DEPTO = 20
```

Resultado:

COD	SALARIO	COMIS	
10	18357.00	?	?
20	18171.50	612.45	18783.95
80	13504.60	128.20	13632.80
190	14252.75	126.50	14379.25

Observação: Quando qualquer operação envolver nulo, o resultado é nulo.

6.2.1.19.1 Condições de valores calculados

Gerar relatório de colaborador que tenha um ganho acima de R$ 20.000,00..

```
SELECT      NOME, SALARIO + COMIS
FROM        SAL_COMIS
WHERE       SALARIO + COMIS > 20000
```

Resultado:

NOME	
CAMILA	20094.15

HELIO	21200.30

6.2.1.19.2 Condições de valores calculados utilizando ORDER BY

Exemplo 01:

```
     SELECT      NOME, SALARIO + COMIS
     FROM        SAL_COMIS
     WHERE       SALARIO + COMIS > 20000
ORDER BY 2 DESC
```

Resultado:

NOME	
HELIO	21200.30
CAMILA	20094.15

Observação:

Na geração do relatório acima, podemos ver que o mesmo está classificado EM ORDEM descendente da somatório da coluna SALARIO e COMIS, ou seja, pela segunda coluna.

Exemplo 02:

```
SELECT      CARGO, SALARIO + COMIS
FROM        SAL_COMIS
ORDER BY    CARGO, 2
```

Resultado:

CARGO	
ATEND	10581.50
ATEND	10993.50
ATEND	11592.80
ATEND	12246.25
ATEND	12714.80
ATEND	13134.75
ATEND	13218.50
ATEND	13559.45
ATEND	13632.80
ATEND	14379.25
ATEND	14973.30
ATEND	22368.60
GER	?
GER	?
GER	?
GER	?
GER	?
GER	?
GER	?
GER	?
GER	?
GER	?
GER	?
VENDAS	16260.60
VENDAS	17458.55
VENDAS	17654.23
VENDAS	17702.20
VENDAS	18649.30
VENDAS	18793.95
VENDAS	18852.55
VENDAS	19129.00

VENDAS	19388.45
VENDAS	19486.00
VENDAS	20094.15
VENDAS	21200.30

6.2.1.20 Concatenação

É utilizado para juntar colunas e gerar uma string.

String1 || String2 ⟶ String1String2
'**ABC'** || '**XWZ**' ⟶ '**ABCXYZ**'

Sobrenome	Meio	Nome
␣Silva	␣da	José
␣Souza	␣de	João

Exemplo 01:

```
     SELECT      NOME || MEIO || SOBRENOME
FROM        TABELA_ACIMA
```

Resultado:

José da Silva
João de Souza

Exemplo 02:

```
     SELECT      NOME || ',' ||MEIO || '***' || SOBRENOME
FROM        TABELA_ACIMA
```

Resultado:

José, da*** Silva João,
de*** Souza

Observação:

O caractere ! ! também pode ser utilizado para a concatenação.

6.2.2 Select Avançado

Utilizado para transformar dados de uma coluna em um único valor. Para dados numéricos são utilizados: **SUM** (total de valores de uma coluna) e **AVG** (média dos valores de uma coluna); para quaisquer tipo de dados são utilizados: **MIN** (menor

valor de uma coluna), **MAX** (maior valor de uma coluna) e **COUNT** (número de ocorrências).

Observações:

- Informações detalhadas não podem ser mostradas, por exemplo, se for solicitada a média salarial por departamento, os valores da coluna NOME não podem ser mostrados;e;
- Nulos são ignorados.

6.2.2.1 SUM / AVG / MIN / MAX

Geram um único valor a partir de um conjunto de valores de uma única coluna. O resultado não contém detalhes de linhas individuais. Podem ser pedidas mais de uma função para a mesma coluna. Nulos são excluídos. Precisão (parte inteira, casa decimal) deriva da coluna.

Para a função **AVG** de uma coluna decimal (p,s) o resultado terá precisão (15,15p+s); expressões deste tipo podem ser argumento de uma column function (função de coluna) , por exemplo, **AVG (SALARIO + COMIS)**.

Exemplo:

```
SELECT   SUM(SALARIO),
         AVG(SALARIO),
         MIN(COMIS),
         MAX(COMIS)
FROM     SAL_COMIS
WHERE    DEPTO = 66
```

Resultado:

| --- | --- | --- | --- |
| 86076.20 | 17215.240000000 | 55.50 | 844.00 |

6.2.2.2 COUNT

Utilizado para efetuar uma contagem de linhas. Pode ser usado das seguintes maneiras:

- **COUNT(*)** – Número de linhas que satisfaçam a condição WHERE;
- **COUNT(DISTINCT nome da coluna)** – Número de valores distintos na coluna.

NULOS não são contados **Exemplo:**

Gerar um relatório com:
1. Total de colaboradores com salários acima de R$ 18.000,00; 2.
Total de seus respectivos departamentos; e;
3. Mostrar média salarial.

```
SELECT  COUNT(DISTINCT DEPTO),
        COUNT(*),
        AVG(SALARIO),
        MIN(COMIS),
FROM    SAL_COMIS
WHERE   SALARIO > 18000
```

Resultado:

----------	----------	----------
8	16	19604.206250000

6.2.2.3 Literais

Podem ser mostradas no resultado do SELECT.

São particularmente úteis quando usadas com funções de coluna que não exibem cabeçalhos.

Podemos selecionar apenas uma cadeia de caracteres, por exemplo, SELECT 'TESTE FUTURE' FROM TABELA_CURSOS.

Observação:

É necessário que a tabela exista.

Exemplo sem Literais

```
SELECT  AVG(COMIS),
        SUM(COMIS),
FROM    SAL_COMIS
```

Resultado:

----------	----------
513,31	12319,45

Exemplo com Literais

```
SELECT  'MEDIA: ',AVG(COMIS),
        'SOMA : ',SUM(COMIS),
FROM    SAL_COMIS
```

Resultado:

----------	----------

MEDIA: 513,31	SOMA: 12319,45

6.2.2.4 Group By

A função desta cláusula é mostrar informações calculadas por grupo de dados, sendo que o resultado é classificado, isto porque o GRUPO BY classifica a tabela em numa fase intermediária para facilitar o cálculo para os grupos.

A classificação se dá em ordem ascendente, amenos que exista um índice definido para a coluna, neste caso a classificação não é efetuada.

A seqüência obedecerá à seqüência do índice que pode estar definida como sendo ordem descendente.

Com o GROUP BY, a column function, calcula um valor para cada agrupamento.

Exemplo 01:

```
SELECT    SUM(SALARIO),
          SUM(COMIS),
FROM      SAL_COMIS
WHERE     CARGO <> 'GER'
GROUP BY DEPTO
```

Resultado:

----------	----------
41269.53	1468.70
45928.85	867.15
59778.80	1913.30
40016.25	1546.50
64942.80	2333.40
67520.70	1911.30
46329.00	2279.10

Exemplo 02:

```
SELECT    DEPTO,
          SUM(SALARIO),
          SUM(COMIS),
FROM      SAL_COMIS
```

```
WHERE    CARGO <> 'GER'
GROUP BY DEPTO
```

Resultado:

DEPTO	----------	----------
15	41269.53	1468.70
20	45928.85	867.15
38	59778.80	1913.30
42	40016.25	1546.50
51	64942.80	2333.40
66	67520.70	1911.30
84	46329.00	2279.10

Observação:

Note que a coluna selecionada, no caso DEPTO, não pode ser diferente da coluna que está na cláusula GROUP BY.

6.2.2.5 Group By ... Order By

Podemos fazer GROUP BY sobre múltiplas coluna. Isto faz com que sejam formados subgrupos dentro de grupos.|

Colunas 'GROUP BY' não precisam ser referenciadas pelo SELECT, mas colunas referenciadas pelo SELECT que não sejam função precisam ser agrupadas pelo 'GROUP BY'. Isto é um erro muito freqüente.

O resultado contém uma linha sumário para cada grupo.

Todos os valores nulos são considerados como um grupo.

Podemos alterar a ordem de classificação conforme abaixo:

Exemplo:

```
     SELECT    DEPTO,
               SUM(SALARIO),
SUM(COMIS),
     FROM      SAL_COMIS
```

Resultado:

```
WHERE    CARGO <> 'GER'
GROUP BY DEPTO
ORDER BY 3
```

DEPTO	----------	----------
20	45928.85	867.15
15	41269.53	1468.70
42	40016.25	1546.50
66	67520.70	1911.30
38	59778.80	1913.30
84	46329.00	2279.10
51	64942.80	2333.40

6.2.2.6 Group By ... Having

A função desta cláusula é eliminar alguns grupos pela cláusula GROUP BY.

Exemplo 01:

```
SELECT   DEPTO,
         SUM(SALARIO)
FROM     SAL_COMIS
WHERE    CARGO <> 'GER'
GROUP BY DEPTO
```

Resultado:

DEPTO	----------
10	83463.45
15	61929.33
20	64286.35
38	77285.55

Resultado:

42	58369.05
51	86092.80
66	86076.20
84	66147.00

Exemplo 02

```
SELECT    DEPTO,
          SUM(SALARIO)
FROM      SAL_COMIS
GROUP BY  DEPTO
HAVING    SUM(SALARIO) > 65000
```

DEPTO	----------
10	83463.45
38	77285.55
51	86092.80
66	86076.20
84	66147.00

Observações:

As linhas que apresentão a somatória de salários acima de R$ 65.000,00 não foram demonstradas.

Geração de relatórios:

1. Gerar relatório com média salarial por departamento. Considerar apenas os cargos diferentes de GERENTE dos departamentos com mais de 3 pessoas.

```
SELECT    DEPTO,
          AVG(SALARIO)
FROM      SAL_COMIS
WHERE     CARGO <> 'GER'
GROUP BY  DEPTO
```

Resultado:

```
HAVING    COUNT(*) > 3
```

Resultado:

DEPTO	----------
38	14944.70
51	16235.70
66	16880.18

2. Gerar relatório com média salarial por departamento em ordem decrescente de média. Considerar apenas os cargos diferentes de GERENTE dos departamentos compostos por pessoas com experiência mínima de 5 anos.

```
SELECT    DEPTO,
          AVG(SALARIO)
FROM      SAL_COMIS
WHERE     CARGO <> 'GER'
GROUP BY DEPTO
HAVING    MIN(ANOS) >= 5
ORDER BY 2 DESC
```

DEPTO	----------
66	19837.25
51	17191.00
20	16212.13
84	15443.00
38	15007.88
15	14505.52
42	14253.83

Observações:

1. O relatório nº 1 mostra que o SELECT e o HAVING podem usar funções diferentes;

Resultado:

2. O relatório nº 2 ilustra que a tabela resultante pode ser classificada.

6.2.2.7 Sumário (funções de coluna)

Funções de coluna (*column function*) **só** podem ser especificados em:

- SELECT – pode especificar **somente** funções de coluna e colunas especificadas no GROUP BY;
- HAVING – pode especificar funções de coluna sobre qualquer coluna contida em uma tabela especificada na cláusula FROM.

Funções de coluna **não** podem ser embutidas uma dentro da outra.

6.2.2.8 Funções escalares

Funções escalares (*scalar function*) transformam um único valor em outro através das seguintes funções: STRING, CONVERSÃO e DATE /TIME.

Ao contrário das funções de coluna que transformam um conjunto de valores de uma coluna em um único valor, as funções escalares operam somente sobre um único valor.

Elas podem estar embutidas uma dentro da outra, por exemplo,
SUBSTR(SUBSTR(...)...).

As funções de coluna podem ser argumentos de uma função escalar, por exemplo, SUBSTR(MAX(...)...).

As funções escalares podem ser argumentos de uma função de coluna, por exemplo, MAX(SUBTR(...)...).

6.2.2.9 Substr(string, inicio, comprimento)

Se o comprimento não for especificado, a função será aplicada para todo o resto.

Se a posição do início ou o início + comprimento ultrapassar o final do string, teremos um erro.

Substring de nulo é nulo.

Exemplo 01:

```
SELECT    DEPTNOME,
          SUBSTRING(DEPTNOME,1,4)
FROM      COD_GEREN
```

Resultado:

DEPTNOME	----------
MATRIZ	MATR
MATRIZ	MATR
MATRIZ	MATR
MATRIZ	MATR
CAMPINAS	CAMP
RIO	RIO
MINAS	MINA
BAHIA	BAHI
RECIFE	RECI
RIO GRANDE	RIO
PARANA	PARA

Exemplo 02:

```
SELECT    DEPTCOD,
          DIVISAO
FROM      COD_GEREN
WHERE     SUBSTR(DIVISAO,4)='ESTE'
```

Resultado:

DEPTCOD	DIVISAO
15	SUDESTE
20	SUDESTE
38	SUDESTE

6.2.2.10 Length (argumento)

Fornece o tamanho de uma coluna do tipo caractere. Se o argumento é nulo, o resultado também será.

Exemplo :

```
SELECT   NOME,                    LENGHT(NOME),    Tamanho variável
         CARGO, LENGHT(CARGO) Tamanho fixo      FROM SAL_COMIS
```

Resultado :

NOME	------	CARGO	------
JOAO	4	GER	6
JOSE	4	VENDAS	6
PAULO	5	GER	6
MESSIAS	7	VENDAS	6
CLEIDE	6	GER	6
CARLOS	6	VENDAS	6
MARIA	5	VENDAS	6
NELSON	6	ATEND	6
LUIS	4	VENDAS	6
RAFAEL	6	GER	6
TEREZA	6	ATEND	6
ROBERTA	7	ATEND	6
ADRIANA	7	ATEND	6
CELSO	5	GER	6
CAMILA	6	VENDAS	6
CARLA	5	GER	6
EDNA	4	ATEND	6
CASSIA	6	ATEND	6

NOME	------	CARGO	------
LIGIA	5	ATEND	6
PAULA	5	ATEND	6
FERNANDA	8	GER	6
LETICIA	7	VENDAS	6
SERGIO	6	ATEND	6
RICARDO	7	GER	6
MAURO	5	ATEND	6
OSWALDO	7	GER	6
IVETE	5	GER	6
SIDNEY	6	VENDAS	6
MARLI	5	GER	6
WILSON	6	VENDAS	6
HELIO	5	VENDAS	6
ELZA	4	VENDAS	6
AUREA	5	ATEND	6
MARCIA	6	VENDAS	6
LEA	3	ATEND	6
			6

6.2.2.11 Value (arg1, arg2, ..., argn)

Dada uma lista de argumentos, esta função faz a varredura da esquerda para a direita até encontrar o primeiro valor nulo.

Todos os argumentos devem ser data types compatíveis (to numéricos ou todos caracteres).

Se todos os argumentos forem nulos, o resultado também será.

Exemplo :

```
     SELECT    COD,
               COMIS,
               VALUE (COMIS, 0)
FROM      SAL_COMIS
```

Resultado:

COD	COMIS	-----
10	?	0.00
20	612.45	612.45
30	?	0.00
40	846.55	846.55
50	?	0.00
60	650.25	650.25
70	1152.00	1152.00
80	128.20	128.20
90	1386.70	1386.70
100	?	0.00
110	206.60	206.60
120	180.00	180.00
130	75.60	75.60
140	?	0.00
150	637.65	637.65
160	?	0.00
170	110.10	110.10
180	236.50	236.50

COD	COMIS	-----
190	126.50	126.50
200	84.20	84.20
210	?	0.00
220	992.80	992.80
230	189.65	189.65
240	?	0.00
250	513.30	513.30
260	?	0.00
270	?	0.00
280	811.50	811.50
290	?	0.00
300	806.10	806.10
310	200.30	200.30
320	844.00	844.00
330	55.50	55.50
340	1285.00	1285.00
350	188.00	188.00

Observação:

Quando a coluna COMIS apresentou valor nulo, houve uma substituição por 0,00

6.2.2.12 Funções de conversão

Utilizada para converter um tipo de representação em outro tipo.

São utilizadas as seguintes funções:

- **DECIMAL, FLOAT, INTEGER** – Converte dados numéricos (número – número);
- **DIGITS** – Converte dados numéricos em alfanuméricos (número – alfa);
- **HEX** – converte hexadecimal em alfanumérico (hexa – alfa). Cada dígito hexadecimal é representado por 2 caracteres da string resultante.

Exemplo :

Gerar um relatório onde SALARIO, que é uma coluna que apresenta a seguinte definição (DECIMAL (18, 2) NOT NULL), seja convertido para DECIMAL (15, 5), INTEGER e DIGITS.

```
SELECT   SALARIO
         DECIMAL(SALARIO, 18, 5),
         INTEGER(SALARIO),
         DIGITS(SALARIO)
FROM   SAL_COMIS
WHERE  COD = 10
```

Resultado:

SALARIO	----------	----------	----------
18357.50	18357.50000	18357	000000000001835750

6.2.2.13 Date / Time

6.2.2.13.1 Dados

1. DATA, TIME, TIMESTAMP armazenados como decimal compactado sem sinal

Data Type	Formato Interno
Date	aaaammdd
Time	hhmmss
Timestamp	aaammddhhmmssnnnnnn

2. Programas lidam só com **formato externo**: string de caracteres

Formato	Formato time	Tamanho	Formato Date	Tamanho
ISO (Standard)	hh.mm.ss	8 bytes	aaaa-mm-dd	10 bytes
USA (americano)	hh:mm AM hh:mm PM	8 bytes	mm/dd/aaaa	10 bytes
EUR (europeu)	hh.mm.ss	8 bytes	dd.mm.aaaa	10 bytes
JIS (japonês)	hh.mm.ss	8 bytes	aaaa-mm-dd	10 bytes

Observação:

O dado **timestamp** é composto de 26 bytes: **aaaa-mm-dd-hh.mm.ss.nnnnnn**

6.2.2.13.2 Aritmética

Somente adição e subtração. Pode usar MIN, MAX e COUNT. Não pode usar SUM e AVG.

6.2.2.13.3 Duração

- **Simples**

Horário ou – ⟶ Horário duração em hhmmss (packed decimal (6.0))
Data ou –Data ⟶ duração em aaaammdd (packed decimal (8.0))

- **Rotulada**

Número “n” seguido das palavras chaves (YEARS, MONTHS, DAYS, HOURS, MINUTES, SECONDS, MICROSECONDS)

- **Somando / Subtraindo duração**

1. Data + ou – duração (data) ⟶ **Resulta em data** DATAFATU + 3 MONTHS
2. Horário + ou – duração(tempo) ⟶ **Resulta em horário** TEMPARQ – (2 HOURS – 35 MINUTES)
3. Timestamp + ou – duração ⟶ (data) **Resulta em timestamp** TIMESTAMP + 3 MONTHS
4. Timestamp + ou – duração ⟶ (tempo) **Resulta em timestamp** TIMESTAMP – 60000 MICROSECONDS

- **Somando datas e tempos**

1. Data – data ⟶ **Resulta em duração**

'2006-12-31' – DATAFATU DATAHOJE
– 2005-11-01

2. Horário – horário ⟶ **Resulta em duração**

'23:59:59' – '02:25:58'
HORATUAL – "01:10:00" **Exemplos:**

Tabela **DATA_HORARIO**

COD_PROJ	DATA_INICIO	DATA_FIM	HORA_INICIO	HORA_FIM
FUTURE01	2003.01.31	2003.10.31	08.00.00	17.00.00
FUTURE02	2004.10.01	2004.12.10	08.00.00	22.00.00
FUTURE03	2005.02.01	2005.12.01	17.00.00	22.00.00

```
SELECT   COD_PROJ,
         DATA_INICIO + 1 MONTH,
         DATA_FIM – 10 DAYS
FROM   DATA_HORARIO
```

Resultado:

COD_PROJ	----------	----------
FUTURE01	2003.02.31	2003.10.21
FUTURE02	2004.11.01	2004.11.30
FUTURE03	2005.03.01	2005.11.22

```
SELECT   COD_PROJ,
         HORA_INICIO + 10 SECONDS,
         DATA_FIM – 10 MINUTES
FROM   DATA_HORARIO
```

Resultado:

COD_PROJ	----------	----------
FUTURE01	08.00.10	17.10.00
FUTURE02	08.00.10	22.10.00
FUTURE03	17.00.10	22.10.00

6.2.2.13.4 Funções escalares

- **CHAR** – controla formato externo de dados

```
SELECT CHAR(HORA_FIM,ISO),
       CHAR(HORA_FIM,USA)
FROM   DATA_HORARIO
WHERE  COD_PROJ = 'FUTURE01'
```

Resultado:

----------	----------
17.00.00	05.00 PM

- **Day, Month, Year, Hour, Minute, Second, Microsecond**
  Extrai parte de uma data, horário ou timestamp. Resulta em um inteiro binário.

```
SELECT DAY(DATA_INICIO),
       MONTH(DATA_INICIO),
       YEAR(DATA_INICIO)
FROM   DATA_HORARIO
WHERE  YEAR(DATA_INICIO) > 2004
```

Resultado:

----------	----------	----------
1	2	2005

- **Days –** número de dias desde 01/01/1900

```
SELECT DATA_FIM - DATA_INICIO,
       DAYS(DATA_FIM) - DAYS(DATA_INICIO),
FROM   DATA_HORARIO
WHERE  COD_PROJ = 'FUTURE01'
```

Resultado:

----------	----------

900	273

- **Date / Time –** extrai data ou horário de um timestamp

6.2.2.13.5 Valores concorrentes

- **Current date** – Data corrente (dia)
- **Current Time** – Horário corrente
- **Current Timestamp** – Data e horário corrente côn vertido para timestamp

```
SELECT *
FROM   DATA_HORARIO
WHERE  DATA_INICIO > CURRENT DATE - 340 DAYS
```

Resultado:

COD_PROJ	DATA_INICIO	DATA_FIM	HORA_INICIO	HORA_FIM
FUTURE03	2005.02.01	2005.12.01	17.00.00	22.00.00

6.2.2.14 Join de tabelas

6.2.2.14.1 Duas Tabelas

Esta técnica serve para obtermos informações que estão espalhadas em uma ou mais tabelas.

As tabelas são relacionadas via dados comuns.

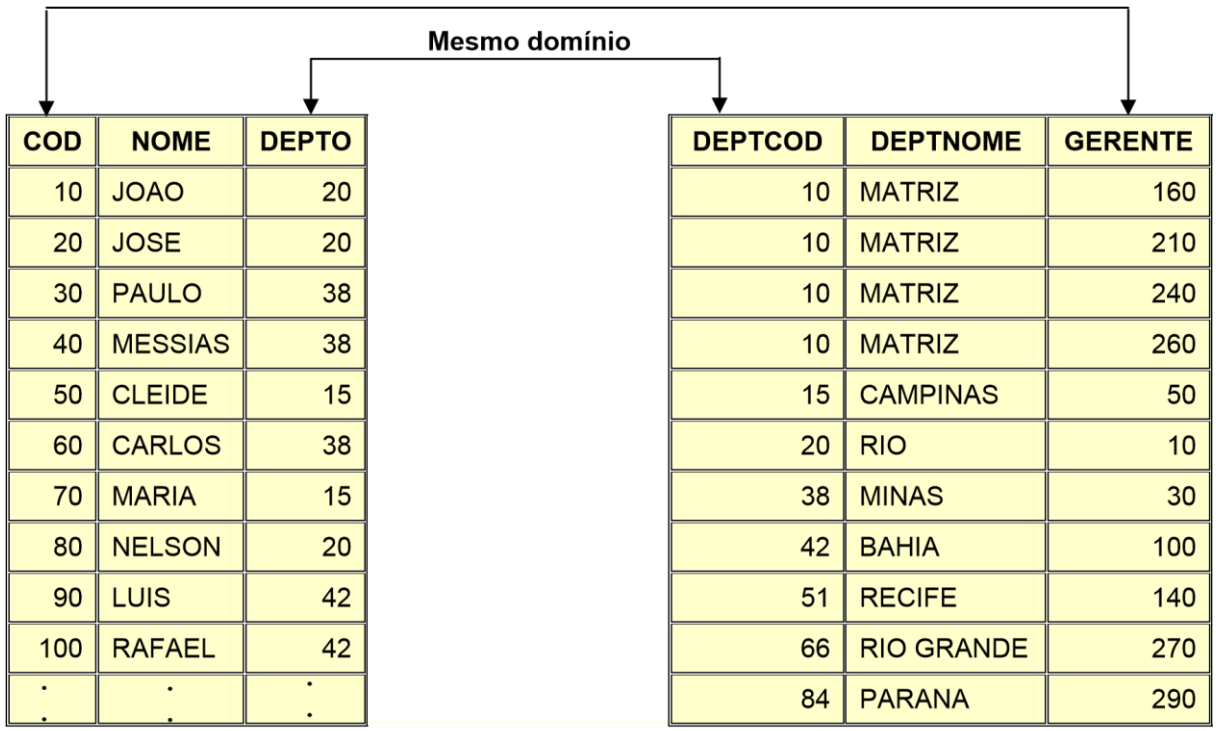
O relacionamento pode ser efetuado através de qualificadores correlacionados, onde atribuímos um rótulo à tabela.

O join é implementado pela clausula **FROM**.

O resultado é um subconjunto do produto cartesiano das tabelas.

Para capturar linhas que não possuam correspondência, pode se utilizar o chamado *OUTER JOIN*.

Ambiente JOIN:



COD	NOME	DEPTO
10	JOAO	20
20	JOSE	20
30	PAULO	38
40	MESSIAS	38
50	CLEIDE	15
60	CARLOS	38
70	MARIA	15
80	NELSON	20
90	LUIS	42
100	RAFAEL	42
.	.	.
.	.	.

SAL_COMIS

DEPTCOD	DEPTNOME	GERENTE
10	MATRIZ	160
10	MATRIZ	210
10	MATRIZ	240
10	MATRIZ	260
15	CAMPINAS	50
20	RIO	10
38	MINAS	30
42	BAHIA	100
51	RECIFE	140
66	RIO GRANDE	270
84	PARANA	290

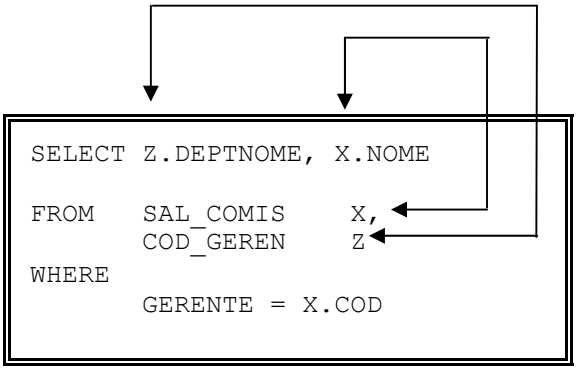
COD_DEPTO

Exemplo 01:

Neste exemplo mostraremos a utilização dos **qualificadores**.

- Gerar um relatório de nome dos departamentos com seus respectivos gerentes

Qualificador Correlacionado



```
SELECT Z.DEPTNOME, X.NOME

FROM   SAL_COMIS     X,
       COD_GEREN     Z
WHERE
       GERENTE = X.COD
```

ou

Qualificador com nome da tabela

```
SELECT DEPTNOME,
       NOME,
FROM   SAL_COMIS,
       COD_GEREN
WHERE
       GERENTE = COD
```

Resultado:

DEPTNOME	NOME
MATRIZ	CARLA
MATRIZ	FERNANDA
MATRIZ	RICARDO
MATRIZ	OSWALDO
CAMPINAS	CLEIDE
RIO	JOAO
MINAS	PAULO
BAHIA	RAFAEL
RECIFE	CELSO
RIO GRANDE	IVETE
PARANA	MARLI

Exemplo 02:

Gerar relatório de nome dos departamentos da divisão sudeste com seus respectivos gerentes:

```
SELECT Z.DEPTNOME,
       X.NOME,
FROM   SAL_COMIS     X,
       COD_GEREN     Z
WHERE
       DIVISAO = 'SUDESTE' AND
       GERENTE = X.COD
```

ou

```
SELECT DEPTNOME,
       NOME,
FROM   SAL_COMIS,
       COD_GEREN
WHERE
       DIVISAO = 'SUDESTE' AND
       GERENTE = COD
```

Resultado:

DEPTNOME	NOME
CAMPINAS	CLEIDE
RIO	JOAO
MINAS	PAULO

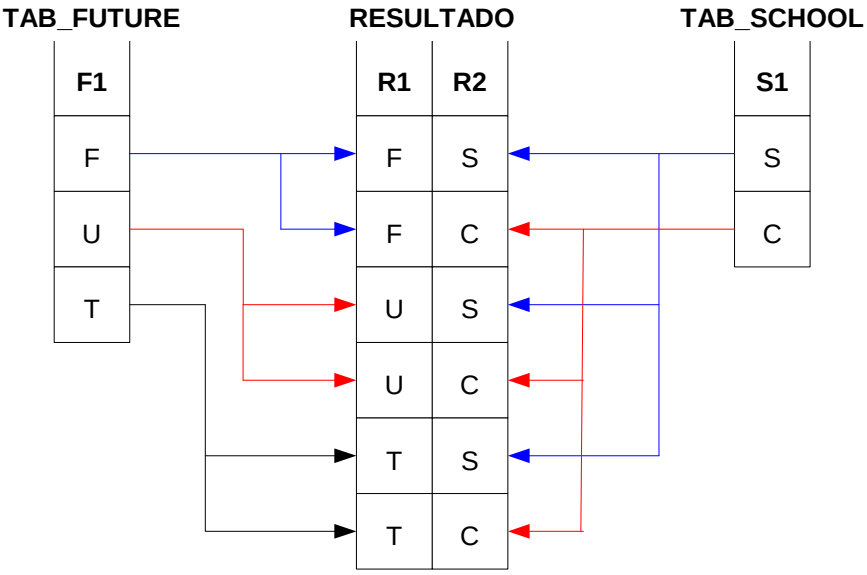
Importante:

Procure não fazer JOIN de tabelas sem a cláusula WHERE.

Funcionamento:

```
SELECT F1,
       S1
FROM   TAB_FUTURE,
       TAB_SCHOOL
```

Produto cartesiano



6.2.2.14.2 Mais de duas tabelas

Exemplo 01:

```
SELECT A.COD,
       A.NOME,
       A.DEPTO,
       B.DEPTNOME,
       C.COD_PEDIDO
FROM   SAL_COMIS    A,
       COD_GEREN    B,
       TAB-PEDIDOS  C
WHERE
       A.DEPTO = B.DEPTCOD AND
       A.COD   = C.COD_VENDA
```

ou

```
SELECT COD,
       NOME,
       DEPTO,
       DEPTNOME,
       COD_PEDIDO
FROM   SAL_COMIS,
       COD_GEREN,
       TAB-PEDIDOS
WHERE
       DEPTO = DEPTCOD AND
       COD   = COD_VENDA
```

Resultado:

COD	NOME	DEPTO	DEPTNONE	COD_PEDIDO
10	JOAO	20	RIO	1234
20	JOSE	20	RIO	3456
20	JOSE	20	RIO	6667
20	JOSE	20	RIO	3580
20	JOSE	20	RIO	7010
30	PAULO	38	MINAS	1478
40	MESSIAS	38	MINAS	5687
40	MESSIAS	38	MINAS	6597
40	MESSIAS	38	MINAS	6540
70	MARIA	15	CAMPINAS	9875
80	NELSON	20	RIO	9963
90	LUIS	42	BAHIA	3252
100	RAFAEL	42	BAHIA	3654
110	TEREZA	15	CAMPINAS	1548
120	ROBERTA	38	MINAS	9854
130	ADRIANA	42	BAHIA	3654

140	CELSO	51	RECIFE	4664
150	CAMILA	51	RECIFE	4466
160	CARLA	10	MATRIZ	4335
170	EDNA	15	CAMPINAS	5589
180	CASSIA	38	MINAS	1269
190	LIGIA	20	RIO	6932
COD	**NOME**	**DEPTO**	**DEPTNONE**	**COD_PEDIDO**
200	PAULA	42	BAHIA	3265
210	FERNANDA	10	MATRIZ	5459
220	LETICIA	51	RECIFE	9597
230	SERGIO	51	RECIFE	9893
240	RICARDO	10	MATRIZ	3536
250	MAURO	51	RECIFE	2021
260	OSWALDO	10	MATRIZ	2326
270	IVETE	66	RIO GRANDE	6861
280	SIDNEY	66	RIO GRANDE	3637
290	MARLI	84	PARANA	3839
300	WILSON	84	PARANA	4041
300	WILSON	84	PARANA	4041
310	HELIO	66	RIO GRANDE	4243
320	ELZA	66	RIO GRANDE	4445
330	AUREA	66	RIO GRANDE	4658
340	MARCIA	84	PARANA	6580
350	LEA	84	PARANA	2568

6.2.2.14.3 Uma tabela com ela mesma

Tabela de cargos e salários = **CARGO_SAL**

REG_COLAB	NOME	CARGO	SALARIO	GERENCIA
010	JORGE	PRE	20000.00	000
020	HUMBERTO	GER	15000.00	010
030	DANIELA	VENDAS	10000.00	020
040	ANA	VENDAS	12000.00	020
050	VITOR	VENDAS	15500.00	020
060	VINICIUS	VENDAS	15800.00	020
100	PAULO	GER	18000.00	010
110	MARIANA	VENDAS	15000.00	100

Gerar relatório de colaboradores que ganham mais que seus respectivos gerentes:

```
SELECT C.NOME,
       C.SALARIO,
       G.NOME,
       G.SALARIO
FROM   CARGO_SAL   C
       CARGO_SAL   G
WHERE  C.GERENCIA = G.REG_COLAB    AND
       C.SALARIO  > G.SALARIO
```

Resultado:

NOME	SALARIO	NOME	SALARIO
VITOR	15500.00	HUMBERTO	15000.00
VINICIUS	15800.00	HUMBERTO	15000.00

Importante:

O relatório acima é clássico e poderia ser resolvido pela técnica de subquery, que será vista mais à frente.

6.2.2.15 Union

Esta técnica efetua a união de uma ou mais tabelas resultado. As querys complexas podem ser feitas usando o UNION (são executadas serialmente). A quantidade de

colunas deve ser a mesma em todos os SELECTs. Os data types das colunas correspondentes devem ser iguais. O tamanho das colunas correspondentes podem ser diferentes. O operador UNION une as tabelas resultantes e elimina as linhas duplicadas, que implica em um SORT (classificação). O operador UNION ALL também une as tabelas mas não elimina linhas duplicadas, portanto, não chama o SORT.

Para os nomes das colunas no cabeçalho são adotadas as do primeiro bloco de SELECT da query.

Exemplo:

Gerar relatório com os valores de salários acrescidos em 17% e 2% nos cargos de atendentes e vendas, respectivamente:

```
SELECT CARGO,
       SALARIO * 1.17
FROM   SAL_COMIS
WHERE  CARGO = 'ATEND'
UNION
    SELECT CARGO,
           SALARIO * 1.02
    FROM   SAL_COMIS
    WHERE  CARGO = 'VENDAS'
ORDER  BY  2  DESC
```

Resultado:

CARGO		CARGO	
VENDAS	21420,00	ATEND	16675,72
VENDAS	19845,63	ATEND	15800,38
VENDAS	19047,99	VENDAS	15763,59
VENDAS	18534,93	ATEND	15642,67
VENDAS	18366,12	ATEND	15245,69
VENDAS	18361,79	ATEND	15157,06
VENDAS	18200,88	ATEND	14634,59
VENDAS	18009,63	ATEND	14342,45
VENDAS	17195,36	ATEND	14051,41

VENDAS	17144,47	ATEND	13465,06
ATEND	16918,20	ATEND	12855,96
VENDAS	16832,89	ATEND	12291,90

6.2.2.15.1 Union ALL

O UNION elimina as linhas redundantes ao contrário do UNIO ALL. O NOME da coluna da tabela resultante é obtida da primeira tabela referenciada na instrução SELECT.

Exemplo:

Gerar relatório com os valores de salários acima de R$ 20.000,00 e abaixo de R$ 12.000,00:

```
SELECT COD,
       SALARIO
FROM   SAL_COMIS
WHERE  SALARIO > 20000.00
UNION ALL
    SELECT COD,
           SALARIO
    FROM   SAL_COMIS
    WHERE  SALARIO < 12000.00
ORDER  BY  SALARIO DESC
```

Resultado:

COD	SALARIO
160	22959.20
260	21234.00
140	21150.00
310	21000.00
50	20659.80
210	20010.00
200	11508.60

330	10988.00
130	10505.90

6.2.2.16 Subquery

É um resultado embutido dentro de um SELECT. O resultado de uma SUBQUERY é usado pelo SELECT "***externo***".

Pode ser usado no WHERE ou no HAVING ou ainda ser encadeado.

A escolha do operador do SELECT externo depende da quantidade de linhas do subquery.

Regras para utilização:

- Deve estar à direita do operador na condição de seleção;
- Deve ser colocado entre parênteses;
- O subquery deve selecionar apenas uma coluna;
- O subquery pode retornar uma ou mais linhas e isto acaba determinando o operador a ser usado na search condition;
- **Não** pode conter **UNION, UNION ALL** ou **ORDER BY**;
- Muitas vezes o SUBQUERY é citado como SUBSELECT. Prefira o termo SUBQUERY.

6.2.2.16.1 Uma linha

Exemplo 01:

Gerar relatório dos colaboradores com salário superior à média da consultoria:

```
     SELECT NOME,
            SALARIO
     FROM   SAL_COMIS
     WHERE  SALARIO > (SELECT AVG(SALARIO)
FROM   SAL_COMIS)
```

Resultado:

NOME	SALARIO	NOME	SALARIO
JOAO	18357.50	FERNANDA	20010.00
JOSE	18171.50	LETICIA	17656.50
PAULO	17506.75	RICARDO	19260.25
MESSIAS	18006.00	OSWALDO	21234.00
CLEIDE	20659.80	IVETE	18555.50
CARLOS	16808.30	SIDNEY	18674.50

Rua Dona Primitiva Via asc
Fone: (0X ww @bn.com.br
82

LUIS	18001.75	MARLI	19818.00
RAFAEL	18352.80	HELIO	21000.00
CELSO	21150.00	ELZA	16858.20
CAMILA	19456.50	MARCIA	17844.00
CARLA	22959.20		

Exemplo 02:

Gerar relatório do colaboradores com maior salário pago pela consultoria:

```
SELECT NOME,
       SALARIO
FROM   SAL_COMIS
WHERE  SALARIO >
       (SELECT MAX(SALARIO)
FROM   SAL_COMIS)
```

Resultado:

NOME	SALARIO
CARLA	22959.20

Observação:

Quando o SELECT interno retorna mais que uma linha, passa a ser necessário o uso de um dos seguintes operadores: **ALL**, **ANY**, **SOME** ou **IN**.

6.2.2.16.2 Várias linhas (ALL)

Exemplo:

Gerar relatório classificado dos colaboradores com salário superior a toda e qualquer média salarial departamental:

```
SELECT NOME,
       SALARIO
FROM   SAL_COMIS
WHERE  SALARIO > ALL
       (SELECT AVG(SALARIO)
        FROM   SAL_COMIS
        GROUP  BY DEPTO)
ORDER BY NOME
```

Resultado:

NOME	SALARIO
CELSO	21150.00
CARLA	22959.20
HELIO	21000.00
OSWALDO	21234.00

6.2.2.16.3 Várias linhas (ANY ou SOME)

Exemplo:

Gerar relatório classificado dos colaboradores com salário maior que a média salarial de algum departamento:

```
SELECT NOME,
       SALARIO
FROM   SAL_COMIS
WHERE  SALARIO > ANY
       (SELECT AVG(SALARIO)
        FROM   SAL_COMIS
        GROUP  BY DEPTO)
ORDER BY NOME
```

Resultado:

NOME	SALARIO
CAMILA	19456,50
CARLA	22959,20
CARLOS	16808,30
CELSO	21150,00
CLEIDE	20659,80
ELZA	16858,20
FERNANDA	20010,00
HELIO	21000,00
IVETE	18555,50
JOAO	18357,50
JOSE	18171,50
LETICIA	17656,50

NOME	SALARIO
LUIS	18001,75
MARCIA	17844,00
MARIA	16502,83
MARLI	19818,00
MESSIAS	18006,00
OSWALDO	21234,00
PAULO	17506,75
RAFAEL	18352,80
RICARDO	19260,25
SIDNEY	18674,50
WILSON	15454,50

6.2.2.16.4 Várias linhas (IN)

Subquery que resulta em uma lista de solicitações; equivale a uma série de 'OR...' sobre uma coluna.

Exemplo:

Gerar relatório dos gerentes da divisão sul:

```
SELECT DEPTO,
       NOME
FROM   SAL_COMIS
WHERE  COD IN
       (SELECT GERENTE
        FROM  COD_GERENT
        WHERE DIVISAO = 'SUL')
```

Resultado:

DEPTO	NOME
66	IVETE
84	MARLI

6.2.2.16.5 Having

Exemplo:

Gerar relatório dos departamentos cuja média salarial seja inferior à média das companhia, ignorando os gerentes, mostrando as médias, classificado pelas médias em ordem decrescente:

```
SELECT DEPTO,
       AVG(SALARIO
FROM   SAL_COMIS
WHERE  CARGO <> 'GER'
GROUP  BY DEPTO HAVING AVG(SALARIO) <
       (SELECT AVG(SALARIO)
        FROM  SAL_COMIS
        WHERE CARGO <> 'GER)
ORDER  BY 2 DESC
```

Resultado:

DEPTO	
66	16880.175000
51	16235.700000
84	15443.000000
20	15309.616667
38	14944.700000
15	13756.510000
42	13338.750000

6.2.2.16.6 Correlacionada

Uma subquery correlacionada é ***executado a cada linha devolvida*** ao SELECT externo.

Importante:

Evite o uso da subquery correlacionada por questões de performance.

Exemplo:

Listar os colaboradores com salários superiores à média de seus respectivos departamentos:

```
SELECT NOME,
       SALARIO
FROM   SAL_COMIS A
WHERE  SALARIO >
       (SELECT AVG(SALARIO)
        FROM   SAL_COMIS
        WHERE  DEPTO = A.DEPTO)
```

Resultado:

NOME	SALARIO
JOAO	18357.00
JOSE	18171.50
PAULO	17506.75
MESSIAS	18006.00
CLEIDE	20659.80
CARLOS	16808.30
MARIA	16502.83
LUIS	18001.75
RAFAEL	18352.80
CELSO	21150.00
CARLA	22959.20
OSWALDO	21234.00
MARLI	19818.00
HELIO	21000.00
MARCIA	17844.00

6.2.2.16.7 Teste (Verdadeiro ou Falso)

É um operador que tem como argumento um SELECT. Esta subquery não devolve nenhuma tabela resultante.

É utilizada a cláusula EXIST, que devolve V ou F, e como faz parte de uma search condition, determina uma eventual execução ou não do SEELECT externo.

A cláusula EXIST pode ser negada por um NOT.

Exemplo:

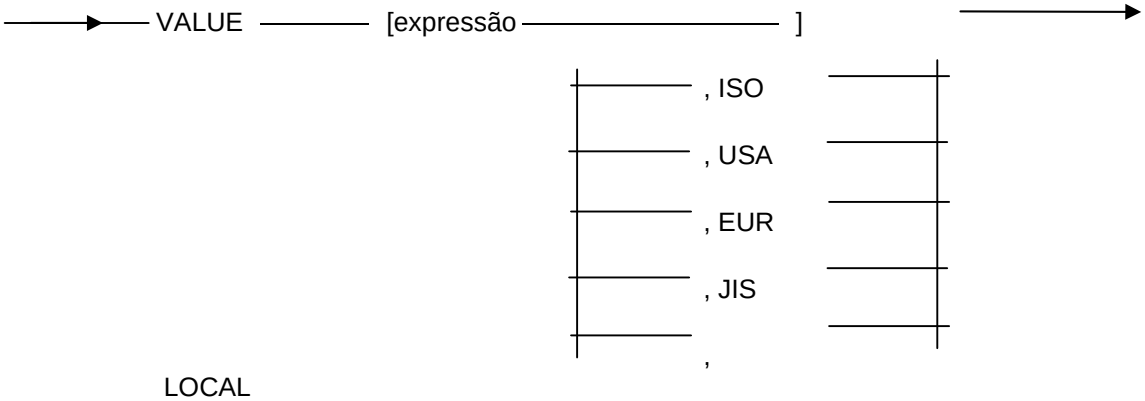
Gerar um relatório dos gerentes que ganham menos que R$ 18.000,00, se existir pelo menos um gerente com salário superior a R$ 22.000,00:

```
SELECT COD,
       NOME,
       SALARIO,
       DEPTO
FROM   SAL_COMIS
WHERE  CARGO = 'GER'     AND
       SALARIO < 18000   AND
       EXISTS
       (SELECT *
        FROM   SAL_COMIS
        WHERE  CARGO = 'GER' AND
SALARIO > 22000)
```

Resultado:

COD	NOME	SALARIO	DEPTO
30	PAULO	17506.75	38

6.2.3 Diagrama de sintaxe SQL (funções escalares)



6.2.4 Insert

Incluir novas linhas em uma tabela.

```
INSERT INTO <nome tabela>
            [(<nome coluna>, [<nome coluna>])]
       VALUES (<relação dos valores a serem incluídos>)
```

onde:

Conteúdo	Descrição
nome tabela	Representa o nome da tabela que será incluído o registro (linha)
nome coluna	Representa o nome da(s) coluna(s) que terão conteúdo no momento da operação. Este comando pode ser executado de duas maneiras: 1) **Quando todos os campos da tabela tiverem conteúdo** – Neste caso não é necessário especificar as colunas, entretanto, a relação dos valores a serem incluídos deverão obedecer a mesma seqüência da definição da tabela; 2) **Quando apenas parte dos campos da tabela tiverem conteúdo** – Neste caso devem ser especificados todas as colunas que terão conteúdo e os valores relacionados deverão obedecer esta seqüência. Para os campos que não têm conteúdo especificado será preenchido com o valor **NULL**.

6.2.4.1 Uma linha

Exemplo:

```
INSERT INTO SAL_COMIS
  VALUES (360, 'VINICIUS', 20, 'ATEND', 0, 13504.60, NULL)
```

ou

```
INSERT INTO SAL_COMIS
          (COD, NOME, DEPTO, CARGO, ANOS, SALÁRIO, COMIS)
  VALUES (360, 'VINICIUS', 20, 'ATEND', 0, 13504.60, NULL)
```

6.2.4.2 Múltiplas linhas

Exemplo:

Incluir na tabela de salários e comissões (SAL_COMIS), os colaboradores do departamento 15, 20 e 38 que se encontram na tabela auxiliar de salários e comissões (TAB_AUX)

COD	NOME	DEPTO	CARGO	ANOS	SALARIO	COMIS
370	ANA	20	ATEND	0	13504.60	?

380	VITOR	35	ATEND	0	13504.60	612.45
390	JORGE	38	ATEND	0	14000.00	?
400	ULISSES	25	GER	0	20000.00	?
410	MARINA	15	VENDAS	0	18500.00	?

```
INSERT INTO SAL_COMIS
        SELECT * FROM TAB_AUX
        WHERE DEPTO IN (15, 20 ,38)
```

6.2.5 Update

Atualizar os dados de uma ou mais linhas em uma tabela.

```
UPDATE <nome tabela>
  SET  <nome coluna> = <novo conteúdo>
       [<nome coluna> = <novo conteúdo>]
WHERE <condição>
```

onde:

Conteúdo	Descrição
nome tabela	Representa o nome da tabela cujo conteúdo será alterado.
nome coluna	Representa o nome da(s) coluna(s) que terão seus conteúdos alterados com o novo conteúdo especificado.
Condição	Representa a condição para a seleção das linhas que serão atualizados. Esta seleção poderá resultar em uma ou várias linhas. Neste caso a alteração irá ocorrer em todas as linhas selecionadas.

Exemplo (uma linha)

Alterar o valor de comissão da colaboradora Letícia (código 220) para R$ 882,80:

```
UPDATE  SAL_COMIS
  SET   COMIS = 882.80
  WHERE COD = 220
```

Exemplo (um subconjunto de linhas)

Reajustar o salário dos atendentes do departamento 38 em 10%::

```
UPDATE  SAL_COMIS
  SET   SALARIO = SALARIO * 1,10
  WHERE DEPTO = 38 AND
        CARGO = 'ATEND'
```

Exemplo (todas as linhas)

Acrescentar 1 (um) ano para o colaboradores:

```
UPDATE  SAL_COMIS
  SET   ANOS = ANOS + 1
```

6.2.6 Delete

Apagar (deletar) uma ou mais linhas em uma tabela

```
DELETE <nome tabela>
WHERE <condição>
```

onde:

Conteúdo	Descrição
nome tabela	Representa o nome da tabela cujo conteúdo será alterado.
Condição	Representa a condição para a deleção das linhas. Essa deleção poderá resultar em uma ou várias linhas. Neste caso a alteração irá ocorrer em todas as linhas selecionadas.

Exemplo (uma linha)

Deletar a linha que possua as informações de Vinícius:

```
DELETE  SAL_COMIS
  WHERE NOME = 'VINICIUS'
```

Exemplo (várias linhas)

Deletar as linhas que possuam departamento = 38:

```
DELETE  SAL_COMIS
  WHERE DEPTO = 38
```

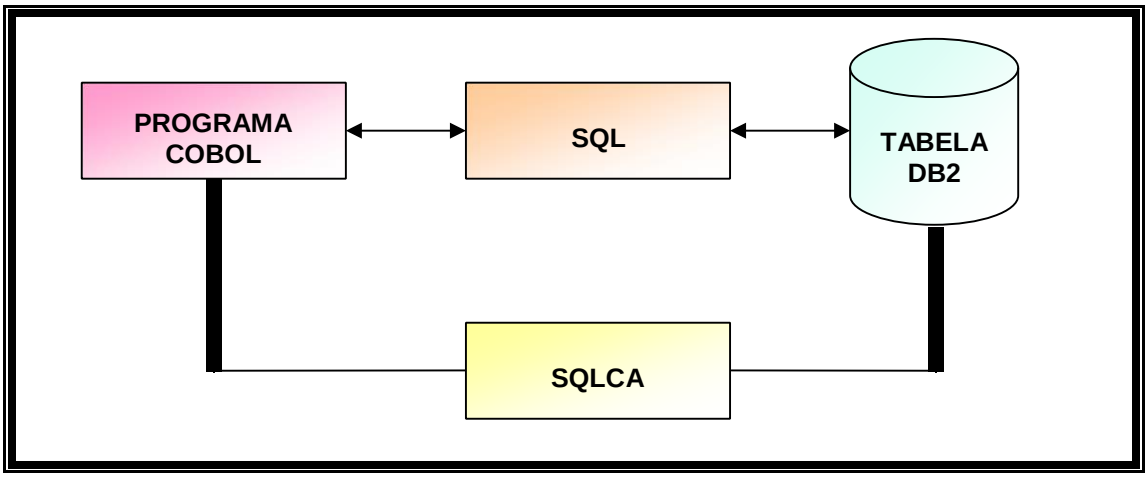
Exemplo (toda tabela)

```
UPDATE  SAL_COMIS
```

7 Programação em linguagens tradicionais (hospedeiras)

Os exemplos de comandos SQL vistos até aqui nos serviram para aprender a sintaxe e funcionamento da linguagem de maneira fácil. Porém, dentro do DB2 essa não é a maneira que utilizaremos para acessar as bases de dados. Na verdade, já vimos que o DB2 é um software que controla os acessos ao bando de dados, não uma linguagem de programação. Sendo assim, ele precisa ser acessado a partir de programas codificados em linguagens como COBOL, PL1, FORTRAN, ASSEMBLER etc.

Observe o esquema a seguir:



Conforme o esquemas acima, podemos afirmar que o COBOL servirá como uma linguagem hospedeira para o DB2 e fará todos os acessos à base de dados através da linguagem SQL..

Todos os comandos do DB2 (SQL) que serão executados dentro de um programa COBOL deverão ser precedidos da linha **EXEC SQL** e seguidos por **END-EXEC**, conforme sintaxe geral a seguir:

```
EXEC     SQL
         <comandos SQL>
END-EXEC
```

7.1 Variáveis Host

São utilizadas para permuta de valores entre a linguagem SQL e a linguagem de programação utilizada como hospedeira.

Sempre que um comando SQL se referencias a uma variável host, esta deverá ter seu nome precedido por dois pontos (:). Seu uso é facultativo, mas recomenda-se sua utilização para evitar possibilidades de confusão.

Variáveis adicionais, chamadas de variáveis indicadoras, são necessárias quando a coluna pode ter o valor nulo.

Seu tamanho deve ser compatível com a coluna, quanto ao seu data type e tamanho.

Observação:

Não podem ser utilizadas para referenciar nome de tabelas ou nome de colunas.

7.1.1 Fornecendo um valor

Uma das possíveis utilizações da variável host é a de fornecer um valor a uma instrução SQL, tornando-a genérica.

O uso da variável host é opcional, seu uso só é obrigatório para as instruções SELECT.

Exemplo:

- **COBOL**

```
      MOVE     2903                TO       CHAVE-REG
.
.
.
```

- **SQL**

```
      EXEC SQL
.
.
         WHERE    NUM_REG     =:    CHAVE-REG
END-EXEC
```

O exemplo acima mostra um SQL com valor à variável host. O programa atribui um valor à variável host, logo após, executa uma instrução SQL onde a referência à variável host é substituída pelo valor previamente atribuído.

7.1.1.1 Insert

```
      END-EXEC
```

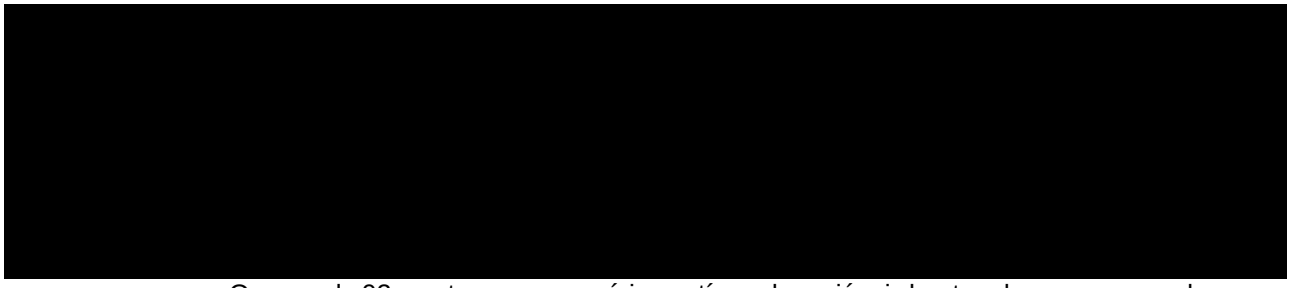
7.1.1.2 Update

7.1.2 Recebendo um valor

Para que acontece o recebimento de algum valor é necessário, uma variável host para cada coluna selecionada precedidas de dois pontos (:) em uma instrução SQL.

Exemplo 01:

Exemplo 02:



O exemplo 02 mostra que uma série contígua de variáveis host podem ser agrupadas em um conjunto chamado de estrutura. Esta estrutura pode ser referenciada e tratada como uma unidade. Esta técnica só é valida para o COBOL e PL1.

7.1.3 Definição de variáveis host

O programador pode usar o utilitário DCLGEN, para gerar variáveis.

• **Dados numéricos**

DB2	PL1	COBOL	ASSEMBLER	FORTRAN
Smallint	DCL N1 BIN FIXED(15);	01 N1 PIC S9(4) COMP.	N1 DS H	INTEGER *2 N1
Integer or Int	DCL N2 BIN FIXED(31);	01 N2 PIC S9(9) COMP.	N2 DS F	INTEGER *4 N2

Decimal (5,2) or Dec (5,2)	DCL N3 BIN FIXED(5,2);	01 N3 PIC S9(3)V9(2) COMP-3.	N3 DC PL5 '000.00'	REAL *8 N3
Float(21) or Real	DCL N4 BIN FLOAT(21);	01 N4 COMP-1.	N4 DS E	REAL *4 or REAL N4
Float or Float(53) or Double precision	DCL N5 BIN FLOAT(53)	01 N5 COMP-2.	N5 DS D	REAL *8 N5 or DOUBLE PRECISION N5

Observação:

A tabela acima mostra as correspondências em data type que devem ser observadas na declaração das variáveis host.

- **Dados alfanuméricos**

DB2	CHAR(10)	VARCHAR(80)
COBOL	01 STR1 PIC X(10).	01 STR2. 49 STR2L PIC S9(4) COMP. 49 STR2C PIC X(80).
PL1	DCL STR1 CHAR(10).	DCL STR2 CHAR(80) VAR:
ASSEMBLER	STR1 DS CL10	STR2 DS H.CL80
FORTRAN	CHARACTER*10 STR1	CHARACTER*80 STR2

- **Double byte character string:**

DB2	CHAR(10)	VARCHAR(80)
COBOL	01 DBC1 PIC G(5). DISPLAY -1.	01 DBC2. 49 DBC2L PIC S9(4) COMP. 49 DBC2C PIC G(40) DISPLAY-1.
PL1	DCL DBC1 GRAPHIC.	DCL DBC2 GRAPHIC(40) VAR;

- **Dados date / time:**

DB2	DATE	TIME	TIMESTAMP

COBOL	01 DT PIC X(10).	01 DT PIC X(08).	01 TS PIC X(26).
PL1	DCL DT CHAR(10).	DCL TM CHAR(08).	DCL TS CHAR(26).
ASSEMBLER	DCL DT CHAR(10).	TM DS CL8	TS DS CL26
FORTRAN	CHARACTER*10 DT	CHARACTER*8 TM	CHARACTER*26 TS

Observação:

Dados DATE / TIME são sempre convertidos para o carácter no assinalamento às variáveis host. As definições acima são para o padrão USA

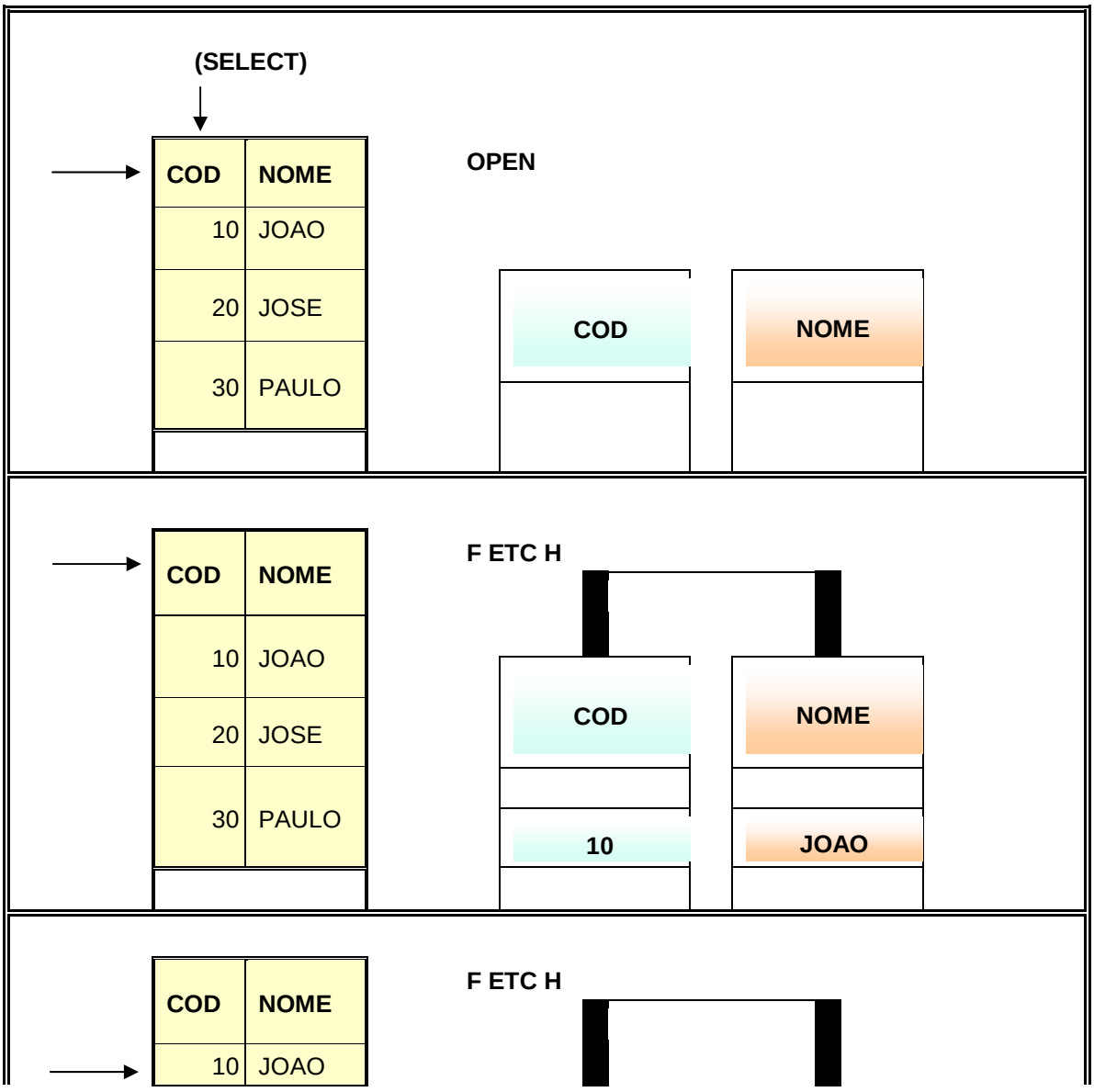
7.2 Processamento de múltiplas linhas

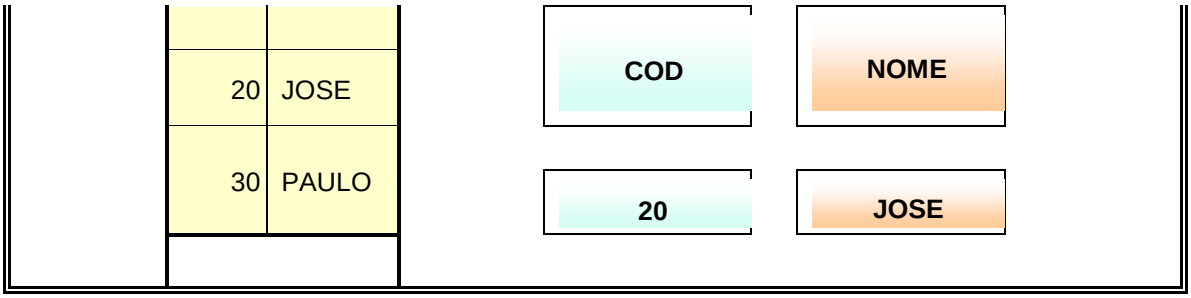
Se o resultado de um SELECT pode ter múltiplas linhas, o programa deve trabalhar com um CURSOR, pois não tem como prever o tamanho da tabela resultante.

Quando o programa executa um OPEN de um cursor, o DB2 executa o SELECT e a tabela resultante fica presa, esperando pelo processamento.

O programa deve prover variáveis HOST para cada coluna do cursor.

O programa executa a instrução SQL FETCH, para movimentar o cursor para o **próximo registro**. As variáveis HOST conterão os valores da linha apontada, conforme abaixo:





7.2.1 Select com Fetch

A instrução DECLARE CURSOR define a seleção dos dados que deverão compor a tabela resultante. Um NOME deve ser atribuído a esta tabela para futuras referencias dentro do programa.

A instrução DECLARE CURSOR não retorna nenhum valor ao programa. Isto é feito pela instrução FETCH.

Um programa típico emite FETCHs sucessivamente até receber o SQLCODE +100 na SQLCA, ou seja, fim da tabela resultante.

Um programa pode ter vários cursores abertos simultaneamente. O cursos deve ser fechado com CLOSE quando não é mais necessário. Quando o programa chegar ao fim, o DB2 fecha todos os cursosres que não forem fechados.

No exemplo que mostraremos abaixo, iremos declarar um cursor como FUT1; a instrução OPEN executará a seleção, que estará limitada pelos valores da variável host COD-COLABORADOR; o cursor não estará apontando para nenhuma linha; o FETCH apontará para a primeira linha da tabela resultante e o programa irá ler o valor das colunas através das variáveis NOME e CARGO:

- **Definindo o CURSOR**

```
     EXEC SQL
        DECLARE  FUT1          CURSOR    FOR
        SELECT
             NOME,
             CARGO
        FROM SAL_COMIS
        WHERE
             COD    = :COD-COLABORADOR      END-
EXEC.
```

- **Abrindo o CURSOR**

```
     EXEC SQL
        OPEN     FUT1      END-
EXEC.
```

- **Fetch das linhas**

```
    EXEC SQL
       FETCH    FUT1          INTO
            :NOME,
            :CARGO      END-
EXEC.
```

- **Fechando o CURSOR**

```
     EXEC SQL
        CLOSE    FUT1
     END-EXEC.
```

7.2.2 Delete via cursor

Um programa tem a opção de eliminar algumas linhas apontadas pelo cursor. A instrução DELETE, com a cláusula **WHERE CURRENT OF nome-do-cursor** faz com que a linha apontada pela última instrução FETCH seja eliminada.

A cláusula DELETE WHERE CURRENT OF não deve ser usada para o cursor cuja instrução SELECT contém:
– ORDER BY, GROUP BY, DISTINCT, UNION, FUNÇÃO e **JOIN.**

A estrutura do exemplo abaixo foi preparada para eliminar linhas apontadas por um cursor, isto permite uma decisão pela eliminação ou não da linha pelo exame prévio do conteúdo das mesmas.

- **Definindo o CURSOR**

```
     EXEC SQL
        DECLARE  FUT2          CURSOR    FOR
        SELECT
             DEPTO
        FROM SAL_COMIS
        WHERE
             DEPTO IN (10, 20, 42)      END-
EXEC.
```

- **Abrindo o CURSOR**

```
     EXEC SQL
        OPEN     FUT2      END-
EXEC.
```

- **Fetch das linhas**

```
     EXEC SQL
        FETCH    FUT2          INTO
:DEPTO      END-EXEC.
```

- **Deleção das linhas**

```
     EXEC SQL
        DELETE   FROM   SAL_COMIS
WHERE CURRENT OF FUT2      END-EXEC.
```

- **Fechando o CURSOR**

```
     EXEC SQL
        CLOSE    FUT2      END-
EXEC.
```

7.2.3 Update via cursor

Um programa pode fazer a atualização de algumas linhas apontadas pelo cursor. A atualização deve ser feita na linha que está sendo apontada pelo cursor especificado na cláusula **WHERE CURRENT OF nome-do-cursor**.
Somente as colunas especificadas na cláusula **FOR UPDATE OF lista-decolunas** da instrução DECLARE podem ser atualizadas. A atualização via cursor não é obrigatória.

A instrução DECLARE não pode ter a cláusula **FOR UPDATE OF** se a instrução SELECT contiver:
– ORDER BY, GROUP BY, DISTINCT, UNION, FUNÇÃO e **JOIN.**

- **Definindo o CURSOR**

```
     EXEC SQL
        DECLARE  FUT3          CURSOR    FOR
        SELECT
             NOME,
             CARGO
        FROM SAL_COMIS
        WHERE
             DEPTO = :COD-DEPTO
FOR UPDATE OF SALARIO      END-
EXEC.
```

- **Abrindo o CURSOR**

```
     EXEC SQL
        OPEN     FUT3      END-
EXEC.
```

- **Fetch das linhas**

```
     EXEC SQL
        FETCH    FUT3         INTO
             :NOME,
```

```
            :CARGO      END-
EXEC.
```

- **Atualização das linhas**

```
     EXEC SQL
        UPDATE   SAL_COMIS
        SET      SALARIO   =  :NOVO-SALARIO
WHERE CURRENT OF FUT3       END-EXEC.
```

- **Fechando o CURSOR**

```
     EXEC SQL
        CLOSE    FUT3       END-
EXEC.
```

7.2.4 Manipulação de cursor

7.2.4.1 Commit

Efetiva a atualização de dados nas tabelas, ou seja, o COMMIT fecha todos os cursores aberto e valida todas as atualizações feitas até então.

Não provoca perda de posição do cursor.

7.2.4.2 Rollback

Desfaz as atualização nas tabelas desde o último COMMIT.

7.2.4.3 Cursor hold

Como já vimos, a cláusula OPEN posiciona o cursor **antes** da primeira linha da tabela resultante; a cláusula FETCH avança o cursor uma linha para frente (próxima linha), portanto não é possível fazer o FETCH voltar o cursor.

Um cursor normal pode ser reaberto, mas a posição é restaurada para o topo. Se o cursor for do tipo WITH HOLD o posicionamento não é perdido.

Muitas vezes é necessária a emissão freqüente de COMMITs em programas de longa duração. As razões são o aumento da concorrência e a diminuição do tempo necessário em eventuais recuperações.

A cláusula CURSOR HOLD **não** pode ser utilizada em uma aplicação projetada para a modalidade pseudo conversacional CICS.

Exemplo:

```
     EXEC SQL
        DECLARE  FUT10  CURSOR  WITH HOLD  FOR
        SELECT
             NOME,
```

```
            CARGO
        FROM SAL_COMIS
        WHERE
            DEPTO = :COD-DEPTO      END-
EXEC.
```

7.3 SQLCA

SQL Communication Area é o nome dado à área de comunicação entre o programa e o DB2. É obrigatório e deve ser rotulado como SQLCA.

Deve ser verificado pelo programa para determinação do sucesso ou não da última execução de uma instrução SQL.

O layout dessa área precisa ser copiado para o programa, bastando para tanto, incluí-lo na WORKING-STORAGE SECTION, conforme abaixo:

```
EXEC     SQL
         INCLUDE SQLCA
END-EXEC
```

O comando INCLUDE é utilizado para copiar no programa ***BOOKs*** de registros do DB2. Pode ser utilizado tanto para a SQLCA, como para registros de tabelas catalogadas através do utilitário DCLGEN.

A SQLCA também disponibiliza um recurso muito importante chamado SQLCODE.

7.3.1 SQLCODE

Campo que contém o código de retorno dos acessos efetuados e **sempre deverá ser testado** após a execução de cada EXEC SQL codificada na PROCEDURE DIVISION descrevendo o código de sucesso ou insucesso da instrução SQL conforme abaixo:

- Um valor negativo indica que uma condição de erro foi encontrada e a instrução não foi efetuada;
- Um valor positivo e diferente de +100 indica que a instrução foi executada, mas algumas condições exigem atenção maior no programa;
- O valor +100 significa que os dados não foram encontrados. Pode ser resultado de um FETCH que atingiu o fim de uma tabela ou uma atualização que não encontrou nenhuma linha em condições de ser atualizada; e
- O valor zero indica que a execução foi bem sucedida.

O campo SQLWARN0 é usado em alguns casos de advertência. Devem ser examinados em caso de SQLCODEs positivos.

O terceiro campo do vetor inteiros SQLERRD informa a quantidade de linhas afetadas por uma requisição de alteração de dados (UPDATA, INSERT ou DELETE). A exceção ocorre no caso de haver uma deleção em massa de uma tabela contida em um tablespace segmentado. Esta situação é detectada com SQLERRD(3) = -1.

Possui 4 (quatro) posições no formato:

S	9	9	9

7.3.2 SQLSTATE

Padroniza códigos de erro de diferentes produtos. Está baseado nas especificações ANSI, possue 5 dígitos e está agrupado por classes.

Um SQLCODE pode corresponder a vários SQLSTATE e vice-versa.

O SQLCA contém o campo SQLSTATE que contém *return codes* padronizados para um ambiente de processamento distribuído.

A informação neste campo pode ser útil para determinar o que aconteceu no APPLICATION SERVER.

Informações mais detalhadas podem ser obtidas se o SQLSTATE for analisado em conjunto com o SQLCODE.

7.3.3 SQL warning

É um valor de caracteres de 1 byte. Se o byte 0 contém um W, os bytes seguintes também podem conter um W.

Se o byte 3 contém um W, não houve variáveis HOST suficientes para receber todas as colunas que forma selecionadas, por exemplo, um SELECT * pode ter sido executado em uma tabela que teve uma coluna adicionada.

O byte 4 aplica-se ao SQL dinâmico, ou seja, adverte o usuário sebre a possibilidade de se afetar todas as linhas de uma tabela devido a um UPDATE ou DELETE sem a cláusula WHERE.

O byte 6 indica correção de situações em que o valor de uma data ou TIMESTAMP resulta em um valor inválido após uma operação, por exemplo, a adição de 1 MONTH a 30 de janeiro resultará em 28 de fevereiro ao invés de 30 de fevereiro que é uma data inválida.

Podemos atribuir uma definição mais resumida, conforme abaixo:

- **SQLWARN1** – Valor da string truncado no assinalamento à variável host;
- **SQLWARN2** – Valores nulos ignorados no cálculo de uma função de coluna;
- **SQLWARN3** – Número de colunas maior que número de variáveis host;
- **SQLWARN4** – Update ou delete sem a cláusula where;
- **SQLWARN5** – Instrução SQL/DS inválida no DB2; e
- **SQLWARN6** – Correção de uma valor date ou timestamp com valor inválido resultante de uma operação aritmética.

7.3.4 Variáveis indicadoras

Se uma coluna selecionada permitir valor nulo, o DB2 irá requerer uam variável indicadora para registrar uma eventual nulidade.

É uma halfword (SMALLINT), deve ser especificada contigüamente à variável correspondente dentro de uma instrução SQL embutida.

Uma coluna que não aceita valor nulo, não requer uma variável indicadora.

Se a coluna apresentar valor nulo em um SELECT ou FETCH, o DB2 atribuirá -1 à variável indicadora e manterá inalterada a variável hot.

Se um programa envolvendo UPDATE ou INSERT atribuir -1 à variável indicadora, o DB2 assumirá valor nulo para a coluna correspondente.

Exemplo:

```
EXEC SQL
   SELECT
        CARGO, DEPTO, SALARIO, COMIS
   INTO :CARGO-REG :CG,
        :NOME-REG,
        :SAL-REG,
        :COMIS
   FROM SAL_COMIS
   WHERE
        COD  = :COD-REG
END-EXEC.
```

	Coluna CARGO	Var. host :CARGO-REG	Var. Indicadora :CG
SELECT / FETCH	"ATEND" NULO	"ATEND" Inalterado	0 -1
UPDATE / INSERT	"VENDAS" NULO	"VENDAS" Inalterado	0 -1

7.3.5 Vetor de variáveis indicadoras

Se as variáveis host estão arranjadas em uma estrutura e pelo menos um de seus componentes trabalha com valor nulo, o programa deve trabalhar com um vetor de variáveis indicadoras.

O vetor deve conter uma variável indicadora para cada componente da estrutura, inclusive para aqueles que não trabalham com nulo. A partir do último componente que trabalha com nulo não é mais necessário definir elementos para o vetor.

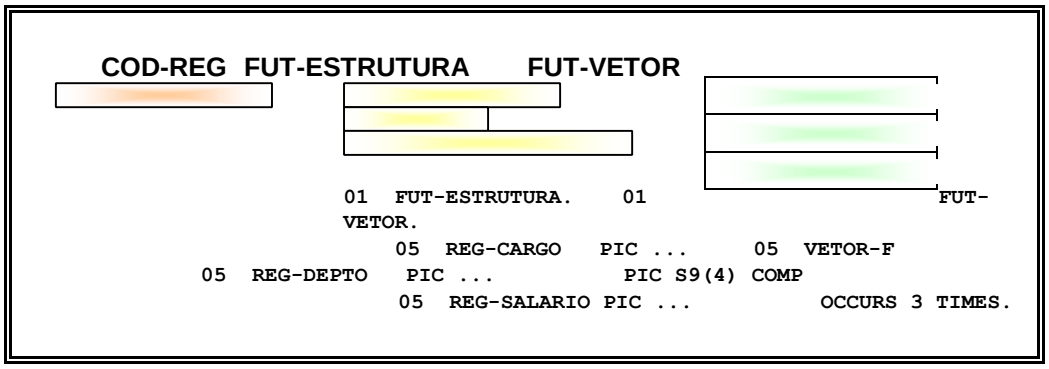
O vetor é uma múltipla ocorrência de variáveis halfword.

Sua utilização vale somente para COBOL e PL1, que aceitam trabalhar com estruturas.

Exemplo:

```
EXEC SQL
   SELECT
        CARGO, DEPTO, SALARIO
   INTO :FUT-ESTRUTURA,
```

```
            :FUT-VETOR
          FROM SAL_COMIS
          WHERE
               COD   = :COD-REG       END-
EXEC.
```



7.3.6 Uso de variáveis indicadoras

Variáveis indicadoras	I n t e r p r e t a ç ã o
0	Variável host contém valor nulo;
-1	Coluna com valor nulo, variável host inalterada;
-2	Resultado nulo devido ao erro aritmético ou de conversão;
N (>0)	String de “n” caracteres truncado no assinalamento à variável host

As variáveis indicadoras podem ser utilizadas em outras situações:

- Valor 0 (zero) – Indica variável host com valor não nulo;
- -1 – Indica que a coluna tem ou deve assumir valor nulo e a variável host permanece inalterada;
- -2 – Aplica-se ao SELECT externo (subquery) ou ao FETCH, indica que a coluna tem resultado nulo e a variável host permanece inalterada. O teste de SQLCODE dá informações adicionais:
  - **+304** – Indica erro de conversão, por exemplo, uma variável host não conseguiu comportar um valor grande;
  - **+802** – Indica erro aritmético resultante de divisão por zero ou overflow.

Erros de conversão ou de aritmética resultam em SQLCODE -304 ou -802 na ausência da variável indicadora.

“n” indica que uma variável host é pequena demais para conter um string de “n” caracteres, neste caso, o string é truncado e o SQLWARN1 é setado para “W”.

7.3.7 Formato SQLCA

SQLCA CHAR(8)			
SQLCABC INTEGER			
SQLCODE INTEGER			
SQLLERRM VARCHAR (70)			
SQLERRP CHAR (8)			
SQLERRD(1) INTEGER			
SQLERRD(2) INTEGER			
SQLERRD(3) INTEGER			
SQLERRD(4) INTEGER			
SQLERRD(5) INTEGER			
SQLERRD(6) INTEGER			
SQLWARN0	SQLWARN1	SQLWARN2	SQLWARN3
SQLWARN4	SQLWARN5	SQLWARN6	SQLWARN7
SQLWARN8	SQLWARN9	SQLWARNA	SQLWARNB

Observação:.

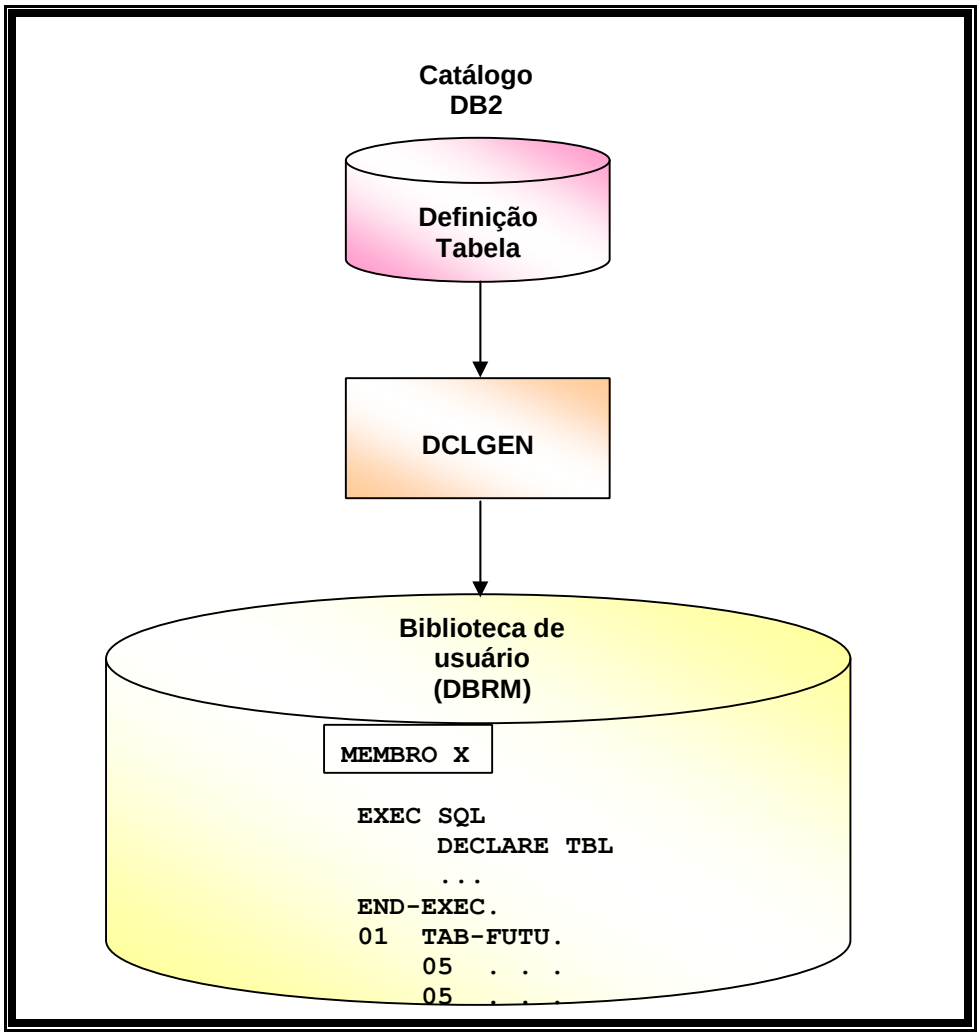
SQLWARNx são todos CHAR(1)

8 DCLGEN

É um programa que gera uma codificação de declaração de variáveis host correspondentes às colunas de uma tabela dada.

Usa o catálogo do DB2 para montar na biblioteca do usuário a instrução SQL 'DECLARATE TABLE' (usado pelo pré compilador para verificação de nomes dos objetos, nome das colunas e a correspondência entre eolunas e variáveis host) e o código de declarações das variáveis host (data structure).

A saída do DCLGEN é inteiramente gravada em um único membro de uma biblioteca, conforme abaixo:



8.1 Saída DCLGEN - COBOL

```
************************************************************************
    *         DCLGEN TABLE(DBADB2.TB_FUCUR                                 *
    *         LIBRARY(DBADB2.JCL.DATA(FUCUR))                              *
    *         ACTION(REPLACE)                                              *
    *         LANGUAGE(COBOL)                                              *
    *         STRUCTURE(FUCUR)                                             *
    *         APOST                                                        *
    *         LABEL(YES)                                                   *
    *         ... IS THE DCLGEN COMMAND THAT MADE THE FOLLOWING STATEMENTS *
************************************************************************
```

SQL DECLARE (usado na pré compile)

```
EXEC SQL DECLARE DBADB2.TB_FUCUR TABLE
( CO_CURSO                       SMALLINT NOT NULL,
  NO_CURSO                       CHAR(30) NOT NULL,
  IC_PER_CURSO                   CHAR(2) NOT NULL,
  DT_INI_CURSO                   DATE NOT NULL,
```

```
           DT_FIM_CURSO                   DATE NOT NULL,
    ) END-EXEC.
```

Estrutura COBOL para I/O de linhas de tabelas

```
   ******************************************************************
* COBOL DECLARATION FOR TABLE DBADB2.TB_FUCUR                       *
   ******************************************************************
    01  FUCUR.
* ***************************************************************
  10 CO-CURSO             PIC S9(4)  USAGE COMP.
* ***************************************************************
  10 NO-CURSO             PIC X(30).
* ***************************************************************
  10 IC-PER-CURSO         PIC X(2).
* ***************************************************************
  10 DT-INI-CURSO         PIC X(10).
* ***************************************************************
  10 DT-FIM-CURSO         PIC X(10).
   ******************************************************************
* THE NUMBER OF COLUMNS DESCRIBED BY THIS DECLARATION IS 5          *
   ******************************************************************
```

Este é o resultado da execução do DCLGEN.

Na parte superior temos a instrução DECLARE com o nome da tabela. Colunas e seus data types. (Notar que o nome da estrutura deriva da tabela).

O nome das variáveis são os nomes das colunas correspondentes.

As variáveis para colunas VARCHAR são subdivididas em dois componentes: comprimento e texto.

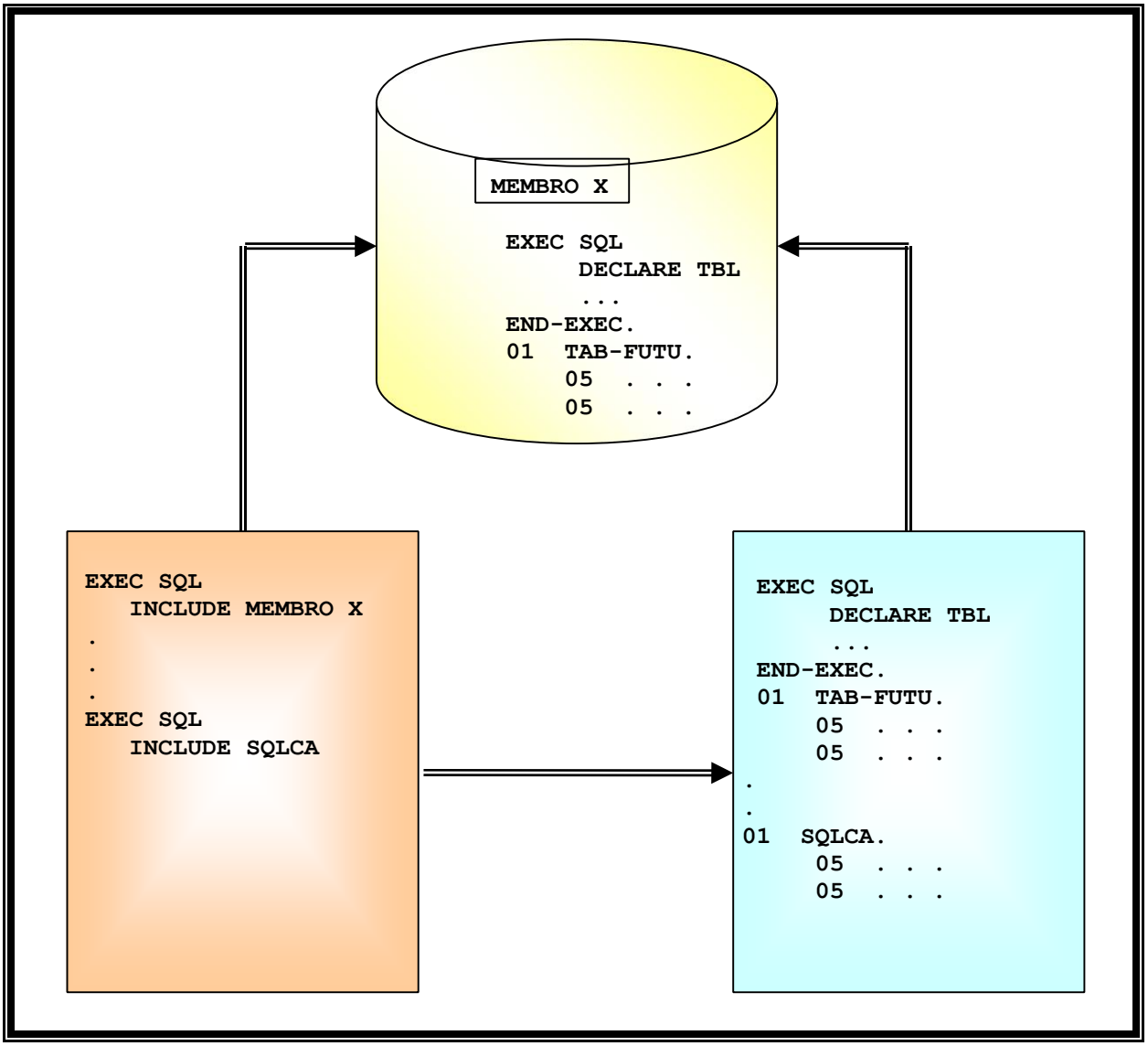
Observação.:
Variáveis indicadoras não são geradas pelo DCLGEN.

8.2 Instrução SQL include

Quando o pré compilador encontra uma instrução INCKLUDE, faz a sua substituição pela saída do DCLGEN.
O INCLUDE aponta para o membro que contém a saída do DCLGEN.

A estrutura de uma SQLCA também pode ser incluída via INCLUDE. Como este bloco está internamento embutido no DB2, não há necessidade de cópia a partir de uma biblioteca.

Veremos abaixo sua estrutura:



9 Preparação e execução do programa

A preparação de programa é o NOME que damos ao processo que torna um programa fonte em um módulo executável.

Esta preparação pode ser feita das seguintes maneiras:

a) Com um job editado à partir da PROC do **Program Preparation on DB2**;
b) Com um job montado pelo programa DSNH; ou
c) Interativamente com o programa DSNH.

Em linhas gerais a preparação de um programa é constituído das seguintes fases:

- Pré compilação;
- Compilação e Linkedição; e
- BIND

O pré compilador pode ser executado com o DB2 fora do ar; a sua função é gerar à partir do fonte dois data sets selados com um timestamp. Os data sets gerados são:

- DBRM (Database Request Module) – Contém todas as instruções SQL embutidas no fonte; e
- Fonte Modificado – É copiado para este data set com modificações. As instruções SQL são transformadas em comentários e os CLLs para o DB2 são inseridos logo a seguir; eventualmente códigos de definição de variáveis HOST são copiados à partir de uma biblioteca de INCLUDE (gerada pelo DCLGEN).

O pré compilador faz a checagem da sintaxe das instruções SQL e da validade das variáveis HOST utilizadas. Um relatório de execução com eventuais mensagens de erro também é gerada.

O compilador gera o módulo objeto à partir do fonte modificado pelo pré compilador. O timestamp é copiado do fonte.

O link editor gera módulo de carga à partir do módulo da linguagem de interface do DB2. O timestamp original é copiado do módulo objeto.

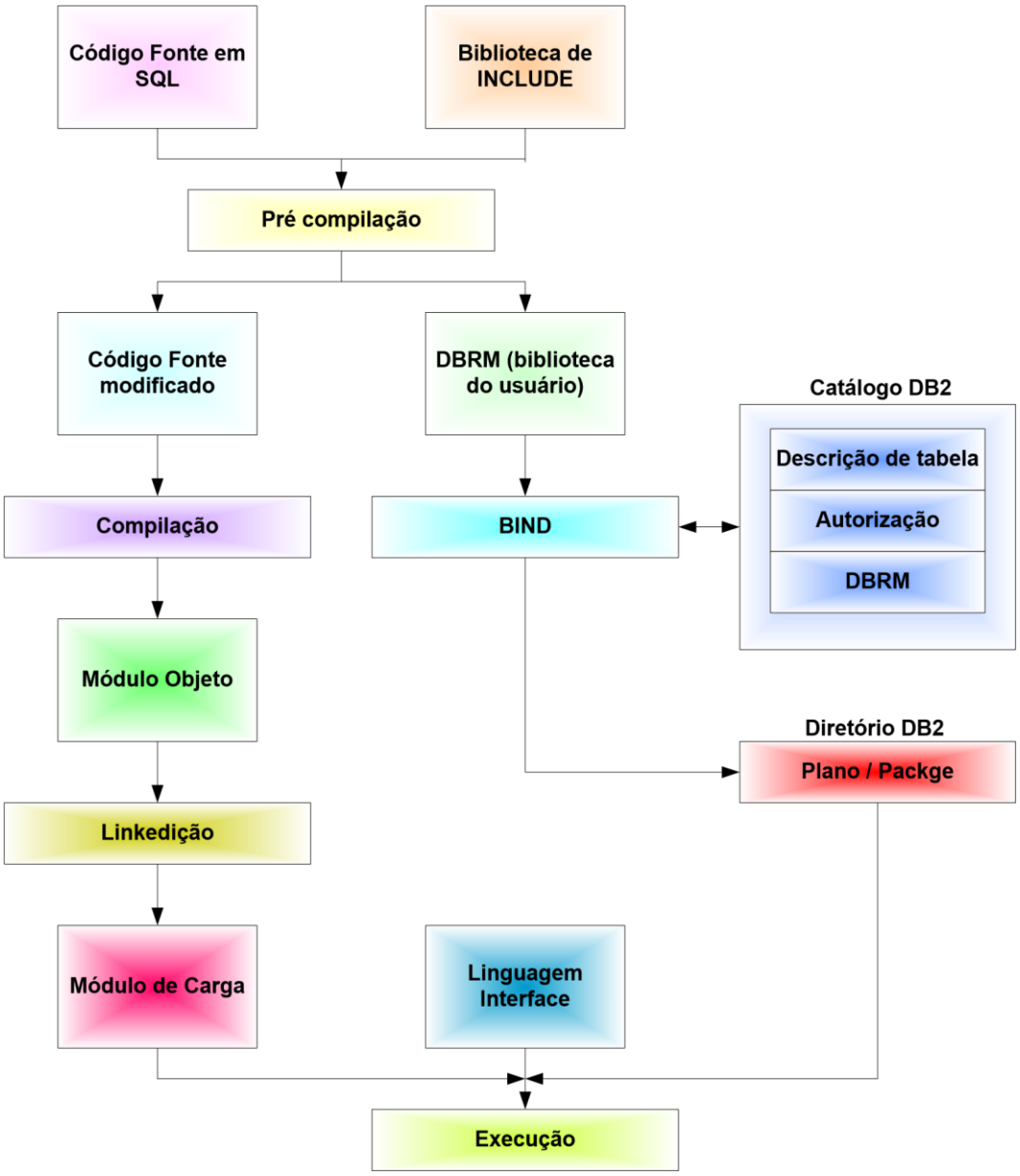
O processador de BIND gera o plano ou package à partirm do DBRM. O timestamp é copiado do DBRM.

Um plano ou packge contém a estratégia de acesso ao dados DB2 com informações do tipo: • Índices utilizados;

- Estratégia de locking;
- Estratégia de execução de um JOIN; etde; e

A fase de compilação / link edição e a fase do BIND são executadas separadamente e podem ser executadas em qualquer ordem.

Abaixo mostraremos o fluxo de preparação e execução de um programa:



9.1 Pré compilador

O pré compilador é executado independentemente do DB2, ou seja, o DB2 não precisa estar no ar e portanto não é feito a checagem com os dados do catálogo. Isto faz com que a carga do desenvolvimento sobre o DB2 seja diminuída.

A instrução DECLARATE TABLE é utilizada para verificação do NOME dos objetos, NOME das colunas e a correspondência entre colunas e variáveis HOST.

Cria um arquivo com o fonte modificado da seguinte maneira:

- Instruções EXEC SQL transformadas em comentários;
- Os comentários das instruções EXEC SQL são seguidos de CALL para o DB2;
- Se existir uma instrução WHENEVER, será inserida uma lógica 'IF ... GO TO' logo após o CALL para o DB2;
- Com a cópia de declaração das tabelas e das variáveis HOST conforme especificada na instrução INCLUDE; e

- Com cópia da estrutura da SQLCA, se assim especificado no INCLUDE.

Cria um DBRM com as instruções SQL copiadas.

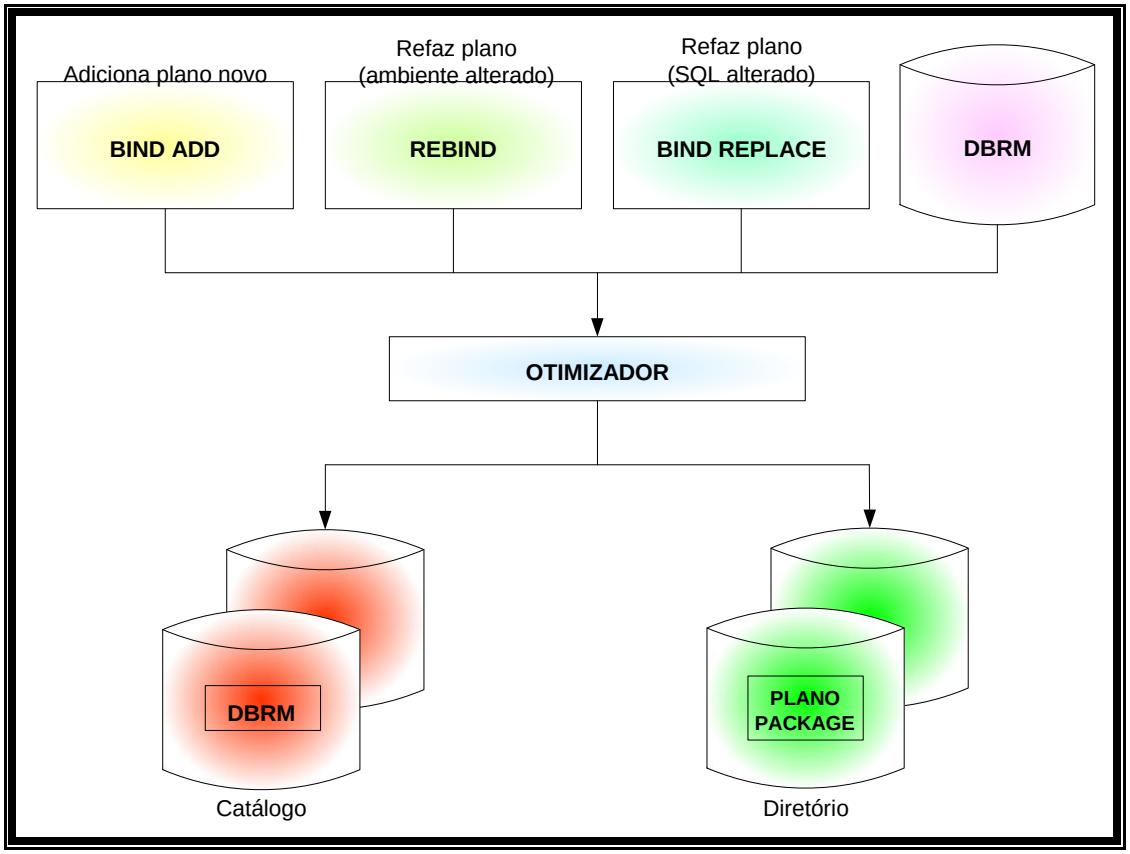
9.2 BIND

O propósito do BIND é a geração do PLANO ou de um PACKAGE.

A partir do DBRM e do catálogo são executados os seguintes procedimentos

- Checagem da consistência das instruções SQL com os objetos manipulados. São checados nomes, data type das colunas etc.
- Checagem da autorização necessária para a execução das instruções SQL;
- Determinação da estratégia de acesso aos objetos de manipulação, pelo otimizador do DB2;
- O plano ou o package é armazenado no diretório e uma cópia do DBRM é carregado no catálogo;
- **BIND ADD** – Utilizado para criar um novo plano / package;
- **BIND REPLACE** – Utilizado quando a própria instrução SQL foi alterada e queremos refazer o plano ou o package existente; e
- **REBIND** – Utilizado quando o SQL permanece inalterado, mas queremos refletir um modificação ocorrida no ambiente tais como criação / eliminação de índice ou atualização das estatísticas do catálogo.

Abaixo mostraremos o fluxo de execução do BIND:



O otimizador do DB2 pesa fatores como CPU e I/O para elaborar uma estratégia de acesso.

9.3 Estratégia de acesso

A estratégia de acesso de uma instrução SQL é determinada pelo otimizador do DB2. O otimizador procura a melhor opção de acesso para minimizar o consumo de recursos e I/O da CPU. Também se baseia nas informações estatísticas contidas no catálogo, tais informações são atualizadas pelo utilitário RUNSTATS.

Exemplo:

```
SELECT
     *
FROM SAL_COMIS
WHERE
     (CARGO = VEND OR SALARIO > 10000) AND
     (DEPTO IN (15,42))
```

Para que o otimizador consiga melhorar a performance, basta efetuar as perguntas abaixo e de acordo com as respectivas respostas, poderá obter êxito:

- Índices disponíveis?
- Número de páginas lidas se tablespace varrido?
- Número de páginas lidas se utilizado índice?
- Melhor ordem de execução dos testes?

9.4 Execução de programa

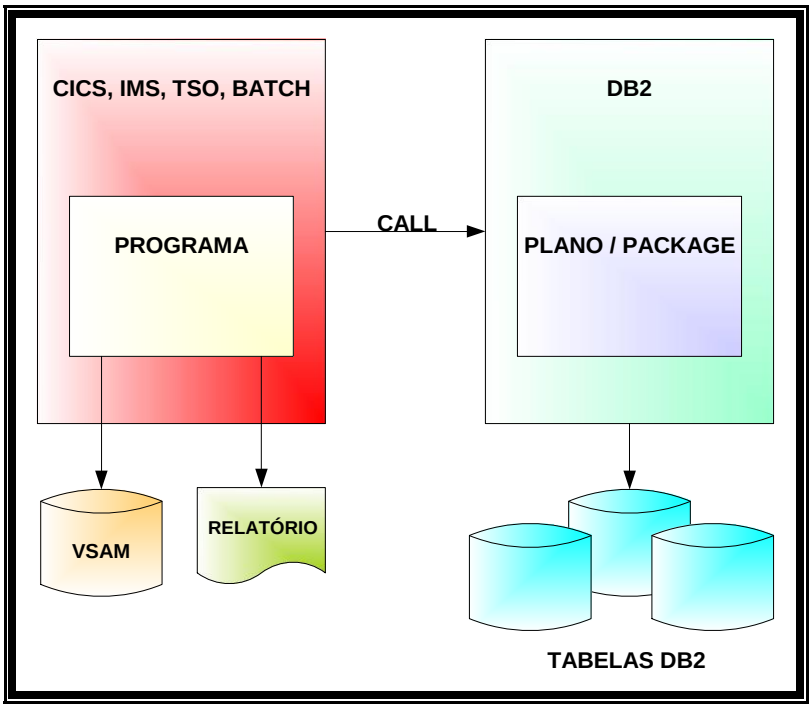
O programa é executado no address space do usuário enquanto o plano é executado no address space do DB2.

O acesso às tabelas são feitos através de CALLs para o DB2.

Para impedir que eventualmente um programa venha a utilizar um plano que não lhe pertence, o DB2 chaca se os timestamps do módulo de carga e do plano são iguais. O programa não é executado se não forem coincidentes.

O DB2 suporta também versões diferentes de packages com **nome** igual mas de versões diferentes. Basta para isto que as diferentes versões estejam na lista do plano. Assim de acordo com a versão do programa que esá em execução e carregando o package correspondente.

Mostraremos abaixo o fluxo de execução:



9.5 Locking

Indica que o dado selecionado está em uso. O propósito do mecanismo de LOCKING é administrar os múltiplos acessos aos dados do DB2 para que os dados sejam acessados e atualizados e forma organizada evitando inconsistências.

Qualquer tipo de acesso aos dados DB2 passam pelo mecanismo de locking, garantindo proteção contra:

- Acesso a um dado com alteração não validada; e
- Atualização de um dado em uso.

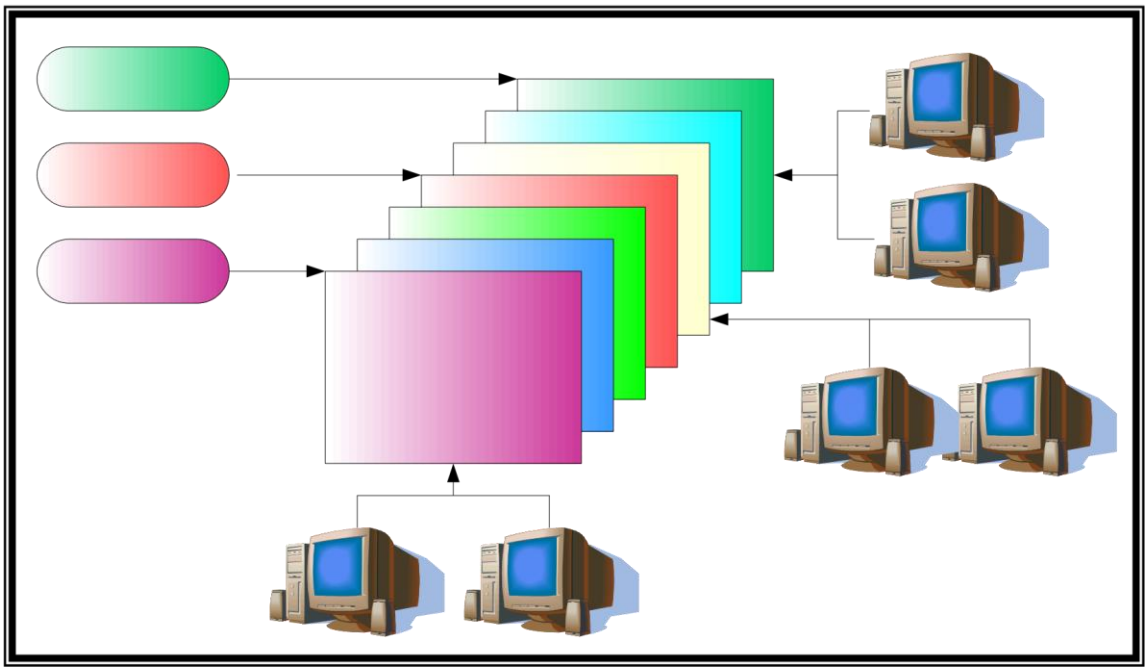
Um lock tem três características:

- Modo; • Granularidade; e
- Duração.

Entenda-se modo de LOCKING como modalidade quanto à exclusividade sobre um recurso que pode ser:

- **S** (**shared**) – Indica compartilhado; o dado está em uso para leitura por um ou mais usuários;
- **X** (**exclusivo**) – Indica exclusividade; o dado está em uso dedicado para um único usuário e os demais interessados ficam enfileirados; e
- **U** (**update**) – Indica que o dado está sendo usado por um usuário com intenção de atualizá-lo; o DB2 aceita compartilhado de leitura enquanto o usuário não resolver atualizar o dado, se o usuário decidir pela atualização ocorrerá espera pelo término do serviço dos demais usuários e enfileiramento de novos usuários querendo acessar os dados para que o lock "U" seja elevado para lock "X".

Abaixo mostraremos o modo LOCKING:



9.6 Granularidade

A granularidade do locking indica o tamanho dos recursos que estão sendo presos.

O lock pode ser dado:

- Individualmente à linhas ou páginas;
- À todos os segmentos de uma tabela de um tablespace segmentado; e
- Individualmente à um tablespace.

Lock em nível de páginas exige mais atividade da parte do DB2 mas oferece maior concorrência.

A granularidade é um dos parâmetros de um tablespace durante a sua definição.
O parâmetro de granularidade pode ser alterado nas situações abaixo:

- Emissão do comando LOCK TABLE pelo usuário; e
- O DB2 pode elevar o nível de lock se o overhead de processamento devido ao locking em nível de página ultrapassar o limite da instalação (parâmetro de geração do DB2)

LOCK em nível de tabela / tablespace é menos trabalhoso para o DB2, mas pode ocorrer um maior enfileiramento pela espera de recursos.

10 Utilitários DB2

Utilitário	Função / Descrição	Processamento concorrente
CHECK INDEX	Testa se os índices estão consistentes com os dados	READ-ONLY PROCESSING
CHECK DATA	Testa integridade referencial	READ-ONLY PROCESSING
COPY	Cria uma cópia imagem de uma tablespace ou um dataset dentro de uma tablespace	ANY TYPE OF PROCESSING
LOAD	Carrega dados de um arquivo seqüencial para uma ou mais tabelas no mesmo tablespace	NONE IN TABLESPACE OR PARTITION BEING LOADED
MERGECOPY	Merge de cópias imagem incrementais ou totais de uma nova cópia imagem incremental e/ou total	ANY TYPE OF PROCESSING
MODIFY	Deleta registro de uma cópia imagem do 'SYSISBM' da tabela catálogo SYSCOPY e o log relacionado da SYSIBM.SYSLGRNG	ANY TYPE OF PROCESSING
QUIESCE	Estabelece um QUIESCE POINT no catálogo para o(s) tablespace(s)	READ-ONLY PROCESSING
RECOVER	Recupera a tablespace, uma págima, partição ou ERRORANGE	NONE IN REPARED TABLESPACE
REORG	Reorganiza um tablespace ou um índice	READ-ONLY DURING UNLOAD; NONE DURING RELOAD
REPAIR	Substitui dados inválidos por dados válidos	NONE IN REPARED TABLESPACE
REPORT	Lista informações de recuperação de tablespace	ANY TYPE OF PROCESSING
RUNSTATS	Atualiza tabelas catálogo com estatísticas sobre utilização de tablespaces, eficiência de índices	ANY TYPE OF PROCESSING
STOSPACE	Atualiza tabelas catálogo com estatísticas sobre utilização de espaço de STORAGE GROUP	ANY TYPE OF PROCESSING

11 Formatação de mensagens de erro

São 2 (duas), as rotinas que geram mensagens de acordo com o SQLCODE na SQLCA, encontradas na biblioteca de samples do DB2: DSNTIAR (utilizada no COBOL, PL1 e ASSEMBLER) e DSNTIR (utilizada no FORTRAN).

Veja a sintaxe, conforme a linguagem utilizada, na tabela abaixo:

Linguagem	Sintaxe
COBOL	CALL **DSNTIAR** USING SQLCA MESSAGE LRECL
PL1	CALL **DSNTIAR** (SQLCA, MESSAGE, LRECL)
ASSEMBLER	CALL **DSNTIAR** (SQLCA, MESSAGE, LRECL)
FORTRAN	CALL **DSNTIR** (MSGLEN, MESSAG, ICODE)

O CALL do DSNTIAR em COBOL, PL1 ou ASSEMBLER deve ter 3 (três) parâmetros: **SQLCA** (contém o SQLCODE e os caracteres que devem ser inseridos no esqueleto da mensagem correspondente); **MESSAGE** (a variável onde será colocada a mensagem formatada) e **LRECL** (um número inteiro fornecendo o comprimento desejado para a linha da mensagem).

O CALL do DSNTIR obedece a regra acima, porém só deve ser utilizado para a linguagem FORTRAN.

12 Códigos de erros (SQLCODES)

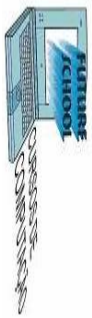
Observação:
Não são todos os códigos que conseguimos traduzir para o português.

12.1 Códigos positivos:

SQLCODES	DESCRIÇÃO
000	SUCCESSFUL EXECUTION
+ 012	THE UNQUALIFIED COLUMN NAME *column-name* WAS INTERPRETED AS A CORRELATED REFERENCE
+ 098	A DYNAMIC SQL STATEMENT ENDS WITH A SEMICOLON.
+ 100	ROW NOT FOUND FOR FETCH, UPDATE OR DELETE, OR THE RESULT OF A QUERY IS AN EMPTY TABLE
+ 110	SQL UPDATE TO A DATA CAPTURE TABLE NOT SIGNALED TO ORIGINATING SUBSYSTEM
+ 111	THE SUBPAGES OPTION IS NOT SUPPORTED FOR TYPE 2 INDEXES
+ 117	THE NUMBER OF INSERT VALUES IS NOT THE SAME AS THE NUMBER OF OBJECT COLUMNS
+ 162	TABLESPACE *databasename.tablespace-name* HAS BEEN PLACED IN CHECK PENDING
+ 203	THE QUALIFIED COLUMN NAME *column-name* WAS RESOLVED USING A NON-UNIQUE OR UNEXPOSED NAME
+ 204	*name* IS AN UNDEFINED NAME
+ 206	*column-name* IS NOT A COLUMN OF AN INSERTED TABLE, UPDATED TABLE, OR ANY TABLE IDENTIFIED IN A FROM CLAUSE
+ 218	THE SQL STATEMENT REFERENCING A REMOTE OBJECT CANNOT BE EXPLAINED
+ 219	THE REQUIRED EXPLANATION TABLE *table-name* DOES NOT EXIST
+ 220	THE COLUMN *column-name* IN EXPLANATION TABLE *table-name* IS NOT DEFINED PROPERLY
+ 222	DELETE HOLE DETECTED USING *cursor-name*
+ 231	CURSOR POSITION OF CURSOR *cursor-name* IS NOT VALID FOR FETCH OF THE CURRENT ROW
+ 236	SQLDA INCLUDES *integer1* SQLVAR ENTRIES, BUT *integer2* ARE REQUIRED FOR *integer3* COLUMNS
+ 237	SQLDA INCLUDES *integer1* SQLVAR ENTRIES, BUT *integer2* ARE REQUIRED BECAUSE AT LEAST ONE OF THE COLUMNS BEING DESCRIBED IS A DISTINCT TYPE
+ 238	SQLDA INCLUDES *integer1* SQLVAR ENTRIES, BUT *integer2* SQLVAR ENTRIES ARE NEEDED FOR *integer3* COLUMNS BECAUSE AT LEAST ONE OF THE COLUMNS BEING DESCRIBED IS A LOB
+ 239	SQLDA INCLUDES *integer1* SQLVAR ENTRIES, BUT *integer2* ARE REQUIRED FOR *integer3* COLUMNS BECAUSE AT LEAST ONE OF THE COLUMNS BEING DESCRIBED IS A DISTINCT TYPE
+ 304	A VALUE WITH DATA TYPE *data-type1*CANNOT BE ASSIGNED TO A HOST VARIABLE BECAUSE THE VALUE IS NOT WITHIN THE RANGE

	OF THE HOST VARIABLE IN POSITION *position-number* **WITH DATA TYPE** *data-type2*
+ 331	**THE NULL VALUE HAS BEEN ASSIGNED TO A HOST VARIABLE BECAUSE THE STRING CANNOT BE TRANSLATED. REASON** *reason-code*, **CHARACTER** *code-point*, **HOST VARIABLE** *position-number*
+ 339	**THE SQL STATEMENT HAS BEEN SUCCESSFULLY EXECUTED, BUT THERE MAY BE SOME CHARACTER CONVERSION INCONSISTENCIES**
+ 394	**USER SPECIFIED OPTIMIZATION HINTS USED DURING ACCESS PATH SELECTION**
+ 395	**USER SPECIFIED OPTIMIZATION HINTS ARE INVALID (REASON CODE =** *reason-code*). **THE OPTIMIZATION HINTS ARE IGNORED**
+ 402	**LOCATION** *location* **IS UNKNOWN**
+ 403	**THE LOCAL OBJECT REFERENCED BY THE CREATE ALIAS STATEMENT DOES NOT EXIST**
+ 434	**OPTION** *keyword* **IS A DEPRECATED FEATURE**
+ 445	**VALUE** *value* **HAS BEEN TRUNCATED**
+ 462	**EXTERNAL FUNCTION OR PROCEDURE** *name* **(SPECIFIC NAME** *specific-name*) **HAS RETURNED WARNING SQLSTATE, WITH DIAGNOSTIC TEXT** *text*
+ 464	**PROCEDURE** *proc* **RETURNED** *num* **QUERY RESULT SETS, WHICH EXCEEDS THE DEFINED LIMIT** *integer*
+ 466	**PROCEDURE** *proc* **RETURNED** *num* **QUERY RESULTS SETS**
+ 494	**NUMBER OF RESULT SETS IS GREATER THAN NUMBER OF LOCATORS**
+ 495	**ESTIMATED PROCESSOR COST OF** *estimate-amount1* **PROCESSOR SECONDS (***estimate-amount2* **SERVICE UNITS) IN COST CATEGORY** *cost-category* **EXCEEDS A RESOURCE LIMIT WARNING THRESHOLD OF** *limitamount* **SERVICE UNITS**
+ 535	**THE RESULT OF THE POSITIONED UPDATE OR DELETE MAY DEPEND ON THE ORDER OF THE ROWS**
+ 541	**THE REFERENTIAL OR UNIQUE CONSTRAINT** *name* **HAS BEEN IGNORED BECAUSE IT IS A DUPLICATE**
+ 551	*auth-id* **DOES NOT HAVE THE PRIVILEGE TO PERFORM OPERATION** *operation* **ON OBJECT** *object-name*
+ 552	*auth-id* **DOES NOT HAVE THE PRIVILEGE TO PERFORM OPERATION** *operation*
+ 558	**THE WITH GRANT OPTION IS IGNORED**
+ 561	**THE ALTER, INDEX, REFERENCES, AND TRIGGER PRIVILEGES CANNOT BE GRANTED PUBLIC AT ALL LOCATIONS**
+ 562	**A GRANT OF A PRIVILEGE WAS IGNORED BECAUSE THE GRANTEE ALREADY HAS THE PRIVILEGE FROM THE GRANTOR**
+ 585	**THE SCHEMA NAME** *schema-name* **APPEARS MORE THAN ONCE IN THE CURRENT PATH**
+ 599	**COMPARISON FUNCTIONS ARE NOT CREATED FOR A DISTINCT TYPE BASED ON A LONG STRING DATA TYPE**
+ 610	**A CREATE/ALTER ON OBJECT** *object-name* **HAS PLACED OBJECT IN** *utility* **PENDING**
+ 645	**WHERE NOT NULL IS IGNORED BECAUSE THE INDEX KEY CANNOT CONTAIN NULL VALUES**
+ 650	**THE TABLE BEING CREATED OR ALTERED CANNOT BECOME A DEPENDENT TABLE**
+ 653	**TABLE** *table-name* **IN PARTITIONED TABLESPACE** *tspace-name* **IS NOT AVAILABLE BECAUSE ITS PARTITIONED INDEX HAS NOT BEEN CREATED**

future school
informática

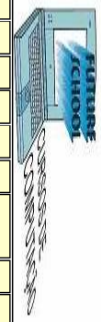


ALLOWED IN FUTURE RELEASES
EDS THE LENGTH IMPOSED BY DB2
EMS
TE
DATA, POSITION *position-number*
AND LOCKMAX 0
NE, OR AN OPERATION THAT

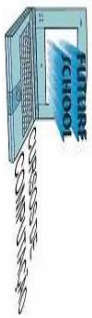
12.2 Códigos negativos:

SQLCODES	DESCRIÇÃO
- 007	STATEMENT CONTAINS THE ILLEGALCHARACTER *character*
- 010	THE STRING CONSTANT BEGINNING *string* IS NOT TERMINATED
- 029	INTO CLAUSE REQUIRED
- 060	INVALID *type* SPECIFICATION : *spec*
- 079	QUALIFIER FOR DECLARED GLOBAL TEMPORARY TABLE *table-name* MUST BE SESSION, NOT *qualifier*
- 084	UNACCEPTABLE SQL STATEMENT
- 097	THE USE OF LONG VARCHAR OR LONG VARGRAPHIC IS NOT ALLOWED IN THIS CONTEXT
- 101	THE STATEMENT IS TOO LONG OR TOO COMPLEX
- 102	LITERAL STRING IS TOO LONG. STRING BEGINS *string*
- 103	*literal* IS AN INVALID NUMERIC LITERAL
- 104	ILLEGAL SYMBOL *"token"*. SOME SYMBOLS THAT MIGHT BE LEGAL ARE: *token-list*
- 105	INVALID STRING
- 107	THE NAME *name* IS TOO LONG. MAXIMUM ALLOWABLE SIZE IS *size*
- 108	THE NAME *name* IS QUALIFIED INCORRECTLY
- 109	*clause* CLAUSE IS NOT PERMITTED
- 110	INVALID HEXADECIMAL LITERAL BEGINNING *string*
- 111	A COLUMN FUNCTION DOES NOT INCLUDE A COLUMN NAME
- 112	THE OPERAND OF A COLUMN FUNCTION IS ANOTHER COLUMN FUNCTION
- 113	INVALID CHARACTER FOUND IN *string*, REASON CODE *nnn*
- 114	THE LOCATION NAME *location* DOES NOT MATCH THE CURRENT SERVER
- 115	A PREDICATE IS INVALID BECAUSE THE COMPARISON OPERATOR *operator* IS FOLLOWED BY A PARENTHESIZED LIST OR BY ANY OR ALL WITHOUT A SUBQUERY
- 117	THE NUMBER OF VALUES ASSIGNED IS NOT THE SAME AS THE NUMBER OF SPECIFIED OR IMPLIED COLUMNS
- 118	THE OBJECT TABLE OR VIEW OF THE DELETE OR UPDATE STATEMENT IS ALSO IDENTIFIED IN A FROM CLAUSE
- 119	A COLUMN IDENTIFIED IN A HAVING CLAUSE IS NOT INCLUDED IN THE GROUP BY CLAUSE
- 120	A WHERE CLAUSE, SET CLAUSE, VALUES CLAUSE, OR A SET HOST-VARIABLE STATEMENT INCLUDES A COLUMN FUNCTION
- 121	THE COLUMN *name* IS IDENTIFIED MORE THAN ONCE IN THE INSERT OR UPDATE OR SET TRANSITION VARIABLE STATEMENT
- 122	A SELECT STATEMENT WITH NO GROUP BY CLAUSE CONTAINS A COLUMN NAME AND A COLUMN FUNCTION IN THE SELECT CLAUSE OR A COLUMN NAME IS CONTAINED IN THE SELECT CLAUSE BUT NOT IN THE GROUP BY CLAUSE

- 123	THE PARAMETER IN POSITION *n* IN THE FUNCTION *name* MUST BE A CONSTANT OR KEYWORD
- 125	AN INTEGER IN THE ORDER BY CLAUSE DOES NOT IDENTIFY A COLUMN OF THE RESULT
- 126	THE SELECT STATEMENT CONTAINS BOTH AN UPDATE CLAUSE AND AN ORDER BY CLAUSE
- 127	DISTINCT IS SPECIFIED MORE THAN ONCE IN A SUBSELECT
- 128	INVALID USE OF NULL IN A PREDICATE
- 129	THE STATEMENT CONTAINS TOO MANY TABLE NAMES
- 130	THE ESCAPE CLAUSE CONSISTS OF MORE THAN ONE CHARACTER, OR THE STRING PATTERN CONTAINS AN INVALID OCCURRENCE OF THE ESCAPE CHARACTER
- 131	STATEMENT WITH LIKE PREDICATE HAS INCOMPATIBLE DATA TYPES
- 132	AN OPERAND OF *value* IS NOT VALID
- 133	A COLUMN FUNCTION IN A SUBQUERY OF A HAVING CLAUSE IS INVALID BECAUSE ALL COLUMN REFERENCES IN ITS ARGUMENT ARE NOT CORRELATED TO THE GROUP BY RESULT THAT THE HAVING CLAUSE IS APPLIED TO
- 134	IMPROPER USE OF LONG STRING COLUMN *column-name* OR AN EXPRESSION THAT RESOLVES TO A LONG STRING
- 136	SORT CANNOT BE EXECUTED BECAUSE THE SORT KEY LENGTH IS GREATER THAN 4000 BYTES
- 137	THE LENGTH RESULTING FROM *operation* IS GREATER THAN *maximum-length*
- 138	THE SECOND OR THIRD ARGUMENT OF THE SUBSTR FUNCTION IS OUT OF RANGE
- 142	THE SQL STATEMENT IS NOT SUPPORTED
- 144	INVALID SECTION NUMBER *number*
- 147	ALTER FUNCTION *function-name* FAILED BECAUSE SOURCE FUNCTIONS CANNOT BE ALTERED
- 148	THE SOURCE TABLE *source-name* CANNOT BE RENAMED OR ALTERED
- 150	THE OBJECT OF THE INSERT, DELETE, OR UPDATE STATEMENT IS A VIEW OR TRANSITION TABLE FOR WHICH THE REQUESTED OPERATION IS NOT PERMITTED
- 151	THE UPDATE STATEMENT IS INVALID BECAUSE THE CATALOG DESCRIPTION OF COLUMN *column-name* INDICATES THAT IT CANNOT BE UPDATED
- 152	THE DROP *clause* CLAUSE IN THE ALTER STATEMENT IS INVALID BECAUSE *constraint-name* IS A *constraint-type*
- 153	THE STATEMENT IS INVALID BECAUSE THE VIEW OR TABLE DEFINITION DOES NOT INCLUDE A UNIQUE NAME FOR EACH COLUMN
- 154	THE STATEMENT FAILED BECAUSE VIEW OR TABLE DEFINITION IS NOT VALID
- 156	THE STATEMENT DOES NOT IDENTIFY A TABLE
- 157	ONLY A TABLE NAME CAN BE SPECIFIED IN A FOREIGN KEY CLAUSE. *object-name* IS NOT THE NAME OF A TABLE
- 158	THE NUMBER OF COLUMNS SPECIFIED FOR THE VIEW IS NOT THE SAME AS THE NUMBER OF COLUMNS SPECIFIED BY THE SELECT CLAUSE, OR THE NUMBER OF COLUMNS SPECIFIED IN THE CORRELATION CLAUSE IN A FROM CLAUSE IS NOT THE SAME AS THE NUMBER OF COLUMNS IN THE CORRESPONDING TABLE, VIEW, TABLE EXPRESSION, OR TABLE FUNCTION

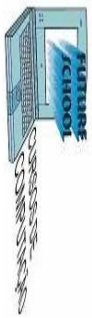


- 159	DROP OR COMMENT ON *object* IDENTIFIES A(N) *object-type1* RATHER THAN A(N) *object-type2*
- 160	THE WITH CHECK OPTION CANNOT BE USED FOR THE SPECIFIED VIEW
- 161	THE INSERT OR UPDATE IS NOT ALLOWED BECAUSE A RESULTING ROW DOES NOT SATISFY THE VIEW DEFINITION
- 164	*auth-id1* DOES NOT HAVE THE PRIVILEGE TO CREATE A VIEW WITH QUALIFICATION *authorization-ID*
- 170	THE NUMBER OF ARGUMENTS SPECIFIED FOR *function-name* IS INVALID
- 171	THE DATA TYPE, LENGTH, OR VALUE OF ARGUMENT *nn* OF *function-name* IS INVALID
- 173	UR IS SPECIFIED ON THE WITH CLAUSE BUT THE CURSOR IS NOT READ-ONLY
- 180	THE DATE, TIME, OR TIMESTAMP VALUE *value* IS INVALID
- 181	THE STRING REPRESENTATION OF A DATETIME VALUE IS NOT A VALID DATETIME VALUE
- 182	AN ARITHMETIC EXPRESSION WITH A DATETIME VALUE IS INVALID
- 183	AN ARITHMETIC OPERATION ON A DATE OR TIMESTAMP HAS A RESULT THAT IS NOT WITHIN THE VALID RANGE OF DATES
- 184	AN ARITHMETIC EXPRESSION WITH A DATETIME VALUE CONTAINS A PARAMETER MARKER
- 185	THE LOCAL FORMAT OPTION HAS BEEN USED WITH A DATE OR TIME AND NO LOCAL EXIT HAS BEEN INSTALLED
- 186	THE LOCAL DATE LENGTH OR LOCAL TIME LENGTH HAS BEEN INCREASED AND EXECUTING PROGRAM RELIES ON THE OLD LENGTH
- 187	A REFERENCE TO A CURRENT DATE/TIME SPECIAL REGISTER IS INVALID BECAUSE THE MVS TOD CLOCK IS BAD OR THE MVS PARMTZ IS OUT OF RANGE
- 188	THE STRING REPRESENTATION OF A NAME IS INVALID
- 189	CCSID *ccsid* IS UNKNOWN OR INVALID FOR THE DATA TYPE OR SUBTYPE
- 190	ATTRIBUTES OF COLUMN *column-name* IN TABLE *table-name* ARE NOT COMPATIBLE WITH THE EXISTING COLUMN
- 191	A STRING CANNOT BE USED BECAUSE IT IS INVALID MIXED DATA
- 197	QUALIFIED COLUMN NAMES IN ORDER BY CLAUSE NOT PERMITTED WHEN UNION OR UNION ALL SPECIFIED
- 198	THE OPERAND OF THE PREPARE OR EXECUTE IMMEDIATE STATEMENT IS BLANK OR EMPTY
- 199	ILLEGAL USE OF KEYWORD *keyword*. TOKEN *token-list* WAS EXPECTED
- 203	A REFERENCE TO COLUMN *column-name* IS AMBIGUOUS
- 204	*name* IS AN UNDEFINED NAME
- 205	*column-name* IS NOT A COLUMN OF TABLE *table-name*
- 206	*column-name* IS NOT A COLUMN OF AN INSERTED TABLE, UPDATED TABLE, OR ANY TABLE IDENTIFIED IN A FROM CLAUSE, OR IS NOT A COLUMN OF THE TRIGGERING TABLE OF A TRIGGER
- 208	THE ORDER BY CLAUSE IS INVALID BECAUSE COLUMN *name* IS NOT PART OF THE RESULT TABLE
- 212	*name* IS SPECIFIED MORE THAN ONCE IN THE REFERENCING CLAUSE OF A TRIGGER DEFINITION
- 214	AN EXPRESSION IN THE FOLLOWING POSITION, OR STARTING WITH *position-or-expression-start* IN THE *clause-type* CLAUSE IS NOT VALID. REASON CODE = *reason-code*

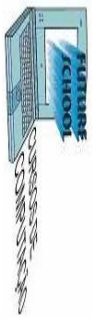


A VALUE CANNOT BE ASSIGNED TO OUTPUT HOST VARIABLE NUMBER *position-number* **BECAUSE THE DATA TYPES ARE NOT COMPARABLE**
A VALUE WITH DATA TYPE *data-type1* **CANNOT BE ASSIGNED TO A HOST VARIABLE BECAUSE THE VALUE IS NOT WITHIN THE RANGE OF THE HOST VARIABLE IN POSITION** *position-number* **WITH DATA TYPE** *data-type2*
[redacted] *position-number* **BECAUSE NO INDICATOR VARIABLE IS SPECIFIED**
A PREDICATE IS INVALID BECAUSE A REFERENCED HOST VARIABLE HAS THE NULL VALUE
DECIMAL HOST VARIABLE OR PARAMETER *number* **CONTAINS NON-DECIMAL DATA**
THE LENGTH OF INPUT HOST VARIABLE NUMBER *position-number* **IS NEGATIVE OR GREATER THAN THE MAXIMUM**
variable-name [redacted] **DEFINITION**
THE NUMBER OF HOST VARIABLES SPECIFIED IS NOT EQUAL TO THE NUMBER OF PARAMETER MARKERS
THE STATEMENT CONTAINS AN AMBIGUOUS HOST VARIABLE REFERENCE

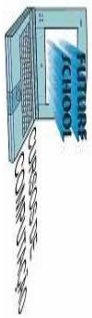
future school
informática



303
304
305
309
310
311
312
313
314

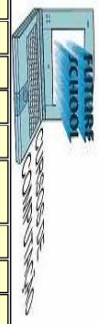


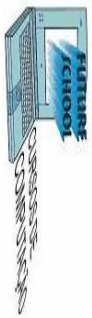
DICATE OPERATOR IS *operator*.



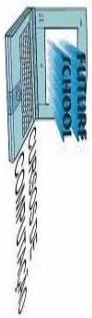
- 327	THE ROW CANNOT BE INSERTED BECAUSE IT IS OUTSIDE THE BOUND OF THE PARTITION RANGE FOR THE LAST PARTITION
- 330	A STRING CANNOT BE USED BECAUSE IT CANNOT BE TRANSLATED. REASON *reason-code*, CHARACTER *code-point*, HOST VARIABLE *position-number*
- 331	A STRING CANNOT BE ASSIGNED TO A HOST VARIABLE BECAUSE IT CANNOT BE TRANSLATED. REASON *reason-code*, CHARACTER *code-point*, POSITION *position-number*
- 332	CHARACTER CONVERSION BETWEEN CCSID *from-ccsid* TO *to-ccsid* REQUESTED BY *reason-code* IS NOT SUPPORTED
- 333	THE SUBTYPE OF A STRING VARIABLE IS NOT THE SAME AS THE SUBTYPE KNOWN AT BIND TIME AND THE DIFFERENCE CANNOT BE RESOLVED BY TRANSLATION
- 338	AN ON CLAUSE IS INVALID
- 339	THE SQL STATEMENT CANNOT BE EXECUTED FROM AN ASCII BASED DRDA APPLICATION REQUESTOR TO A V2R2 DB2 SUBSYSTEM
- 350	INVALID SPECIFICATION OF A LARGE OBJECT COLUMN
- 351	AN UNSUPPORTED SQLTYPE WAS ENCOUNTERED IN POSITION *position-number* OF THE SELECT-LIST
- 352	AN UNSUPPORTED SQLTYPE WAS ENCOUNTERED IN POSITION *position-number* OF THE INPUT-LIST
- 355	A LOB COLUMN IS TOO LARGE TO BE LOGGED
- 359	THE RANGE OF VALUES FOR THE IDENTITY COLUMN IS EXHAUSTED
- 372	ONLY ONE ROWID OR IDENTITY COLUMN IS ALLOWED IN A TABLE
- 373	DEFAULT CANNOT BE SPECIFIED FOR IDENTITY COLUMN *column-name*
- 390	THE FUNCTION *function-name*, SPECIFIC NAME *specific-name*, IS NOT VALID IN THE CONTEXT IN WHICH IT OCCURS
- 392	SQLDA PROVIDED FOR CURSOR *cursor* HAS BEEN CHANGED FROM THE PREVIOUS FETCH
- 396	*object-type object-name* ATTEMPTED TO EXECUTE AN SQL STATEMENT DURING FINAL CALL PROCESSING
- 397	THE OPTION GENERATED IS SPECIFIED WITH A COLUMN THAT IS NOT A ROW ID OR DISTINCT TYPE BASED ON A ROW ID
- 398	A LOCATOR WAS REQUESTED FOR HOST VARIABLE NUMBER *position-number* BUT THE VARIABLE IS NOT A LOB
- 399	ATTEMPTED TO INSERT AN INVALID VALUE INTO A ROWID COLUMN
- 400	THE CATALOG HAS THE MAXIMUM NUMBER OF USER DEFINED INDEXES
- 401	THE OPERANDS OF AN ARITHMETIC OR COMPARISON OPERATION ARE NOT COMPARABLE
- 402	AN ARITHMETIC FUNCTION OR OPERATOR *arith-fop* IS APPLIED TO CHARACTER OR DATETIME DATA
- 404	THE SQL STATEMENT SPECIFIES A STRING THAT IS TOO LONG
- 405	THE NUMERIC LITERAL literal CANNOT BE USED AS SPECIFIED BECAUSE IT IS OUT OF RANGE
- 406	A CALCULATED OR DERIVED NUMERIC VALUE IS NOT WITHIN THE RANGE OF ITS OBJECT COLUMN
- 407	AN UPDATE, INSERT, OR SET VALUE IS NULL, BUT THE OBJECT COLUMN *column-name* CANNOT CONTAIN NULL VALUES
- 408	THE VALUE IS NOT COMPATIBLE WITH THE DATA TYPE OF ITS TARGET
- 409	INVALID OPERAND OF A COUNT FUNCTION

- 410	THE FLOATING POINT LITERAL *literal* CONTAINS MORE THAN 30 CHARACTERS
- 411	CURRENT SQLID CANNOT BE USED IN A STATEMENT THAT REFERENCES REMOTE OBJECTS
- 412	THE SELECT CLAUSE OF A SUBQUERY SPECIFIES MULTIPLE COLUMNS
- 413	OVERFLOW OCCURRED DURING NUMERIC DATA TYPE CONVERSION
- 414	A LIKE PREDICATE IS INVALID BECAUSE THE FIRST OPERAND IS NOT A STRING
- 415	THE CORRESPONDING COLUMNS, *column-number*, OF THE OPERANDS OF A UNION OR A UNION ALL DO NOT HAVE COMPARABLE COLUMN DESCRIPTIONS
- 416	AN OPERAND OF A UNION CONTAINS A LONG STRING COLUMN
- 417	A STATEMENT STRING TO BE PREPARED INCLUDES PARAMETER MARKERS AS THE OPERANDS OF THE SAME OPERATOR
- 418	A STATEMENT STRING TO BE PREPARED CONTAINS AN INVALID USE OF PARAMETER MARKERS
- 419	THE DECIMAL DIVIDE OPERATION IS INVALID BECAUSE THE RESULT WOULD HAVE A NEGATIVE SCALE
- 420	THE VALUE OF A STRING ARGUMENT WAS NOT ACCEPTABLE TO THE *function-name* FUNCTION
- 421	THE OPERANDS OF A UNION OR UNION ALL DO NOT HAVE THE SAME NUMBER OF COLUMNS
- 423	INVALID VALUE FOR LOCATOR IN POSITION *position-#*
- 426	DYNAMIC COMMIT NOT VALID AT AN APPLICATION SERVER WHERE UPDATES ARE NOT ALLOWED
- 427	DYNAMIC ROLLBACK NOT VALID AT AN APPLICATION SERVER WHERE UPDATES ARE NOT ALLOWED
- 430	*routine-type routine-name* (SPECIFIC NAME *specific-name*) HAS ABNORMALLY TERMINATED
- 433	VALUE *value* IS TOO LONG
- 435	AN INVALID SQLSTATE *sqlstate* IS SPECIFIED IN THE FUNCTION RAISE_ERROR OR IN A SIGNAL SQLSTATE STATEMENT
- 438	APPLICATION RAISED ERROR WITH DIAGNOSTIC TEXT: *text*
- 440	NO *routine-type* BY THE NAME *routine-name* HAVING COMPATIBLE ARGUMENTS WAS FOUND
- 441	INVALID USE OF 'DISTINCT' OR 'ALL' WITH SCALAR FUNCTION *function-name*
- 443	EXTERNAL FUNCTION *function-name* (SPECIFIC NAME *specific-name*) HAS RETURNED AN ERROR SQLSTATE WITH DIAGNOSTIC TEXT *msg-text*
- 444	USER PROGRAM *name* COULD NOT BE FOUND
- 449	CREATE OR ALTER STATEMENT FOR FUNCTION OR PROCEDURE *routine-name* CONTAINS AN INVALID FORMAT OF THE EXTERNAL NAME CLAUSE OR IS MISSING THE EXTERNAL NAME CLAUSE
- 450	USER-DEFINED FUNCTION OR STORED PROCEDURE *name*, PARAMETER NUMBER *parmnum*, OVERLAYED STORAGE BEYOND ITS DECLARED LENGTH
- 451	THE *data-item* DEFINITION, IN THE CREATE FUNCTION FOR *function-name* CONTAINS DATA TYPE *type* WHICH IS NOT APPROPRIATE FOR AN EXTERNAL FUNCTION WRITTEN IN THE GIVEN LANGUAGE
- 453	THERE IS A PROBLEM WITH THE RETURNS CLAUSE IN THE CREATE FUNCTION STATEMENT FOR *function-name*



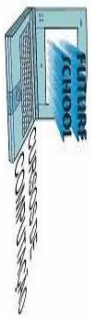


SQL CALL STATEMENT SPECIFIED A NULL VALUE FOR INPUT PARAMETER *number*, **BUT THE STORED PROCEDURE DOES NOT SUPPORT NULL VALUES**
INVOCATION OF FUNCTION OR PROCEDURE *name* **FAILED DUE TO REASON** *rc*
CURSOR *cursor-name* **WAS LEFT OPEN BY EXTERNAL FUNCTION** *function-name* **SPECIFIC NAME** *specific-name*
A USER DEFINED DATA TYPE CANNOT BE CALLED THE SAME NAME AS A SYSTEM PREDEFINED TYPE (BUILT-IN TYPE)
THE RESULT TYPE *type-1* **OF THE SOURCE FUNCTION CANNOT BE CAST TO THE RETURNS TYPE** *type-2* **OF THE USER-DEFINED FUNCTION** *function-name*
REFERENCE TO FUNCTION *function-name* [redacted] **SCHEMA**
DROP OR REVOKE ON OBJECT TYPE *type1* **CANNOT BE PROCESSED BECAUSE OBJECT** *name* **OF TYPE** *type2* **IS DEPENDENT ON IT**
THE PROCEDURE *procedure-name* **HAS NOT YET BEEN CALLED**
THE PROCEDURE *procedure-name* **RETURNED NO LOCATORS**
IN CREATE FUNCTION FOR *function-name* [redacted] **PARAMETERS OF THE SOURCE FUNCTION**
object-type object-name **ATTEMPTED TO EXECUTE AN SQL STATEMENT WHEN THE DEFINITION OF THE FUNCTION OR PROCEDURE DID NOT SPECIFY THIS ACTION**
NUMBER *number* **DIRECTLY SPECIFIED IN AN SQL STATEMENT IS OUTSIDE THE RANGE OF ALLOWABLE VALUES IN THIS CONTEXT (***minval, maxval***)**
CREATE STATEMENT FOR USER-DEFINED FUNCTION *function-name* **MUST HAVE A RETURNS CLAUSE AND: THE EXTERNAL CLAUSE** [redacted]
THE CREATE FUNCTION FOR *function-name* **HAS A PROBLEM WITH PARAMETER NUMBER** *number*, **IT MAY INVOLVE A MISMATCH WITH A**



470
471
472
473
475
476
478
480
482
483
487
490
491
492

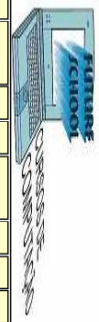
THE SIGNATURE OF SOME OTHER
SPECIFIC NAME DOES NOT MATCH



	SOURCE FUNCTION
- 495	ESTIMATED PROCESSOR COST OF *estimate-amount1* PROCESSOR SECONDS (*estimate-amount2* SERVICE UNITS) IN COST CATEGORY *cost-category* EXCEEDS A RESOURCE LIMIT ERROR THRESHOLD OF *limitamount* SERVICE UNITS
- 496	THE SQL STATEMENT CANNOT BE EXECUTED BECAUSE IT REFERENCES A RESULT SET THAT WAS NOT CREATED BY THE CURRENT SERVER
- 497	THE MAXIMUM LIMIT OF INTERNAL IDENTIFIERS HAS BEEN EXCEEDED FOR DATABASE *database-name*
- 499	CURSOR *cursor-name* HAS ALREADY BEEN ASSIGNED TO THIS OR ANOTHER RESULT SET FROM PROCEDURE *procedure-name*
- 500	THE IDENTIFIED CURSOR WAS CLOSED WHEN THE CONNECTION WAS DESTROYED
- 501	THE CURSOR IDENTIFIED IN A FETCH OR CLOSE STATEMENT IS NOT OPEN
- 502	THE CURSOR IDENTIFIED IN AN OPEN STATEMENT IS ALREADY OPEN
- 503	A COLUMN CANNOT BE UPDATED BECAUSE IT IS NOT IDENTIFIED IN THE UPDATE CLAUSE OF THE SELECT STATEMENT OF THE CURSOR
- 504	THE CURSOR NAME *cursor-name* IS NOT DEFINED
- 507	THE CURSOR IDENTIFIED IN THE UPDATE OR DELETE STATEMENT IS NOT OPEN
- 508	THE CURSOR IDENTIFIED IN THE UPDATE OR DELETE STATEMENT IS NOT POSITIONED ON A ROW
- 509	THE TABLE IDENTIFIED IN THE UPDATE OR DELETE STATEMENT IS NOT THE SAME TABLE DESIGNATED BY THE CURSOR
- 510	THE TABLE DESIGNATED BY THE CURSOR OF THE UPDATE OR DELETE STATEMENT CANNOT BE MODIFIED
- 511	THE FOR UPDATE CLAUSE CANNOT BE SPECIFIED BECAUSE THE TABLE DESIGNATED BY THE CURSOR CANNOT BE MODIFIED
- 512	STATEMENT REFERENCE TO REMOTE OBJECT IS INVALID
- 513	THE ALIAS *alias-name* MUST NOT BE DEFINED ON ANOTHER LOCAL OR REMOTE ALIAS
- 514	THE CURSOR *cursor-name* IS NOT IN A PREPARED STATE
- 516	THE DESCRIBE FOR STATIC STATEMENT DOES NOT IDENTIFY A PREPARED STATEMENT
- 517	CURSOR *cursor-name* CANNOT BE USED BECAUSE ITS STATEMENT NAME DOES NOT IDENTIFY A PREPARED SELECT STATEMENT
- 518	THE EXECUTE STATEMENT DOES NOT IDENTIFY A VALID PREPARED STATEMENT
- 519	THE PREPARE STATEMENT IDENTIFIES THE SELECT STATEMENT OF THE OPENED CURSOR *cursor-name*
- 525	THE SQL STATEMENT CANNOT BE EXECUTED BECAUSE IT WAS IN ERROR AT BIND TIME FOR SECTION = *sectno* PACKAGE = *pkgname* CONSISTENCY TOKEN = X'contoken'
- 526	THE REQUESTED OPERATION OR USAGE DOES NOT APPLY TO *table type* TEMPORARY TABLE *table name*
- 530	THE INSERT OR UPDATE VALUE OF FOREIGN KEY *constraint-name* IS INVALID
- 531	PARENT KEY IN A PARENT ROW CANNOT BE UPDATED BECAUSE IT HAS ONE OR MORE DEPENDENT ROWS IN RELATIONSHIP *constraint-name*
- 532	THE RELATIONSHIP *constraint-name* RESTRICTS THE DELETION OF ROW WITH RID X'rid-number'

- 533	**INVALID MULTIPLE-ROW INSERT**
- 534	**THE PRIMARY KEY CANNOT BE UPDATED BECAUSE OF MULTIPLE-ROW UPDATE**
- 536	**THE DELETE STATEMENT IS INVALID BECAUSE TABLE** *table-name* **CAN BE AFFECTED BY THE OPERATION**
- 537	**THE PRIMARY KEY CLAUSE, A FOREIGN KEY CLAUSE, OR A UNIQUE CLAUSE IDENTIFIES COLUMN** *column-name* **MORE THAN ONCE**
- 538	**FOREIGN KEY** *name* **DOES NOT CONFORM TO THE DESCRIPTION OF A PARENT KEY OF TABLE** *table-name*
- 539	**TABLE** *table-name* **DOES NOT HAVE A PRIMARY KEY**
- 540	**THE DEFINITION OF TABLE** *table-name* **IS INCOMPLETE BECAUSE IT LACKS A PRIMARY INDEX OR A REQUIRED UNIQUE INDEX**
- 542	*column-name* **CANNOT BE A COLUMN OF A PRIMARY KEY, A UNIQUE CONSTRAINT, OR A PARENT KEY BECAUSE IT CAN CONTAIN NULL VALUES**
- 543	**A ROW IN A PARENT TABLE CANNOT BE DELETED BECAUSE THE CHECK CONSTRAINT** *check-constraint* **RESTRICTS THE DELETION**
- 544	**THE CHECK CONSTRAINT SPECIFIED IN THE ALTER TABLE STATEMENT CANNOT BE ADDED BECAUSE AN EXISTING ROW VIOLATES THE CHECK CONSTRAINT**
- 545	**THE REQUESTED OPERATION IS NOT ALLOWED BECAUSE A ROW DOES NOT SATISFY THE CHECK CONSTRAINT** *check-constraint*
- 546	**THE CHECK CONSTRAINT** *constraint-name* **IS INVALID**
- 548	**A CHECK CONSTRAINT THAT IS DEFINED WITH** *column-name* **IS INVALID**
- 549	**THE** *statement* **STATEMENT IS NOT ALLOWED FOR** *object_type1 object_name* **BECAUSE THE BIND OPTION DYNAMICRULES(RUN) IS NOT IN EFFECT FOR** *object_type2*
- 551	*auth-id* **DOES NOT HAVE THE PRIVILEGE TO PERFORM OPERATION** *operation* **ON OBJECT** *object-name*
- 552	*auth-id* **DOES NOT HAVE THE PRIVILEGE TO PERFORM OPERATION** *operation*
- 553	*auth-id* **SPECIFIED IS NOT ONE OF THE VALID AUTHORIZATION IDS**
- 554	**AN AUTHORIZATION ID CANNOT GRANT A PRIVILEGE TO ITSELF**
- 555	**AN AUTHORIZATION ID CANNOT REVOKE A PRIVILEGE FROM ITSELF**
- 556	*authid2* **CANNOT HAVE THE** *privilege* **PRIVILEGE** *on_object* **REVOKED BY** *authid1* **BECAUSE THE REVOKEE DOES NOT POSSESS THE PRIVILEGE OR THE REVOKER DID NOT MAKE THE GRANT**
- 557	**INCONSISTENT GRANT/REVOKE KEYWORD** *keyword*. **PERMITTED KEYWORDS ARE** *keyword-list*
- 558	**INVALID CLAUSE OR COMBINATION OF CLAUSES ON A GRANT OR REVOKE**
- 559	**ALL AUTHORIZATION FUNCTIONS HAVE BEEN DISABLED**
- 567	*bind-type* **AUTHORIZATION ERROR USING** *auth-id* **AUTHORITY PACKAGE =** *package-name* **PRIVILEGE =** *privilege*
- 571	**THE STATEMENT WOULD RESULT IN A MULTIPLE SITE UPDATE**
- 573	**TABLE** *table-name* **DOES NOT HAVE A UNIQUE KEY WITH THE SPECIFIED COLUMN NAMES**
- 574	**THE SPECIFIED DEFAULT VALUE OR IDENTITY ATTRIBUTE VALUE CONFLICTS WITH THE DEFINITION OF COLUMN** *column-name*
- 577	*object-type object-name* **ATTEMPTED TO MODIFY DATA WHEN THE DEFINITION OF THE FUNCTION OR PROCEDURE DID NOT SPECIFY**

	THIS ACTION
- 579	*object-type object-name* **ATTEMPTED TO READ DATA WHEN THE DEFINITION OF THE FUNCTION OR PROCEDURE DID NOT SPECIFY THIS ACTION**
- 580	**THE RESULT-EXPRESSIONS OF A CASE EXPRESSION CANNOT ALL BE NULL**
- 581	**THE DATA TYPES OF THE RESULT-EXPRESSIONS OF A CASE EXPRESSION ARE NOT COMPATIBLE**
- 582	**THE SEARCH-CONDITION IN A SEARCHED-WHEN-CLAUSE CANNOT BE A QUANTIFIED PREDICATE, IN PREDICATE, OR AN EXISTS PREDICATE**
- 583	**THE USE OF FUNCTION** *function-name* **IS INVALID BECAUSE IT IS NOT DETERMINISTIC OR HAS AN EXTERNAL ACTION**
- 585	**THE SCHEMA NAME** *schema-name* **CANNOT APPEAR MORE THAN ONCE IN THE CURRENT PATH**
- 586	**THE TOTAL LENGTH OF THE CURRENT PATH SPECIAL REGISTER CANNOT EXCEED 254 CHARACTERS**
- 587	**A LIST OF** *item-references* **ARE NOT IN THE SAME FAMILY**
- 590	**PARAMETER NAME** *parameter-name* **IS NOT UNIQUE IN THE CREATE FOR ROUTINE** *routine-name*
- 592	**NOT AUTHORIZED TO CREATE FUNCTIONS OR PROCEDURES IN WLM ENVIRONMENT** *env-name*
- 593	**NOT NULL MUST BE SPECIFIED FOR ROWID OR DISTINCT TYPE COLUMN** *column-name*
- 601	**THE NAME OF THE OBJECT TO BE CREATED OR THE TARGET OF A RENAME STATEMENT IS IDENTICAL TO THE EXISTING NAME** *name* **OF THE OBJECT TYPE** *obj-type*
- 602	**TOO MANY COLUMNS SPECIFIED IN A CREATE INDEX**
- 603	**A UNIQUE INDEX CANNOT BE CREATED BECAUSE THE TABLE CONTAINS ROWS WHICH ARE DUPLICATES WITH RESPECT TO THE VALUES OF THE IDENTIFIED COLUMNS**
- 604	**A DATA TYPE DEFINITION SPECIFIES AN INVALID LENGTH, PRECISION, OR SCALE ATTRIBUTE**
- 607	**OPERATION OR OPTION** *operation* **IS NOT DEFINED FOR THIS OBJECT**
- 611	**ONLY LOCKMAX 0 CAN BE SPECIFIED WHEN THE LOCK SIZE OF THE TABLESPACE IS TABLESPACE OR TABLE**
- 612	*column-name* **IS A DUPLICATE COLUMN NAME**
- 613	**THE PRIMARY KEY OR A UNIQUE CONSTRAINT IS TOO LONG OR HAS TOO MANY COLUMNS**
- 614	**THE INDEX CANNOT BE CREATED OR THE LENGTH OF A COLUMN CANNOT BE CHANGED BECAUSE THE SUM OF THE INTERNAL LENGTHS OF THE IDENTIFIED COLUMNS IS GREATER THAN THE ALLOWABLE MAXIMUM**
- 615	*operation-type* **IS NOT ALLOWED ON A PACKAGE IN USE**
- 616	*obj-type1 obj-name1* **CANNOT BE DROPPED BECAUSE IT IS REFERENCED BY** *obj-type2 obj-name2*
- 617	**A TYPE 1 INDEX IS NOT VALID FOR TABLE** *table-name*
- 618	**OPERATION** *operation* **IS NOT ALLOWED ON SYSTEM DATABASES**
- 619	**OPERATION DISALLOWED BECAUSE THE DATABASE IS NOT STOPPED**
- 620	**KEYWORD** *keyword* **IN** *stmt type* **STATEMENT IS NOT PERMITTED FOR A** *space type* **SPACE IN THE** *database type* **DATABASE**



- 621	DUPLICATE DBID *dbid* WAS DETECTED AND PREVIOUSLY ASSIGNED TO *database-name*
- 622	*for mixed data* IS INVALID BECAUSE THE MIXED DATA INSTALL OPTION IS NO
- 623	A CLUSTERING INDEX ALREADY EXISTS ON TABLE *table-name*
- 624	TABLE *table-name* ALREADY HAS A PRIMARY KEY OR UNIQUE KEY CONSTRAINT WITH SPECIFIED COLUMNS
- 625	TABLE *table-name* DOES NOT HAVE AN INDEX TO ENFORCE THE UNIQUENESS OF THE PRIMARY OR UNIQUE KEY
- 626	THE ALTER STATEMENT IS NOT EXECUTABLE BECAUSE THE PAGE SET IS NOT STOPPED
- 627	THE ALTER STATEMENT IS INVALID BECAUSE THE PAGESET HAS USER-MANAGED DATA SETS
- 628	THE CLAUSES ARE MUTUALLY EXCLUSIVE
- 629	SET NULL CANNOT BE SPECIFIED BECAUSE FOREIGN KEY *name* CANNOT CONTAIN NULL VALUES
- 630	THE WHERE NOT NULL SPECIFICATION IS INVALID FOR TYPE 1 INDEXES
- 631	FOREIGN KEY *name* IS TOO LONG OR HAS TOO MANY COLUMNS
- 632	THE TABLE CANNOT BE DEFINED AS A DEPENDENT OF *table-name* BECAUSE OF DELETE RULE RESTRICTIONS
- 633	THE DELETE RULE MUST BE *delete-rule*
- 634	THE DELETE RULE MUST NOT BE CASCADE
- 635	THE DELETE RULES CANNOT BE DIFFERENT OR CANNOT BE SET NULL
- 636	THE PARTITIONING KEYS FOR PARTITION *part-num* ARE NOT SPECIFIED IN ASCENDING OR DESCENDING ORDER
- 637	DUPLICATE *keyword* KEYWORD
- 638	TABLE *table-name* CANNOT BE CREATED BECAUSE COLUMN DEFINITION IS MISSING
- 639	A NULLABLE COLUMN OF A FOREIGN KEY WITH A DELETE RULE OF SET NULL CANNOT BE A COLUMN OF THE KEY OF A PARTITIONED INDEX
- 640	LOCKSIZE ROW CANNOT BE SPECIFIED BECAUSE TABLE IN THIS TABLESPACE HAS TYPE 1 INDEX
- 643	CHECK CONSTRAINT EXCEEDS MAXIMUM ALLOWABLE LENGTH
- 644	INVALID VALUE SPECIFIED FOR KEYWORD *keyword* IN *stmt-type* STATEMENT
- 646	TABLE *table-name* CANNOT BE CREATED IN SPECIFIED TABLE SPACE *table-space-name* BECAUSE IT ALREADY CONTAINS A TABLE
- 647	BUFFERPOOL *bp-name* CANNOT BE SPECIFIED BECAUSE IT HAS NOT BEEN ACTIVATED
- 650	THE ALTER INDEX CANNOT BE EXECUTED, REASON *reason*
- 651	TABLE DESCRIPTION EXCEEDS MAXIMUM SIZE OF OBJECT DESCRIPTOR
- 652	VIOLATION OF INSTALLATION DEFINED EDIT OR VALIDATION PROCEDURE *proc-name*
- 653	TABLE *table-name* IN PARTITIONED TABLE SPACE *tspace-name* IS NOT AVAILABLE BECAUSE ITS PARTITIONED INDEX HAS NOT BEEN CREATED
- 655	THE CREATE OR ALTER STOGROUP IS INVALID BECAUSE THE STORAGE GROUP WOULD HAVE BOTH SPECIFIC AND NON-SPECIFIC VOLUME IDS

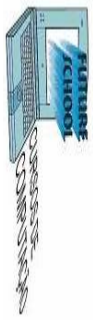
- 658	A *object-type* CANNOT BE DROPPED USING THE *statement* STATEMENT
- 660	INDEX *index-name* CANNOT BE CREATED OR ALTERED ON PARTITIONED TABLE SPACE *tspace-name* BECAUSE KEY LIMITS ARE NOT SPECIFIED
- 661	INDEX *index-name* CANNOT BE CREATED ON PARTITIONED TABLE SPACE *tspace-name* BECAUSE THE NUMBER OF PART SPECIFICATIONS IS NOT EQUAL TO THE NUMBER OF PARTITIONS OF THE TABLE SPACE
- 662	A PARTITIONED INDEX CANNOT BE CREATED ON A NON-PARTITIONED TABLE SPACE *tspace-name*
- 663	THE NUMBER OF KEY LIMIT VALUES IS EITHER ZERO, OR GREATER THAN THE NUMBER OF COLUMNS IN THE KEY OF INDEX *index-name*
- 665	THE PART CLAUSE OF AN ALTER STATEMENT IS OMITTED OR INVALID
- 666	*stmt-verb object* CANNOT BE EXECUTED BECAUSE *function* IS IN PROGRESS
- 667	THE CLUSTERING INDEX FOR A PARTITIONED TABLE SPACE CANNOT BE EXPLICITLY DROPPED
- 668	THE COLUMN CANNOT BE ADDED TO THE TABLE BECAUSE THE TABLE HAS AN EDIT PROCEDURE
- 669	THE OBJECT CANNOT BE EXPLICITLY DROPPED. REASON *reason-code*
- 670	THE RECORD LENGTH OF THE TABLE EXCEEDS THE PAGE SIZE LIMIT
- 671	THE BUFFERPOOL ATTRIBUTE OF THE TABLE SPACE CANNOT BE ALTERED AS SPECIFIED BECAUSE IT WOULD CHANGE THE PAGE SIZE OF THE TABLE SPACE
- 672	OPERATION DROP NOT ALLOWED ON TABLE *table_name*
- 676	ONLY A 4K PAGE BUFFERPOOL CAN BE USED FOR AN INDEX
- 677	INSUFFICIENT VIRTUAL STORAGE FOR BUFFERPOOL EXPANSION
- 678	THE LITERAL literal SPECIFIED FOR THE INDEX LIMIT KEY MUST CONFORM TO THE DATA TYPE *data-type* OF THE CORRESPONDING COLUMN *column-name*
- 679	THE OBJECT *name* CANNOT BE CREATED BECAUSE A DROP IS PENDING ON THE OBJECT
- 680	TOO MANY COLUMNS SPECIFIED FOR A TABLE, VIEW OR TABLE FUNCTION
- 681	COLUMN *column-name* IN VIOLATION OF INSTALLATION DEFINED FIELD PROCEDURE. RT: *return-code*, RS: *reason-code*, MSG: *message-token*
- 682	FIELD PROCEDURE *procedure-name* COULD NOT BE LOADED
- 683	THE SPECIFICATION FOR COLUMN, DISTINCT TYPE, FUNCTION, OR PROCEDURE *data-item* CONTAINS INCOMPATIBLE CLAUSES
- 684	THE LENGTH OF LITERAL LIST BEGINNING *string* IS TOO LONG
- 685	INVALID FIELD TYPE, *column-name*
- 686	COLUMN DEFINED WITH A FIELD PROCEDURE CAN NOT COMPARE WITH ANOTHER COLUMN WITH DIFFERENT FIELD PROCEDURE
- 687	FIELD TYPES INCOMPARABLE
- 688	INCORRECT DATA RETURNED FROM FIELD PROCEDURE, *column-name*, *msgno*

- 689	**TOO MANY COLUMNS DEFINED FOR A DEPENDENT TABLE**
- 690	**THE STATEMENT IS REJECTED BY DATA DEFINITION CONTROL SUPPORT. REASON** *reason-code*
- 691	**THE REQUIRED REGISTRATION TABLE** *table-name* **DOES NOT EXIST**
- 692	**THE REQUIRED UNIQUE INDEX** *index-name* **FOR DDL REGISTRATION TABLE** *table-name* **DOES NOT EXIST**
- 693	**THE COLUMN** *column-name* **IN DDL REGISTRATION TABLE OR INDEX** *table-name (index-name)* **IS NOT DEFINED PROPERLY**
- 694	**THE DDL STATEMENT CANNOT BE EXECUTED BECAUSE A DROP IS PENDING ON THE DDL REGISTRATION TABLE** *table-name*
- 696	**THE DEFINITION OF TRIGGER** *trigger-name* **INCLUDES AN INVALID USE OF CORRELATION NAME OR TRANSITION TABLE NAME** *name*. **REASON CODE=***reason-code*
- 697	**OLD OR NEW CORRELATION NAMES ARE NOT ALLOWED IN A TRIGGER DEFINED WITH THE FOR EACH STATEMENT CLAUSE. OLD_TABLE OR NEW_TABLE NAMES ARE NOT ALLOWED IN A TRIGGER WITH THE BEFORE CLAUSE**
- 713	**THE REPLACEMENT VALUE** *value* **FOR** *special-register* **IS INVALID**
- 715	**PROGRAM** *program-name* **WITH MARK** *release-dependency-mark* **FAILED BECAUSE IT DEPENDS ON FUNCTIONS OF THE RELEASE FROM WHICH FALLBACK HAS OCCURRED**
- 716	**PROGRAM** *program-name* **PRECOMPILED WITH INCORRECT LEVEL FOR THIS RELEASE**
- 717	*bind-type* **FOR** *object-type object-name* **WITH MARK** *release-dependency-mark* **FAILED BECAUSE** *object-type* **DEPENDS ON FUNCTIONS OF THE RELEASE FROM WHICH FALLBACK HAS OCCURRED**
- 718	**REBIND OF PACKAGE** *package-name* **FAILED BECAUSE IBMREQD OF** *ibmreqd* **IS INVALID**
- 719	**BIND ADD ERROR USING** *auth-id* **AUTHORITY PACKAGE** *package-name* **ALREADY EXISTS**
- 720	**BIND ERROR, ATTEMPTING TO REPLACE PACKAGE =** *package_name* **WITH VERSION =** *version2* **BUT THIS VERSION ALREADY EXISTS**
- 721	**BIND ERROR FOR PACKAGE =** *pkg-id* **CONTOKEN =** *contoken'X IS NOT UNIQUE SO IT CANNOT BE CREATED*
- 722	*bind-type* **ERROR USING** *auth-id* **AUTHORITY PACKAGE** *package-name* **DOES NOT EXIST**
- 723	**AN ERROR OCCURRED IN A TRIGGERED SQL STATEMENT IN TRIGGER** *trigger-name*, **SECTION NUMBER** *section-number*. **INFORMATION RETURNED: SQLCODE** *sqlerror*, **SQLSTATE** *sqlstate*, **AND MESSAGE TOKENS** *token-list*
- 724	**THE ACTIVATION OF THE** *object-type* **OBJECT** *object-name* **WOULD EXCEED THE MAXIMUM LEVEL OF INDIRECT SQL CASCADING**
- 725	**THE SPECIAL REGISTER** *register* **AT LOCATION** *location* **WAS SUPPLIED AN INVALID VALUE**
- 726	**BIND ERROR ATTEMPTING TO REPLACE PACKAGE =** *package-name*. **THERE ARE ENABLE OR DISABLE ENTRIES CURRENTLY ASSOCIATED WITH THE PACKAGE**
- 728	**DATA TYPE** *data-type* **IS NOT ALLOWED IN DB2 PRIVATE PROTOCOL PROCESSING**
- 729	**A STORED PROCEDURE SPECIFYING COMMIT ON RETURN CANNOT BE THE TARGET OF A NESTED CALL STATEMENT**
- 730	**THE PARENT OF A TABLE IN A READ-ONLY SHARED DATABASE MUST ALSO BE A TABLE IN A READ-ONLY SHARED DATABASE**
- 731	**USER-DEFINED DATASET** *dsname* **MUST BE DEFINED WITH SHAREOPTIONS(1,3)**
- 732	**THE DATABASE IS DEFINED ON THIS SUBSYSTEM WITH THE ROSHARE READ ATTRIBUTE BUT THE TABLE SPACE OR INDEX SPACE HAS NOT BEEN DEFINED ON THE OWNING SUBSYSTEM**

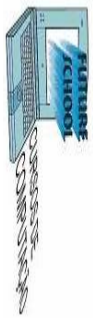
- 733	THE DESCRIPTION OF A TABLE SPACE, INDEX SPACE, OR TABLE IN A ROSHARE READ DATABASE MUST BE CONSISTENT WITH ITS DESCRIPTION IN THE OWNER SYSTEM
- 734	THE ROSHARE ATTRIBUTE OF A DATABASE CANNOT BE ALTERED FROM ROSHARE READ
- 735	DATABASE dbid CANNOT BE ACCESSED BECAUSE IT IS NO LONGER A SHARED DATABASE
- 736	INVALID OBID *obid* SPECIFIED
- 737	IMPLICIT TABLE SPACE NOT ALLOWED
- 739	CREATE OR ALTER FUNCTION *function-name* FAILED BECAUSE FUNCTIONS CANNOT MODIFY DATA WHEN THEY ARE PROCESSED IN PARALLEL.
- 740	FUNCTION *name* IS DEFINED WITH THE OPTION MODIFIES SQL DATA WHICH IS NOT VALID IN THE CONTEXT IN WHICH IT WAS INVOKED
- 741	A *database-type* DATABASE IS ALREADY DEFINED FOR MEMBER *member-name*
- 742	DSNDB07 IS THE IMPLICIT WORK FILE DATABASE
- 746	THE SQL STATEMENT IN AN EXTERNAL FUNCTION, TRIGGER, OR IN STORED PROCEDURE *name* VIOLATES THE NESTING SQL RESTRICTION
- 747	TABLE *table-name* IS NOT AVAILABLE UNTIL THE AUXILIARY TABLES AND INDEXES FOR ITS EXTERNALLY STORED COLUMNS HAVE BEEN CREATED
- 748	AN INDEX ALREADY EXISTS ON AUXILIARY TABLE *table-name*
- 750	THE SOURCE TABLE *source-name* CANNOT BE RENAMED BECAUSE IT IS REFERENCED IN EXISTING VIEW DEFINITIONS OR TRIGGER DEFINITIONS
- 751	*object-type object-name* (SPECIFIC NAME *specific name*) ATTEMPTED TO EXECUTE AN SQL STATEMENT *statement* THAT IS NOT ALLOWED
- 752	THE CONNECT STATEMENT IS INVALID BECAUSE THE PROCESS IS NOT IN THE CONNECTABLE STATE
- 763	INVALID TABLE SPACE NAME *table-space-name*
- 764	A LOB TABLE SPACE AND ITS ASSOCIATED BASE TABLE SPACE MUST BE IN THE SAME DATABASE
- 765	TABLE IS NOT COMPATIBLE WITH DATABASE
- 766	THE OBJECT OF A STATEMENT IS AN AUXILIARY TABLE FOR WHICH THE REQUESTED OPERATION IS NOT PERMITTED
- 767	MISSING OR INVALID COLUMN SPECIFICATION FOR INDEX *index-name*
- 768	AN AUXILIARY TABLE ALREADY EXISTS FOR THE SPECIFIED COLUMN OR PARTITION
- 769	SPECIFICATION OF CREATE AUX TABLE DOES NOT MATCH THE CHARACTERISTICS OF THE BASE TABLE
- 770	TABLE *table-name* CANNOT HAVE A LOB COLUMN UNLESS IT ALSO HAS A ROWID COLUMN
- 771	INVALID SPECIFICATION OF A ROWID COLUMN
- 797	ATTEMPT to CREATE TRIGGER *trigger-name* WITH AN UNSUPPORTED TRIGGERED SQL STATEMENT
- 798	YOU CANNOT INSERT A VALUE INTO A COLUMN THAT IS DEFINED WITH THE OPTION GENERATED ALWAYS COLUMN *column-name*
- 802	EXCEPTION ERROR *exception-type* HAS OCCURRED DURING *operation-type* OPERATION ON *data-type* DATA, POSITION *position-number*

- 803	AN INSERTED OR UPDATED VALUE IS INVALID BECAUSE THE INDEX IN INDEX SPACE *indexspace-name* CONSTRAINS COLUMNS OF THE TABLE SO NO TWO ROWS CAN CONTAIN DUPLICATE VALUES IN THOSE COLUMNS. RID OF EXISTING ROW IS X*rid*
- 804	AN ERROR WAS FOUND IN THE APPLICATION PROGRAM INPUT PARAMETERS FOR THE SQL STATEMENT, REASON *reason*
- 805	DBRM OR PACKAGE NAME *location-name.collection-id.dbrmname. consistency -token* NOT FOUND IN PLAN *plan-name*. REASON *reason*
- 807	ACCESS DENIED: PACKAGE *package-name* IS NOT ENABLED FOR ACCESS FROM *connection-type connection-name*
- 808	THE CONNECT STATEMENT IS NOT CONSISTENT WITH THE FIRST CONNECT STATEMENT
- 811	THE RESULT OF AN EMBEDDED SELECT STATEMENT OR A SUBSELECT IN THE SET CLAUSE OF AN UPDATE STATEMENT IS A TABLE OF MORE THAN ONE ROW, OR THE RESULT OF A SUBQUERY OF A BASIC PREDICATE IS MORE THAN ONE VALUE
- 812	THE SQL STATEMENT CANNOT BE PROCESSED BECAUSE A BLANK COLLECTION-ID WAS FOUND IN THE CURRENT PACKAGESET SPECIAL REGISTER WHILE TRYING TO FORM A QUALIFIED PACKAGE NAME FOR PROGRAM *program-name.consistencytoken* USING PLAN *plan-name*
- 815	A GROUP BY OR HAVING CLAUSE IS IMPLICITLY OR EXPLICITLY SPECIFIED IN A SUBSELECT OF A BASIC PREDICATE OR THE SET CLAUSE OF AN UPDATE STATEMENT
- 817	THE SQL STATEMENT CANNOT BE EXECUTED BECAUSE THE STATEMENT WILL RESULT IN A PROHIBITED UPDATE OPERATION
- 818	THE PRECOMPILER-GENERATED TIMESTAMP *x* IN THE LOAD MODULE IS DIFFERENT FROM THE BIND TIMESTAMP *y* BUILT FROM THE DBRM *z*
- 819	THE VIEW CANNOT BE PROCESSED BECAUSE THE LENGTH OF ITS PARSE TREE IN THE CATALOG IS ZERO
- 820	THE SQL STATEMENT CANNOT BE PROCESSED BECAUSE *catalog-table* CONTAINS A VALUE THAT IS NOT VALID IN THIS RELEASE
- 822	THE SQLDA CONTAINS AN INVALID DATA ADDRESS OR INDICATOR VARIABLE ADDRESS
- 840	TOO MANY ITEMS RETURNED IN A SELECT OR INSERT LIST
- 842	A CONNECTION TO *location-name* ALREADY EXISTS
- 843	THE SET CONNECTION OR RELEASE STATEMENT MUST SPECIFY AN EXISTING CONNECTION
- 846	INVALID SPECIFICATION OF AN IDENTITY COLUMN
- 867	INVALID SPECIFICATION OF A ROWID COLUMN
- 870	THE NUMBER OF HOST VARIABLES IN THE STATEMENT IS NOT EQUAL TO THE NUMBER OF DESCRIPTORS
- 872	A VALID CCSID HAS NOT YET BEEN SPECIFIED FOR THIS SUBSYSTEM
- 873	DATA ENCODED WITH DIFFERENT CCSIDS CANNOT BE REFERENCED IN THE SAME SQL STATEMENT
- 874	THE ENCODING SCHEME SPECIFIED FOR THE TABLE IS NOT THE SAME AS THAT USED FOR THE TABLE SPACE CONTAINING THIS TABLE
- 875	*operand* CANNOT BE USED WITH THE ASCII DATA REFERENCED
- 876	'*object*' CANNOT BE CREATED, REASON '*reason*'
- 877	CCSID ASCII OR CCSID UNICODE IS NOT ALLOWED FOR THIS DATABASE OR TABLE SPACE

- 878	THE PLAN_TABLE USED FOR EXPLAIN CANNOT BE ASCII OR UNICODE
- 879	CREATE or ALTER STATEMENT FOR *obj-name* CANNOT DEFINE A COLUMN, DISTINCT TYPE, FUNCTION OR STORED PROCEDURE PARAMETER AS MIXED OR GRAPHIC WITH ENCODING SCHEME *encoding-scheme*
- 880	SAVEPOINT *savepoint-name* DOES NOT EXIST OR IS INVALID IN THIS CONTEXT
- 881	A SAVEPOINT WITH NAME *savepoint-name* ALREADY EXISTS, BUT THIS SAVEPOINT NAME CANNOT BE REUSED
- 882	SAVEPOINT DOES NOT EXIST
- 900	THE SQL STATEMENT CANNOT BE EXECUTED BECAUSE THE APPLICATION PROCESS IS NOT CONNECTED TO AN APPLICATION SERVER
- 901	UNSUCCESSFUL EXECUTION CAUSED BY A SYSTEM ERROR THAT DOES NOT PRECLUDE THE SUCCESSFUL EXECUTION OF SUBSEQUENT SQL STATEMENTS
- 902	POINTER TO THE ESSENTIAL CONTROL BLOCK (CT/RDA) HAS VALUE 0, REBIND REQUIRED
- 904	UNSUCCESSFUL EXECUTION CAUSED BY AN UNAVAILABLE RESOURCE. REASON *reason-code*, TYPE OF RESOURCE *resource-type*, AND RESOURCE NAME *resource-name*
- 905	UNSUCCESSFUL EXECUTION DUE TO RESOURCE LIMIT BEING EXCEEDED, RESOURCE NAME = *resource-name* LIMIT = *limit-amount1* CPU SECONDS (*limit-amount2* SERVICE UNITS) DERIVED FROM *limit-source*
- 906	THE SQL STATEMENT CANNOT BE EXECUTED BECAUSE THIS FUNCTION IS DISABLED DUE TO A PRIOR ERROR
- 908	*bind-type* ERROR USING *auth-id* AUTHORITY. BIND, REBIND OR AUTO-REBIND OPERATION IS NOT ALLOWED
- 909	THE OBJECT HAS BEEN DELETED
- 910	THE SQL STATEMENT CANNOT ACCESS AN OBJECT ON WHICH A DROP OR ALTER IS PENDING
- 911	THE CURRENT UNIT OF WORK HAS BEEN ROLLED BACK DUE TO DEADLOCK OR TIMEOUT. REASON *reason-code*, TYPE OF RESOURCE *resource-type*, AND RESOURCE NAME *resource-name*
- 913	UNSUCCESSFUL EXECUTION CAUSED BY DEADLOCK OR TIMEOUT. REASON CODE *reason-code*, TYPE OF RESOURCE *resource-type*, AND RESOURCE NAME *resource-name*
- 917	BIND PACKAGE FAILED
- 918	THE SQL STATEMENT CANNOT BE EXECUTED BECAUSE A CONNECTION HAS BEEN LOST
- 919	A ROLLBACK OPERATION IS REQUIRED
- 922	AUTHORIZATION FAILURE: *error-type* ERROR. REASON *reason-code*
- 923	CONNECTION NOT ESTABLISHED: DB2 *condition* REASON *reason-code*, TYPE *resource-type*, NAME *resource-name*
- 924	DB2 CONNECTION INTERNAL ERROR, *function-code, return-code, reason-code*
- 925	COMMIT NOT VALID IN IMS, CICS OR RRSAF ENVIRONMENT
- 926	ROLLBACK NOT VALID IN IMS, CICS OR RRSAF ENVIRONMENT
- 927	THE LANGUAGE INTERFACE (LI) WAS CALLED WHEN THE CONNECTING ENVIRONMENT WAS NOT ESTABLISHED. THE PROGRAM SHOULD BE INVOKED UNDER THE DSN COMMAND



CHANGES, BUT THE DATA CANNOT
COMMUNICATIONS DATABASE
WS SQL OPERATIONS, REASON
2= *rc2*



DESCRIPTION
Remove the expression from the ORDER BY clause. If attempting to reference a column of the result, change the sort key to the *simple-integer* or *simple column-name* form. See the ORDER BY syntax diagram in the DB2 SQL Reference for more information
Remove DISTINCT from the select clause

Códigos complementares para SQLCODE = -330 (reason-codes)

DESCRIPTION
Length exception (for example, expansion required for PC MIXED data exceeds the maximum length of the string)
Invalid code point (for example, use of the ERRORBYTE option of SYSSTRINGS)
Form exception (for example, invalid MIXED data)
Translate procedure error (for example, an exit set the length control field of the string to an invalid value)
SBCS character found in string contained in a wchar_t host variable. If the *reason-code* is 12, *code-point* is the invalid code point. Otherwise, *code-point* is either blank or an additional *reason-code* returned by an exit. If the string is t he value of an input host variable, the *position-number* is the ordinality of the variable in the SQLDA. If the string is not the value of a host variable, the *position-number* is blank

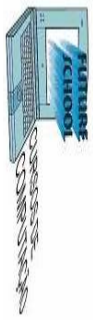
Códigos complementares para SQLCODE = -331 (reason-codes)

DESCRIPTION
for length exception (e.g., expansion required for PC MIXED data exceeds the maximum length of the string)

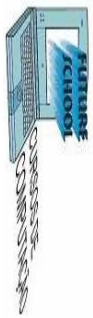
CODE
1
2

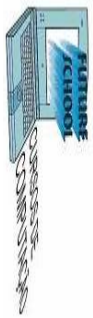
CODE
8
12
16
20
24

CODE
8

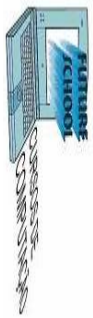


CS identifier is allowed.)



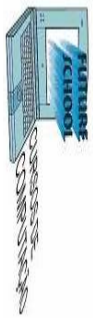


orce the primary key, unique key, or referential constraint of the table

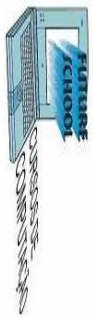


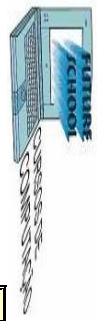
CODE	DESCRIPTION
0001	The DROP TABLE statement attempted to drop a table that resides in a partitioned table space
0002	The DROP INDEX statement attempted to drop an index required to enf

future sc.
informát



lone





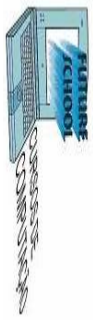
DESCRIPTION
Data definition control support is running under the controlling by application name mode. The statement is rejected because the current application is not registered in application registration table with DEFAULTAPPL on.
Data definition control support is running under the controlling by application name with exceptions mode. The statement is rejected because the object is not registered in object registration table and the current application is not registered in application registration table with DEFAULTAPPL on.
Data definition control support is running under the controlling by application name with exceptions mode. The statement is rejected because the object is registered in object registration table but the current application does not match.
Data definition control support is running under the controlling by object name with exceptions mode. The statement is rejected because the object is registered in object registration table but the current application does not match.
Data definition control support is running under the controlling by object name mode. The statement is rejected because the object is registered in object registration table but the current application does not match.
Data definition control support is running under the controlling by object name mode. The statement is rejected because the object is not registered in object registration table.

Códigos complementares para SQLCODE = -696 (reason-codes)

DESCRIPTION
NEW correlation name and NEW_TABLE name are not allowed in a DELETE trigger
OLD correlation name and OLD_TABLE name are not allowed in an INSERT trigger
OLD_TABLE name and NEW_TABLE name are not allowed in a BEFORE trigger

Códigos complementares para SQLCODE = -804 (reason-codes)

DESCRIPTION

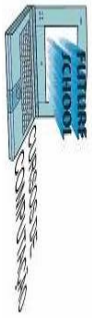


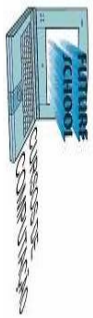
CODE
1
2
3
4
5
6

CODE
1
2
3

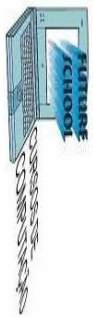
CODE

future sc.
informát

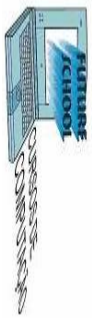




DESCRIPTION
– The DBRM name was not found in the member list of the plan a nd there is no package list for the plan. Refer to the first SQL statement under problem determination for assistance in determining the problem. – The package name was not found because there is no package list for the plan. Refer to the second SQL statement under Problem Determination for assistance in determining the problem.
The DBRM name 'dbrm-name' did not match an entry in the member list or the package list. Any of the following conditions could be the problem: Bind conditions: – The 'collection-id' in the package list was not correct when the application plan 'plan-name' was bound. Refer to the second SQL statement under Problem Determination for assistance in determining the problem. – The 'location-name' in the package list was not correct when the application 'plan-name' was bound. Refer to the second SQL s tatement under Problem Determination for assistance in determining the problem. – The 'location-name' in the CURRENTSERVER option for the bind subcommand was not correct when the application plan 'plan-name' was bound. Refer

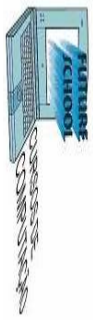


CODE
01
02

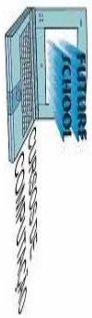


B type host variable, but

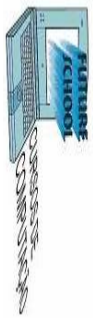
to the third SQL statement under Problem Determination for assist ance in determining the problem. Application conditions: – The CURRENT PACKAGESET special register was not set correctly by the application. – The application was not connected to the proper location.
– The DBRM name 'dbrm-name' matched one or more entries in the package list and the search of those entries did not find the pa ckage. The conditions listed under reason 02 or the following conditions might be the problem. – The DBRM of the version of the application program being execut ed was not bound (A package with the same consistency token as that of the application program was not found.) Refer to the fourth and fifth SQL statements under the Problem Determination section. – The incorrect version of the application program is being executed.
The package, 'collection-id.dbrm-name.consistencytoken', does no t exist at the remote site, 'lo cation-name'. Refer to the fifth SQL statement under the Problem Determination section.



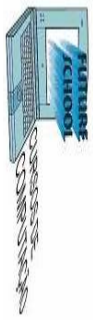
03
04


FUTURE
SCHOOL

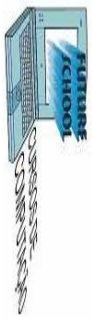
Ação
Verificar na lista de símbolos, os possíveis símbol[redacted]
Verificar se o número de colunas é igual ao número de variáveis definidas no inser ou se no comando de insert estão sendo informadas as colunas de cláusulas values. Caso não esteja, todas as colunas da tabela devem ser informadas na cláusula values, note que novas colunas podem ter sido inseridas na tabela sem alteração no programa.
Para tabela particionada, reconstruir a tabela a partir do utilitário unload/load do DB2. Utilize o unload com “PARM SQL”.
Verificar se o nome do objeto (tabela view, coluna etc) foi digitado corretamente jou se owner do objeto está de acordo com o ambiente.
Verificar se o nome da coluna está correto.
Corrija o comando SQL, adicionando a coluna da cláusula order by na tabela resultante ou eliminando a coluna da cláusula order by
Corrigir o valor da coluna e executar a aplicação novamente
Ajustar o tamanho da variável host
Verifique os tipos de dados de todos os operandos do comando SQL e, tenha certeza que esses tipos de dados são comparáveis e compatíveis como estão sendo usados.
Verificar a definição da coluna na tabela; se a mesma deve possuir valores nulos, solicitar a alteração da tabela.
Verificar a definição da tabela e tenha certeza que a variável host ou o valor da literal definida para a coluna é apropriada para o tipo de dado e observe também se a ordem das colunas estão definidas corretamente, por exemplo, no insert os valores definidos podem estar em uma ordem diferente dos valores esperados para serem inseridos se não forem especificadas as colunas após o nome da tabela, isto é, insert into tabx (a, b, c) values (1, 2, 3).
Abrir o cursor antes de fetch ou close.
Fechar o cursor aberto
Abrir o cursor
Posicionar cursor antes de update/delete
Verificar se existem relacionamentos dependentes desta cahve



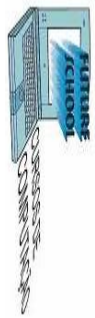
Colunas ou variáveis maiores que 254 caracteres não podem ser referenciadas em:
Funções, com exceção de substr, lenght ou value;
Cláusulas group by e order by; Instrução select distinct;
Predicados, com exceção de exist oi like;
Sub select de um union sem a keyword all;
Chaves primárias, únicas ou estrangeiras; e
Índices.



SQLCODE
104
117
134
151
204
206
208
302
304
401
407
408
501
502
507
508
531

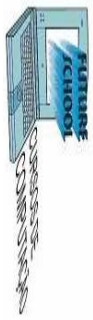

FUTURE
SCHOOL

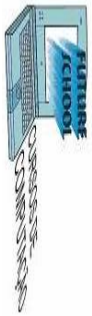
Para deletar uma linha da tabela mãe no relacionamento, é necessário deletar antes, todas as linhas da tabela pendente
Alterar os dados para atender a condição ou solicitar a alteração da regra
Solicitar autorização
Solicitar autorização
Verificar se concatenação está ok; Verificar se a compilação (step BIND) foi realizada com sucesso; Se ambiente CICS, verificar se foi emitido o comando NEWCOPY; e Se todas as opções acima estão corretas, compilar o programa novamente ou somente realizar o BIND.
Executar o comando: DISPLAY BD(DBXX) SPACENAME(TBXXXX); Executar RPF DB2 (opção (utilitários)) DISPLDB; Verificar campo STATUS; Se igual a 'COPY', falta rodar utilitário de copy; Se igual a 'CHECK', falta rodar utilitário de check; Se igual a 'RECV', falta rodar utilitário de recovery; Se igual a 'STOP', verificar qual o motivo do stop; Se igual a 'UT': -- Existe um utilitário rodando no tablespace; -- Verificar emitindo o comando – DIS UTIL(*) da seguinte maneira: RPF DB2 (opção (utilitários)) DISUTIL; -- Re executar o job correspondente com RESTART(PHASE); e Se nenhuma das ocorrências acima, verificar na mensagem o campo 'REASON CODE' e pesquisar o motivo na RPF DB2.
Diminuir o uso da CPU, colocando condições mais restritas no query ou faça um programa para obter o resultado desejado
Rodar processo novamente ou aguardar término do job que está prendendo o recurso



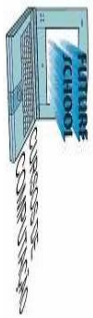
Verificar se está acessando o DB2 corretamente; Verificar se o nome do plano está correto; Verificar se o nome do programa está correto; Verificar se a compilação / BIND foi realizado com sucesso; Se ambiente CICS, verificar se foi emitido o comando NEWCOPY; e Se todas as opções acima estão corretas, compilar o programa novamente ou somente realizar o BIND.

532
545
551
552
805
818
904
905
911




FUTURE
SCHOOL

Utilizar protocolos específicos de CICS para efetivar o trabalho
Utilizar protocolos específicos de CICS para desfazer o trabalho
Corrigir o JCL e re-executar o programa



925
926
927