1 Apresentação

O acesso a uma base OpenBase por comandos SQL é realizado pelo sistema TSQL. Para a sua utilização não é necessário efetuar previamente qualquer alteração ou adaptação nos dados da base.
O TSQL pode operar tanto em modo local como no modo cliente/servidor em que uma aplicação acessa através de comandos SQL uma base que não reside na mesma máquina.
Por ser uma linguagem especializada, a SQL não possui determinadas características das linguagens de programação genérica. Por isso ela deve ser executada no modo interativo ou no modo programável em que comandos SQL são inseridos em programas desenvolvidos em outras linguagens.
No modo interativo, o TSQL oferece os utilitários tsqlwi no WINDOWS ou tsqlI no UNIX.
No modo programável, o TSQL oferece :

- No WINDOWS, um driver ODBC que efetua a conexão entre uma base OpenBase e programas desenvolvidos em ambientes como Visual Basic, Delphi, Java, Crystal Report, etc.
- No UNIX e no WINDOWS, uma interface nativa TSQL do tipo CLI (Call Level Interface) para desenvolvimento de aplicações em "C". Estas aplicações também podem ser desenvolvidas com a interface ODBC.

Este manual é dirigido a programadores que possuam alguma familiaridade com a linguagem SQL e está estruturado nos seguintes capítulos e apêndices:

Capítulo 1 - Conceitos, que apresenta e descreve alguns conceitos básicos em que a SQL está baseada: esquema (base de dados), transação, concorrência, etc. São apresentados também os diversos níveis do mecanismo de acesso concorrente.

Capítulo 2 - Elementos da linguagem, que contém a descrição dos elementos comuns aos comandos SQL: tipos de dados, expressão, condição de seleção, etc.

Capítulo 3 - Predicados, onde é feita a descrição dos predicados de uma condição de seleção de um comando SQL.

Capítulo 4 - Cláusulas, descreve as cláusulas de um comando SQL.

Capítulo 5 - Comandos, apresenta os comandos da SQL distribuídos em dois grupos. Os comandos de manipulação de dados e de manipulação de esquema.

Capítulo 6 – Interface nativa de programação.

Capítulo 7 – Catálogo SQL, onde são descritas as tabelas do catálogo de uma base de dados que contém informações relevantes para o usuário. Estas tabelas descrevem a estrutura de uma base.

Capítulo 8 - Utilitários, onde são apresentados os diversos utilitários disponíveis no TSQL.

Apêndice 1 - Instalação.

Apêndice 2 - Ativação do servidor TSQL.

Apêndice 3 - Referência a uma base OpenBase, onde são descritos os parâmetros a serem especificados por uma aplicação com comandos SQL para a abertura de uma base.

Apêndice 4 - ODBC, onde é descrito como efetuar uma conexão ODBC com o TSQL.

Apêndice 5 - Configuração, onde são descritos os possíveis parâmetros de configuração como o tamanho da área de ordenação, número do serviço TSQL, etc.

Apêndice 6 - Descrição dos ambientes interativos tsqlI e tsqlwi para execução de comandos SQL .

2 Conceitos básicos

2.1 Notação TSQL

Este capítulo tem como objetivo apresentar a notação usada no manual de referência.
Em todos os tópicos do manual há duas áreas, uma para a descrição da sintaxe e outra para as regras gerais de uso do elemento da linguagem. Em todos os tópicos, há, ainda, uma sentença "Veja também …" que remete o leitor a outros pontos do manual, fortemente relacionados com alguma explicação ali apresentada.
Descreve-se aqui a notação usada para a definição da sintaxe da linguagem. Todos os elementos possuem um nome seguido de dois pontos (:) e de sua definição.
nome_do_elemento : definição
Na definição podem existir nomes definidos a seguir, nomes de outras entradas do manual, palavras reservadas da linguagem em letras maiúsculas, símbolos da linguagem e símbolos da notação.
Os símbolos da notação têm os seguintes significados:
1. Barra vertical '|' significa OU. Deve ser usado o elemento da descrição anterior a barra ou o elemento que segue a barra.
2. Colchetes '[]' significam cláusula opcional. Elementos entre colchetes podem ou não ser colocados na sentença, conforme o caso.
3. Chaves '{}' significam grupamento de elementos. Os elementos entre chaves devem ser lidos, para efeito da notação, como um só elemento.
4. Três pontos '…' significam repetição. Um elemento ou um conjunto deles seguido de três pontos devem ser repetidos uma ou mais vezes.
A partir desta, observa-se que uma lista de um ou mais elementos separada por espaço é representada na forma:
nome_do_elemento…
Uma lista de um ou mais elementos separados por vírgula é representada na forma:
nome_do_elemento [{,nome_de_elemento}…]
Nomes de elementos do tipo 'identificador_de_…' são nomes que identificam aquele tipo de elemento e seguem a definição de identificador.
Quando um dos símbolos da notação é também um símbolo da linguagem, o símbolo da linguagem aparece entre aspas (""). Por exemplo: "[" campo "]".
Neste caso o campo não é opcional, deve-se especificar um colchete seguido do valor do campo e seguido de outro colchete.

2.2 Base de dados

Uma base de dados SQL é uma coleção de informações organizadas em subconjuntos chamados de esquema. Um esquema, por sua vez, é formado por um identificador de autorização (nome do esquema) , um conjunto de tabelas e um conjunto de índices sobre cada tabela. Toda base SQL contém um catálogo, que é um esquema composto de tabelas que armazenam as descrições de todos os objetos (tabelas e índices) da base e os relacionamentos entre eles. O identificador de autorização do catálogo é o identificador **SYSSQL**. Através de consultas SQL ao catálogo, o usuário descobre as informações sobre a estrutura de uma base de dados.
Em uma base de dados OpenBASE, todas as tabelas e índices constituem um único esquema cujo identificador **tsql** é o identificador de autorização. Como a base não possui fisicamente tabelas de um catálogo SQL, o TSQL simula a execução de consultas a estas tabelas.
Por exemplo, a consulta **select * from SYSSQL.SYSTABLES** recupera as informações sobre todas as tabelas de uma base OpenBase. A consulta **select * from SYSSQL.SYSCOLUMNS where TNAME = "t1"** recupera as informações sobre as colunas da tabela **t1**.
Veja também Catálogo.

2.3 Controle de concorrência

Concorrência é a capacidade que o sistema possui, em um ambiente multi-usuário, de permitir que várias transações acessem simultaneamente informações de uma base de dados. O grau de concorrência é o volume de transações executadas pelo sistema durante um determinado intervalo de tempo. Por outro lado, a

consistência integral dos dados em um ambiente multi-usuário implica em que alterações efetuadas por uma transação não devem afetar as outras transações em curso.
Para assegurar as características da concorrência e da consistência dos dados, o TSQL usa os métodos de bloqueio automático de linhas e de bloqueio programado de tabelas (comando LOCK TABLE).
Para se obter o melhor grau de concorrência, assegurando a consistência dos dados, é necessário que o esquema de bloqueio permita um balanceamento entre estas características em função da aplicação. Este balanceamento é obtido através do nível de isolamento da aplicação e o uso do comando LOCK TABLE.
No TSQL, os bloqueios podem ser do tipo compartilhado ou exclusivo. Em geral, o sistema coloca um bloqueio compartilhado sobre a linha a ser lida. Este tipo de bloqueio impede que uma outra transação altere a linha durante o tempo de duração do bloqueio. Outras transações podem ler a linha normalmente. O instante em que um bloqueio compartilhado é relaxado depende do nível de isolamento.
Bloqueios exclusivos são colocados sobre as linhas que são alteradas. Quando uma linha possui um bloqueio exclusivo, nenhuma outra transação pode referenciá-la. Bloqueios exclusivos são mantidos até o fim da transação que os gerou.
Os bloqueios acima descritos são automaticamente adquiridos pelo sistema. O nível de isolamento é uma característica de cada aplicação. Se o nível não é especificado, o sistema assume o nível de estabilidade de cursor (nível 1).
Em qualquer nível de isolamento é também possível requerer explicitamente bloqueios de tabelas utilizando o comando LOCK TABLE. Bloqueios de tabelas são automaticamente relaxados no fim de uma transação. Estes bloqueios podem ser automaticamente adquiridos pelo sistema quando é executado um comando que afeta a definição de uma tabela (CREATE INDEX, por exemplo).
O bloqueio de uma tabela deve ser utilizado, no caso de uma transação efetuar operações de leitura ou atualização em um grande número de linhas. Desta forma o sistema não é onerado com a obtenção do bloqueio de cada linha.
Veja também o comando LOCK TABLE.
O nível de isolamento é que determina quando os bloqueios de linha são adquiridos e relaxados pelo sistema. O TSQL oferece os seguintes níveis:

2.3.1 Nível 0. Sem bloqueio

O sistema não mantém bloqueios sobre as linhas referenciadas. A transação só opera com leitura de linhas e não pode efetuar alterações na base de dados. Este nível de bloqueio pode ser utilizado para gerar relatórios sem onerar o sistema na tentativa de obtenção dos bloqueios.
Observa-se que as linhas lidas neste nível podem refletir alterações que não foram ainda confirmadas por um comando COMMIT e que poderão ser ser eventualmente canceladas (comando ROLLBACK).

2.3.2 Nível 1. Estabilidade de cursor

Bloqueios compartilhados são relaxados na leitura de uma próxima linha da tabela e bloqueios exclusivos são mantidos até o fim da transação. A estabilidade de cursor é utilizada com frequência nas aplicações interativas nas quais os usuários podem percorrer linhas, uma de cada vez. Neste ambiente, só é necessário manter um bloqueio sobre uma linha se esta é modificada. Após a passagem do usuário para a próxima linha, bloqueios sobre linhas não modificadas são relaxados.

2.3.3 Nível 2. Leitura repetida

Bloqueios compartilhados e exclusivos são mantidos até o fim da transação. Desta forma, linhas lidas não são alteradas por outra transação até o fim da transação na qual elas foram lidas. Este nível provê um alto grau de consistência e uma relativo baixo grau de concorrência.
Este nível de isolamento não garante que toda operação de leitura sobre uma determinada tabela é reproduzível dentro de uma mesma transação, porque é possível a inserção de linhas por parte de uma transação concorrente. Através do comando LOCK TABLE é possível impedir a inserção de linhas na tabela por uma transação.

2.3.4 Nível 3. Uso exclusivo sem transação

Toda a base de dados é bloqueada para o usuário. Essencialmente, o sistema passa a operar em modo mono-usuário. Este nível deve ser usado para a execução de operações tais como salvar uma base de dados, recuperar uma base de dados, etc. O gerenciamento de transações não é efetuado, e consequentemente as atualizações na base não podem ser desfeitas através do comando ROLLBACK.

2.3.5 Nível 4. Uso exclusivo com transação

Este nível opera da mesma forma que o nível anterior com a diferença que o gerenciamento de transações é efetuado.

2.4 Transação

Uma transação consiste em uma seqüência de operações na base de dados que possui a característica de ser atômica em relação ao controle da concorrência e da recuperação. Desta forma, existem apenas duas alternativas: todas as operações presentes numa transação são realizadas como se fossem uma ou nenhuma delas é realizada.

Se um programa efetua uma operação SQL e não existe uma transação em curso para o programa, então é automaticamente iniciada uma transação para ele. Cada operação SQL subsequente efetuada pelo mesmo programa faz parte da mesma transação até que ela seja terminada. Consequentemente, transações não podem ser aninhadas.

Uma transação é encerrada com a execução de um comando COMMIT ou um comando ROLLBACK. No caso do comando ROLLBACK, todas as alterações efetuadas na base pela transação são canceladas. O comando COMMIT confirma as alterações.

2.5 Cursor

No caso da execução de um comando do tipo SELECT que produz um conjunto resultado de linhas, é necessária a existência de um mecanismo para que uma aplicação possa recuperar o conjunto linha a linha. Isto é realizado através do conceito de cursor, que assim foi nomeado por apontar para a linha corrente no conjunto resultado da mesma forma que o cursor na tela de um monitor indica a posição corrente.

A execução de um comando SELECT por uma aplicação provoca então a associação de um determinado cursor ao conjunto resultado. Desta forma, quando uma aplicação executa um comando de leitura do tipo FETCH em relação ao conjunto resultado, o cursor associado é movido para a próxima linha e esta última é retornada para a aplicação.

A forma para associar um identificador de cursor a um comando SELECT varia segundo a linguagem de desenvolvimento e pode ser explícita ou implícita.

Frequentemente as aplicações também utilizam cursores para alterar dados. Por exemplo, um usuário pode querer visualizar todos os pedidos de um determinado cliente. O programa associa então um cursor a uma consulta sobre a tabela **pedidos** e exibe cada pedido na tela esperando por um sinal do usuário para avançar para a próxima linha. O procedimento segue desta forma até que o fim do conjunto resultado seja atingido.

Neste exemplo em que a consulta recupera dados de uma única tabela e não efetua somatórios, o cursor implicitamente aponta para uma linha da tabela **pedidos**, considerando que cada linha do conjunto resultado provêm de uma única linha da tabela. Desta forma enquanto os dados são percorridos o usuário pode identificar dados a serem alterados na tabela. Por exemplo, a quantidade de um determinado pedido pode estar incorreta ou o usuário pode querer eliminar o pedido. Neste caso, a linha não é identificada por um predicado SQL e o programa utiliza o cursor para indicar qual linha deve ser alterada. A SQL oferece os comandos UPDATE POSICIONADO e DELETE POSICIONADO para este fim.

No exemplo seguinte onde os pedidos de um determinado cliente formam o conjunto resultado, seja **C1** o identificador de cursor associado pela aplicação ao comando:

SELECT pedido, data, produto, quantidade FROM pedidos WHERE cod_cliente = 10

Para aumentar em 20% o campo quantidade da linha corrente, o seguinte comando seria executado pela aplicação:

UPDATE pedidos SET quantidade = quantidade * 1.2 WHERE CURRENT OF C1

Para eliminar o pedido representado pela linha corrente, o seguinte comando seria executado pela aplicação:

DELETE FROM pedidos WHERE CURRENT OF C1

No TSQL um cursor pode ser movimentado incrementalmente para frente ou para trás, ou ser movido para uma posição absoluta ou relativa.

Veja também Especificação de consulta atualizável e Comandos.

2.6 Parâmetro SQLCODE

Neste manual, o parâmetro SQLCODE se refere ao código de erro retornado pelo TSQL após a execução de algum comando SQL. No caso da interface nativa, por exemplo, ele corresponde ao parâmetro sqlcode das diversas chamadas. A lista dos possíveis códigos de erro é apresentada no Apêndice 7.

3 Elementos da linguagem

3.1 Palavras reservadas

3.1.1 Sintaxe

ABS	**ABSOLUTE**	**ADD**
ALL	**ALTER**	**AND**
ANY	**AS**	**ASC**
AUTHORIZATION	**AVG**	
BEGIN	**BETWEEN**	**BINARY**
BIT	**BLOB**	**BOTH**
BY		
CASCADE	**CAST**	**CHAR**
CHARACTER	**CHECK**	**CLOB**
CLOSE	**COMMIT**	**CONTINUE**
COUNT	**CREATE**	**CURRENT**
CURSOR		
DATETIME	**DAY**	**DEC**
DECIMAL	**DECLARE**	**DELETE**
DESC	**DESCRIBE**	**DESCRIPTOR**
DISPLACEMENT	**DISTINCT**	**DOUBLE**
DROP		
END	**ESCAPE**	**EXCLUSIVE**
EXEC	**EXECUTE**	**EXISTS**
EXTEND		
FETCH	**FIRST**	**FLOAT**
FOR	**FOREIGN**	**FOUND**
FRACTION	**FREE**	**FROM**
GO	**GOTO**	**GRANT**
GROUP		
HAVING	**HOUR**	
IMMEDIATE	**IN**	**INDEX**
INDICATOR	**INSENSITIVE**	**INSERT**
INT	**INTEGER**	**INTERVAL**
INTO	**IS**	
KEY		
LANGUAGE	**LARGE**	**LAST**
LEADING	**LENGTH**	**LIKE**
LOCAL	**LOCATOR**	**LOCK**
LOWER		
MAX	**MIN**	**MINUTE**
MOD	**MODE**	**MODULE**
MONTH		
NEXT	**NONE**	**NOT**
NULL	**NUMERIC**	
OBJECT	**OF**	**ON**
ONLY	**OPEN**	**OPTION**
OR	**ORDER**	**OVERLAY**
PLACING	**POSITION**	**PRECISION**
PREPARE	**PRIMARY**	**PRIOR**
PRIVILEGES	**PROCEDURE**	**PUBLIC**
READ	**REAL**	**REFERENCES**
RELATIVE	**REVOKE**	**ROLLBACK**
SCHEMA	**SCROLL**	**SECOND**
SECTION	**SELECT**	**SET**
SHARE	**SMALLINT**	**SOME**

SQL	**SQLCODE**	**SQLERROR**
SUBSTRING	**SUM**	**SYSTEM**
TABLE	**TIME**	**TO**
TRAILING	**TRIM**	
UNION	**UNIQUE**	**UPDATE**
UPPER	**USER**	**USING**
VALUES	**VARCHAR**	**VARYING**
VIEW		
WHENEVER	**WHERE**	**WITH**
WORK		
YEAR		

3.1.2 Regras

As palavras reservadas podem ser escritas em letras maiúsculas ou minúsculas.

3.2 Identificador

3.2.1 Sintaxe

identificador : { letra | _ { letra | dígito | # } }
[[_] { letra | dígito | # }]…

3.2.2 Regras

Um identificador deve ter no máximo 18 caracteres.

3.3 Constante

3.3.1 Sintaxe

constante :
cadeia_de_caracteres |
constante_numérica |
constante_data_hora
cadeia_de_caracteres :
cadeia_padrão |
cadeia_C
constante_numérica :
constante_numérica_exata |
constante_numérica_aproximada
constante_data_hora :
DATETIME < {data [hora] | hora [data]} > [qualificador_data_hora]
hora :
valor_hora [“:” valor_minuto [“:” valor_segundo]] |
valor_minuto [“:” valor_segundo] |
valor_segundo
data :
[[[BC | AD] valor_ano -] valor_mes -] valor_dia |
valor_dia [/ valor_mes [/ valor_ano [AC | DC]]]
qualificador_data_hora : campo_data_hora TO campo_data_hora
campo_data_hora : YEAR | MONTH | DAY | HOUR | MINUTE | SECOND

3.3.2 Regras

Uma cadeia de caracteres padrão é delimitada por caracteres apóstrofe (‘) e pode conter qualquer caracter válido. O apóstrofe (‘) dentro de uma cadeia deve ser representado por um par de caracteres apóstrofe (‘‘).
Uma cadeia de caracteres C é delimitada por aspas (“) e segue as mesmas regras de formação de cadeia da linguagem C, podendo conter caracteres de controle como “\n”, “\r”, etc.
Uma constante numérica exata é uma seqüência de dígitos precedida ou não por sinal (+ ou -).
Ela pode conter uma parte inteira e uma parte decimal separados por um ponto (.). Se a constante não tiver parte decimal, o ponto no fim é opcional. Porém, se o número só tiver parte decimal o ponto no inicio é obrigatório (não é necessário colocar um 0 inteiro antes do ponto).
Uma constante numérica aproximada é uma mantissa seguida pelo caracter ‘E’ (ou ‘e’) seguido de um expoente. A mantissa é uma constante numérica exata e o expoente deve ser um inteiro precedido

opcionalmente por um sinal. O valor da constante numérica aproximada é o valor da mantissa multiplicado pela potência de 10 correspondente ao expoente especificado.
Os valores de ano, mês, dia, hora, minuto e segundo são inteiros sem sinal e devem respeitar a correção do valor de uma data. Os identificadores **BC** (ou **AC**) e **AD** (ou **DC**) indicam, respectivamente, datas antes e depois da era cristã.
Em geral, uma constante_data_hora é auto-explicativa no que diz respeito ao significado de seus valores (2/4/1992 e 1992-4-2 significam 2 de abril de 1992 e 17:22:30 indica 17 horas, 22 minutos e 30 segundos). Existem apenas três casos que podem gerar ambigüidade para os quais existem uma interpretação padrão. Para alterá-los é necessária a utilização do qualificador_data_hora.

DATETIME <valor>	indica um dia.
DATETIME <valor / valor>	indica dia e mês.
DATETIME <valor - valor>	indica mês e dia.
DATETIME <valor : valor>	indica hora e minuto.

Em todos os outros casos, o possível qualificador_data_hora deve corresponder à definição da constante. Dentro de uma constante numérica, não pode haver nenhum caracter branco, seja entre o sinal e o número, seja entre dígitos (Veja Tipos de Dados).

3.4 Comentário

3.4.1 Sintaxe

comentário : — caracter… fim_de_linha

3.4.2 Regras

Pode-se inserir comentários (delimitados por — e o fim da linha), após identificadores, palavras reservadas e constantes.

3.5 Tipos de dados

3.5.1 Sintaxe

tipo_de_dado :
tipo_cadeia_de_caracteres |
tipo_numérico_exato |
tipo_numérico_aproximado |
tipo_data_hora
tipo_cadeia_de_caracteres :
{ CHARACTER | CHAR } [VARYING] [(tamanho)] |
VARCHAR [(tamanho)] |
{ CHARACTER | CHAR } LARGE OBJECT [(tamanho_longo)] |
CLOB [(tamanho_longo)]
tipo_cadeia_binária_longa :
BINARY LARGE OBJECT [(tamanho_longo)] |
BLOB [(tamanho_longo)]
tamanho_longo :
tamanho |
tamanho K |
tamanho M |
tamanho G
tipo_numérico_exato :
INTEGER |
INT |
SMALLINT |
{ NUMERIC | DECIMAL | DEC } [(precisão [,escala])]
tipo_numérico_aproximado :
DOUBLE [PRECISION] |
REAL |
FLOAT [(precisão)]
tipo_data_hora : DATETIME [qualificador_data_hora]
qualificador_data_hora : campo_data_hora TO campo_data_hora
campo_data_hora : YEAR | MONTH | DAY | HOUR | MINUTE | SECOND

3.5.2 Regras

Tamanho, precisão e escala são números inteiros.
CHAR é equivalente a CHARACTER. VARCHAR é equivalente a CHARACTER VARYING. CLOB é equivalente a CHARACTER LARGE OBJECT. BLOB é equivalente a BINARY LARGE OBJECT. INT é equivalente a INTEGER. DEC e NUMERIC são equivalentes a DECIMAL. DOUBLE é equivalente a DOUBLE PRECISION.
O tipo CHARACTER especifica o tipo de dados cadeia de caracteres de tamanho fixo. O campo tamanho especifica o tamanho da cadeia e deve estar entre 1 e 32760. Se for omitido, é igual a 1.
O tipo VARCHAR especifica o tipo de dados cadeia de caracteres de tamanho variável. O campo tamanho especifica o tamanho máximo da cadeia e deve estar entre 1 e 32760. Se for omitido, é igual a 1.
O tipo CLOB especifica o tipo de dados cadeia de caracteres de tamanho variável. O campo tamanho_longo especifica o tamanho da cadeia e deve estar entre 1 e 1023 (*bytes*, *Kbytes*, *Mbytes* ou *Gbytes*). Se for omitido, é igual a 1 *byte*.
O tipo BLOB especifica o tipo de dados cadeia binária de tamanho variável. O campo tamanho_longo especifica o tamanho da cadeia e deve estar entre 1 e 1023 (*bytes*, *Kbytes*, *Mbytes* ou *Gbytes*). Se for omitido, é igual a 1 *byte*.
O tipo INTEGER especifica os números inteiros entre -2147483648 e 2147483647.
O tipo SMALLINT especifica os números inteiros entre -32768 e 32767.
O tipo DECIMAL especifica valores decimais de representação exata e ponto fixo. O número total de dígitos é especificado pela precisão que pode variar entre 0 e 18 (se for omitida, é igual a 18). O número de dígitos fracionários é especificado pela escala e varia entre 0 e o valor da precisão. Se a escala for omitida, o número de dígitos fracionários é igual a zero.
Os tipos DOUBLE, REAL e FLOAT especificam valores decimais numéricos não exatos com ponto flutuante e número de dígitos binários igual a 56 para a DOUBLE PRECIOSION, 24 para o tipo real ou especificada pela precisão para o tipo FLOAT. O tipo FLOAT é equivalente ao tipo REAL se a precisão for omitida ou for menor ou igual a 24 e equivalente ao tipo DOUBLE se a precisão for maior que 24.
O tipo DATETIME especifica datas com uma precisão indicada pelo qualificador. O campo inicial do qualificador deve ser sempre maior que o final e indica uma precisão que inclui todos os campos entre eles. Se o qualificador_data_hora for omitido, é igual a YEAR TO SECOND.
Um campo_data_hora é restrito a uma determinada faixa de valores. O campo ano (YEAR) pode variar entre 9999 AC e 9999 DC. O campo mês (MONTH) varia entre 1 e 12. O valor de dia (DAY) está entre 1 e 31 e está sujeito aos valores do mês e ano para formar uma data correta. O campo hora (HOUR) varia entre 0 e 23 e os valores de minuto (MINUTE) e segundo (SECOND) variam entre 0 e 59.
A correspondencia entre os tipos do OpenBase e os tipos da SQL é apresentada na forma de uma tabela que se encontra na descrição do utilitário tsqlbd no capítulo Utilitários.

3.6 Tabela

3.6.1 Sintaxe

tabela : [identificador_de_autorização .] identificador_de_tabela

3.6.2 Regras

A referência a uma tabela deve especificar uma tabela definida na base de dados.

3.7 Coluna

3.7.1 Sintaxe

coluna : [{ tabela | identificador_de_correlação } .] identificador_de_coluna

3.7.2 Regras

A referência a uma coluna deve especificar uma coluna de uma tabela definida na base de dados.
Se a coluna estiver qualificada por uma tabela ou identificador de correlação, o nome da coluna deve estar definido na tabela (uma ou mais) associada ao qualificador e será considerada a tabela de escopo mais interno. O escopo de uma tabela ou identificador de correlação é definido pela cláusula FROM contida na subconsulta ou especificação de consulta mais próxima.
Caso a coluna não esteja qualificada, está implicitamente especificada (se for a única possível) uma tabela ou correlação dentro do escopo mais interno que contenha a coluna em sua definição.

Veja também Tabela e Cláusula FROM.

3.8 Função de conjunto

3.8.1 Sintaxe

Função_de_conjunto :
COUNT(*) |
funçao_DISTINCT |
função_ALL
função_DISTINCT : {AVG | MAX | MIN | SUM | COUNT} (DISTINCT expressão)
função_ALL : { AVG | MAX | MIN | SUM | COUNT} ([ALL] expressão)

3.8.2 Regras

O argumento de uma função DISTINCT é um conjunto de valores derivado da aplicação da expressão a cada linha da tabela ou grupo, eliminados todos os valores nulos ou duplicados. Já o argumento de uma função ALL é o conjunto de valores derivado da aplicação da expressão, eliminados apenas os valores nulos.
O resultado da função COUNT(*) é a cardinalidade (número de linhas) da tabela ou grupo. O tipo é INTEGER.
O resultado da função DISTINCT COUNT é a cardinalidade de seu argumento. O tipo é INTEGER.
O resultado da função AVG é a média aritmética dos valores de seu argumento e o da função SUM é o somatório dos valores de seu argumento. Ambas só podem ser aplicadas a valores de tipo numérico.
O tipo do resultado da função AVG ou SUM é:

- DOUBLE se o tipo do argumento é DOUBLE ou REAL, ou
- INTEGER se o tipo do argumento é INTEGER ou SMALLINT, ou
- DECIMAL se o tipo do argumento é DECIMAL.

Se o tipo do resultado da função AVG ou SUM é DECIMAL:

- a precisão do resultado da função AVG ou SUM é 18,
- a escala do resultado da função SUM é igual à escala do argumento, e
- a escala do resultado da função AVG é igual à escala do argumento somada a 18 e subtraída da precisão do argumento.

As funções MAX e MIN tem como resultado respectivamente o maior e o menor valor de seu argumento e é do mesmo tipo da coluna ou da avaliação da expressão. As funções DISTINCT MAX e DISTINCT MIN são equivalentes a MAX e MIN, respectivamente.
Em uma especificação de consulta não pode ocorrer mais de uma função DISTINCT.
Se é vazio o conjunto dos valores do argumento das funções AVG, MAX, MIN ou SUM, então o resultado é nulo.
Veja também Expressão e Tipos de dados.

3.9 Função escalar

3.9.1 Sintaxe

função_escalar :
função EXTEND |
função_CAST |
função_SUBSTRING |
função_LENGTH |
função_POSITION |
função_OVERLAY |
função_UPPER |
função_LOWER |
função_TRIM |
função_ABS |
função_MOD
função_EXTEND : EXTEND (expressão [,qualificador_data_hora])
qualificador_data_hora : campo_data_hora TO campo_data_hora
campo_data_hora : YEAR | MONTH | DAY | HOUR | MINUTE | SECOND
função_CAST : CAST(operando_cast AS tipo_de_dado_alvo)
operando_cast : expressão | NULL
tipo_de_dado_alvo : tipo_de_dado

função_LENGTH : LENTGH (expressão)
função_SUBSTRING :
SUBSTRING(expressão, expressão [,expressão]) |
SUBSTRING(expressão FROM expressão [FOR expressão])
função_POSITION : POSITION(expressão IN expressão)
função_OVERLAY: OVERLAY(expressão PLACING expressão
FROM expressão [FOR expressão])
função_UPPER: UPPER(expressão)
função_LOWER: LOWER(expressão)
função_TRIM:
TRIM([[especificação_trim] [caracter_trim] FROM] expressão)
especificação_trim : LEADING | TRAILING | BOTH
função_ABS : ABS(expressão)
função_MOD : MOD(expressão, expressão)

3.9.2 Regras

A função EXTEND ajusta a precisão do argumento para a precisão indicada pelo qualificador. Esta função só pode ser aplicada a valores de tipo DATETIME e retorna um valor do mesmo tipo e com precisão especificada pelo qualificador. Se este for omitido, é retornado um valor de precisão YEAR TO SECOND.
Se a precisão do argumento da função EXTEND contém campos não incluídos na precisão especificada pelo qualificador_data_hora, então os campos excedentes são truncados. Se a precisão especificada pelo qualificador contém campos à esquerda dos campos do argumento, então campos adicionais são iniciados com valores extraídos da data e hora correntes. O resultado deve constituir uma data correta.
Se a precisão especificada pelo qualificador contém campos à direita dos campos do argumento, então estes campos adicionais são iniciados com o valor 1 para os campos mês e dia e com o valor 0 para os campos hora, minuto e segundo.
A função CAST transforma o tipo do valor de uma expressão. Se a expressão for igual a NULL ou o valor resultante da expressão for nulo então o resultado da função cast é um valor nulo cujo tipo é o tipo especificado pelo tipo_de_dado_alvo. Se o tipo da expressão for numérico exato ou numérico aproximado, o tipo do parâmetro tipo_de_dado_alvo não poderá ser DATETIME e vice-versa. Se a expressão e o tipo_de_dado_alvo forem ambos do tipo CHAR ou VARCHAR e o tamanho de tipo_de_dado_alvo não for igual ao tamanho da cadeia resultante da expressão, o resultado será truncado ou completado com brancos a direita. Na conversão de uma cadeia de caracteres para um tipo numérico ou para um tipo DATETIME, a cadeia deverá representar um número válido ou uma data válida senão será retornado um erro. A conversão de uma data para um tipo DATETIME segue as mesmas regras que a função EXTEND.
A função LENGTH retorna um SMALLINT indicando o tamanho do operando que deve ser uma cadeia. Os caracteres brancos não significativos, ou seja, à direita do último caracter não branco são excluídos no cálculo do tamanho. Se o operando for nulo, o resultado da função também será nulo.
A função SUBSTRING retorna a subcadeia do primeiro argumento que começa no caracter de ordem definida pelo segundo argumento e cujo tamanho é dado pelo terceiro argumento. O primeiro argumento deve ser do tipo cadeia de caracteres e os demais do tipo numérico exato com precisão igual a zero. Se a cadeia é do tipo VARCHAR, o resultado será do mesmo tipo, senão seu tamanho será o especificado pelo terceiro argumento. No caso de um dos argumentos ser nulo, o resultado retornado será nulo. O argumento da função que especifica a posição inicial deve ser maior que zero e menor ou igual ao tamanho da cadeia original. O terceiro argumento, se especificado, deve ser maior ou igual a zero e menor ou igual ao tamanho da cadeia original menos a posição inicial especificada acrescido de um (tamanho da cadeia original a partir da posição inicial). Quando este argumento não é especificado, é considerado um tamanho igual ao tamanho da cadeia a partir da posição inicial.
A função POSITION retorna um SMALLINT indicando a posição da primeira ocorrência da subcadeia correspondente ao primeiro operando na cadeia correspondente ao segundo operando. Zero é retornado se o primeiro operando não for subcadeia. Se o primeiro operando for de tamanho 0, 1 é retornado. Se algum dos operandos for nulo , o resutado será nulo.
A função OVERLAY modifica a cadeia especificada pelo operando 1 substituindo uma determinada subcadeia por uma cadeia chamada cadeia de substituição e definida pelo operando 2. A posição inicial da subcadeia é definida pelo operando 3 e o seu tamanho pelo operando 4. Quando o tamanho da subcadeia não for definido ou igual a zero nada é substituido e a cadeia retornada pela função é a cadeia original na qual foi inserida a cadeia de sustituição na posição determinada.
A função UPPER transforma cada letra minúscula do operando, na letra maiúscula que lhe corresponde.

A função LOWER transforma cada letra maiúscula do operando na letra minúscula que lhe corresponde.
A função TRIM dependendo da especificação_trim elimina caracteres iguais ao caracter_trim a direita ou a esquerda ou dos dois lados do operando expressão que deve ser do tipo cadeia de caracteres. O valor default da especificação_trim é TRAILING e o valor default do caracter_trim é branco.
A função ABS retorna o valor absoluto da expressão que deve ser do tipo numérica.
A função MOD retorna o resto da divisão do primeiro operando pelo segundo operando que devem ser do tipo númericos exatos.
Veja também Expressão e Tipos de dados.

3.10 Valor

3.10.1 Sintaxe

valor : constante | USER | CURRENT | parâmetro_dinâmico
parâmetro_dinâmico : ?

3.10.2 Regras

O valor especificado por USER é igual ao identificador de autorização do usuário. O tipo é uma cadeia de caracteres de tamanho 18.
O valor especificado por CURRENT é igual à data e hora correntes. O tipo é DATETIME YEAR TO SECOND.
Parâmetros dinâmicos podem ser especificados para entrada ou saída. Ou seja, os valores podem ser passados para o comando SQL a partir de variáveis do programa de aplicação (entrada) ou podem ser variáveis do programa de aplicação onde valores devem ser armazenados (saída).
Veja também Constante.

3.11 Expressão

3.11.1 Sintaxe

expressão : [+ | -] primário [{ + | - | * | / | "||" } expressão]…
primário :
valor |
coluna |
função_de_conjunto |
função_escalar |
(expressão)

3.11.2 Regras

A avaliação de uma expressão respeita a seguinte precedência:

1. expressões entre parênteses
2. operações unárias
3. operações de multiplicação e divisão
4. operações de adição e subtração
5. operações de mesma precedência da esquerda para a direita.

Se o valor de qualquer primário for nulo, então o valor da expressão é nulo.
Em uma divisão, o divisor deve ser diferente de 0.
Uma expressão contendo uma cadeia de caracteres ou um primário do tipo data-hora não deve conter operadores matemáticos.
Um operador unário não pode ser seguido de uma constante numérica iniciada por um sinal.
O tipo do resultado de uma operação matemática é:

- DOUBLE se o tipo de algum operando é DOUBLE ou REAL, ou
- DECIMAL se o tipo de algum operando é DECIMAL, ou
- INTEGER se o tipos dos operandos são INTEGER e/ou SMALLINT.

Se o tipo do resultado de uma operação matemática é DECIMAL:

- a precisão do resultado da operação de soma ou subtração é a maior das precisões dos operandos mais 1 (menor ou igual a 18),
- a escala do resultado da operação de soma ou subtração é a maior das escalas dos operandos,
- a precisão do resultado da operação de multiplicação ou divisão é a soma das precisões dos operandos,
- a escala do resultado da operação de multiplicação ou divisão é a soma das escalas dos operandos.

O resultado da operação "||" é a concatenação de duas cadeias de caracteres, excluídos os caracteres brancos não significativos. No caso de um dos operandos ser do tipo VARCHAR, o resultado é do tipo VARCHAR, senão o tamanho do resultado da concatenação é a soma do tamanho de seus operandos.
Veja também Valor, Coluna e Função.

3.12 Condição

3.12.1 Sintaxe

condição : item_de_seleção |
item_de_seleção {AND|OR} item_de_seleção
item_de_seleção : [NOT] { predicado | (condição) }
predicado :
predicado_de_comparação |
predicado_BETWEEN |
predicado_LIKE |
predicado_NULL |
predicado_IN |
predicado_quantificado |
predicado_EXISTS

3.12.2 Regras

Retorna o valor verdadeiro, falso ou não conhecido, dependendo do resultado da aplicação de operadores lógicos a outras condições. O valor não conhecido como resultado final de uma condição é tratado como falso pelas cláusulas SQL (WHERE e HAVING), pois ambas as respostas não retornam nenhum valor válido. A diferença entre esses valores se dá na avaliação da condição segundo a tabela-verdade abaixo.
A avaliação de uma condição é feita na seguinte ordem:

1. Expressões entre parênteses.
2. Operações NOT.
3. Operações AND.
4. Operações OR.
5. Operações de mesma precedência da esquerda para a direita.

Os operadores lógicos, NOT, AND e OR são definidos, segundo as seguinte tabela:

NOT		AND	V	F	?	OR	V	F	?
V	F	V	V	F	?	V	V	V	V
F	V	F	F	F	F	F	V	F	?
?	?	?	?	F	?	?	V	?	?

V denota o valor verdadeiro
F denota o valor falso
? denota o valor não conhecido
O resultado de uma condição onde não apareçam operadores lógicos é o valor do predicado especificado.
Veja também Predicado.

3.13 Especificação de consulta

3.13.1 Sintaxe

especificação_de_consulta :
cláusula_SELECT
cláusula_FROM
[cláusula_WHERE]
[cláusula_GROUP_BY]
[cláusula_HAVING]

3.13.2 Regras

Retorna uma tabela definida pelas cláusulas SELECT, FROM e opcionalmente pelas cláusulas WHERE, GROUP BY e HAVING.
O resultado de uma especificação de consulta é a tabela formada pelas colunas definidas na cláusula SELECT a partir das tabelas especificadas na cláusula FROM e segundo as restrições das demais cláusulas. Cada coluna da tabela resultante tem o mesmo tipo, tamanho, precisão e escala da expressão da qual é derivada.
Se o resultado das cláusulas for uma tabela agrupada de zero grupos, o resultado da especificação de consulta é uma tabela vazia.
Apenas uma função DISTINCT AVG, COUNT ou SUM deve aparecer em uma especificação de consulta, a não ser que esteja presente em uma subconsulta.
Uma especificação de consulta é dita atualizável se:

- a opção DISTINCT não for especificada na cláusula SELECT;
- toda expressão especificada na cláusula SELECT for uma referência a uma coluna;
- a cláusula FROM especificar exatamente uma única tabela que seja atualizável;
- a cláusula WHERE não incluir uma subconsulta;
- não forem especificadas as cláusulas GROUP BY ou HAVING.

Cada uma das cláusulas de uma especificação de consulta realiza determinada operação sobre uma tabela, retornando uma outra tabela modificada de algum modo.
A primeira cláusula a ser executada é a cláusula FROM que faz o produto cartesiano entre as tabelas especificadas. A seguir, a cláusula WHERE, se houver, seleciona linhas da tabela gerada pela cláusula anterior de acordo com uma condição. A cláusula GROUP BY agrupa uma tabela e a cláusula HAVING seleciona grupos da tabela segundo uma condição.
Finalmente, a cláusula SELECT é executada sobre a tabela resultante da aplicação das demais cláusulas e indica as colunas que pertencerão a tabela retornada pela consulta.
Veja também Cláusulas.

3.14 Expressão de consulta

3.14.1 Sintaxe

expressão_de_consulta :
{ especificação_de_consulta |
(expressão_de_consulta) }
[UNION [ALL]
{ especificação_de_consulta |
(expressão_de_consulta) }]…

3.14.2 Regras

Uma expressão de consulta especifica uma tabela a ser retornada numa consulta.
Se o operador UNION não for especificado:

- a expressão de consulta é atualizável, no caso da especificação de consulta ser atualizável;
- a descrição da expressão de consulta é igual à descrição da especificação de consulta.

Se o operador UNION for especificado:

- a expressão de consulta não é atualizável;
- grau de todas as especificações de consulta deve ser o mesmo;
- os nomes das colunas da tabela especificada pela expressão de consulta são iguais aos nomes das colunas da primeira especificação de consulta;
- os tipos das colunas correspondentes das especificações de consulta devem ser comparáveis;
- se o tipo de duas colunas correspondentes é CHARACTER, o tipo da coluna resultante é CHARACTER de tamanho igual ao maior dos dois tamanhos;
- se o tipo de uma das duas colunas correspondentes é NUMERIC e o outro é numérico exato, o tipo da coluna resultante é NUMERIC;
- se o tipo de uma das duas colunas correspondentes é numérico aproximado, o tipo da coluna resultante é DOUBLE;
- se o tipo de duas colunas correspondentes é DATETIME, o tipo da coluna resultante é DATETIME, as precisões das duas colunas devem ter o mesmo campo inicial que será o campo inicial da precisão da coluna resultante e o campo final da precisão resultante é o maior dos campos finais;

- se uma das duas colunas correspondentes pode ter valores nulos, então a coluna resultante pode ter valores nulos.

Na avaliação de uma expressão de consulta, as expressões entre parênteses são primeiro avaliadas. As operações de mesma precedência são avaliadas da esquerda para a direita.
O operador UNION elimina as linhas duplicadas na operação de união entre os operandos. Para que linhas duplicadas não sejam eliminadas, é necessário especificar a opção UNION ALL.
Veja também Especificação de consulta, Cláusulas e Comando SELECT dinâmico.

3.15 Subconsulta

3.15.1 Sintaxe

subconsulta : (especificação_de_consulta)

3.15.2 Regras

Uma subconsulta especifica uma coleção de valores obtidos do resultado de uma especificação de consulta. Ou seja, retorna a tabela formada pelas colunas especificadas na cláusula SELECT a partir da tabelas referidas na cláusula FROM e segundo as restrições das demais cláusulas.
O grau da tabela (número de colunas) derivada na subconsulta é o número de elementos da cláusula SELECT e deve ser igual a um, a não ser quando a subconsulta está especificada em um predicado EXISTS.
Podem ser feitas, nas cláusulas SELECT, WHERE ou HAVING da subconsulta, referências a colunas das tabelas referidas nas cláusulas FROM mais externas. É a chamada referência externa. Quando um nome de coluna aparece na subconsulta e esta não pertence às tabelas especificadas na cláusula FROM, ela pode referenciar colunas das tabelas tratadas pelos níveis externos da consulta.
Deste modo, podem existir situações ambíguas na definição das cláusulas. A SQL considera que uma referência a uma coluna numa subconsulta indica uma coluna de uma tabela referenciada no mesmo nível ou em algum dos níveis externos, priorizando o mais próximo.
Possíveis ambigüidades devem ser resolvidas com o uso de qualificadores ou identificadores de correlação.
Veja também Cláusula e Especificação de consulta.

4 Predicados

4.1 Predicado de comparação

4.1.1 Sintaxe

predicado_de_comparação : expressão comparador {expressão | subconsulta}
comparador : = | <> | < | <= | > | >= | **!=** | **!>** | **!<**

4.1.2 Regras

Compara dois valores que devem ser de tipos comparáveis. O valor de uma comparação é verdadeiro, falso ou não conhecido. A comparação tem resultado não conhecido se o valor de uma das expressões for nulo ou se o resultado da subconsulta é nulo ou vazio. O resultado da subconsulta deve ter no máximo um único valor.
O comparador "!=" é equivalente a "<>" e retorna verdadeiro se as expressões são diferentes. Os comparadores "!>" e "!<" eqüivalem, respectivamente aos operadores "<=" e ">=".
A comparação de duas cadeias de caracteres é feita caracter a caracter da esquerda para a direita e respeitando a ordenação da tabela ASCII. Se duas cadeias de tamanhos diferentes são comparadas, a menor delas é completada com caracteres brancos para que tenham o mesmo comprimento e a seguir comparadas.
A comparação de dois valores do tipo data-hora é efetuada após o ajuste de ambos os valores à precisão especificada pelo maior dos campos iniciais das precisões dos valores até o maior dos campos finais.
Veja também Condição, Expressão e Subconsulta.

4.2 Predicado BETWEEN

4.2.1 Sintaxe

predicado_between : expressão [NOT] BETWEEN expressão AND expressão

4.2.2 Regras

O predicado BETWEEN especifica a comparação de um valor dentro de uma faixa de valores. Para isso, os tipos resultantes das três expressões devem ser comparáveis.

A sentença “x BETWEEN y AND z” é equivalente à sentença “x >= y AND x <= z”, assim como o predicado “x NOT BETWEEN y AND z” é equivalente à condição “NOT (x BETWEEN y AND z)”.
Veja também Expressão e Condição.

4.3 Predicado LIKE

4.3.1 Sintaxe

predicado_like : expressão [NOT] LIKE padrão [ESCAPE valor_escape]
padrão : expressão
valor_escape : expressão

4.3.2 Regras

O predicado LIKE compara um valor com um padrão de caracteres. O valor da expressão, o padrão e o valor ESCAPE devem ser do tipo cadeia de caracteres.
O valor do predicado é não conhecido quando a expressão ou o padrão tem valor nulo.
Quando o valor ESCAPE não é especificado ou é nulo, o caracter sublinhado (“_”) no padrão representa a ocorrência de um caracter qualquer e o caracter porcentagem (“%”) representa a seqüência de 0 ou mais caracteres. Os demais caracteres representam a si mesmos.
Se o valor ESCAPE for especificado, a interpretação acima dos caracteres sublinhado e porcentagem é ignorada quando precedidos pelo primeiro caracter do valor ESCAPE. Neste caso, o par de caracteres representará o próprio caracter sublinhado ou porcentagem.
O predicado “x NOT LIKE y” é equivalente à “NOT (x LIKE y)”.
Veja também Expressão e Condição.

4.3.3 Exemplo

O exemplo 1 retorna verdadeiro se o nome contiver a palavra Silva em qualquer parte da cadeia:

nome LIKE ‘%Silva%’

O exemplo 2 retorna verdadeiro se a referência do produto contém exatamente 3 caracteres, sendo o primeiro deles um S:

produto LIKE ‘S__’

O exemplo 3 retorna verdadeiro se a referência do produto começa com o caracter sublinhado:

produto LIKE ‘_%’ ESCAPE ‘\’

4.4 Predicado NULL

4.4.1 Sintaxe

predicado_NULL : expressão IS [NOT] NULL

4.4.2 Regras

O predicado compara o valor da expressão com o valor nulo, retornando verdadeiro se a expressão for nula. A condição “NOT (x IS NULL)” equivale a “x IS NOT NULL”.
Veja também Expressão e Condição.

4.5 Predicado IN

4.5.1 Sintaxe

predicado_IN : expressão [NOT] IN { (expressão [,expressão]…) | subconsulta }

4.5.2 Regras

O predicado IN compara o valor da expressão com uma coleção de valores especificado na lista ou na subconsulta, assim os tipos de todos os valores envolvidos devem ser comparáveis. Uma lista de valores não deve conter funções de grupamento ou especificações de coluna.
Os predicados “x IN subconsulta” e “x =ANY subconsulta” são equivalentes, assim como “x NOT IN S” e “NOT (x IN S)”.
Veja também Expressão, Valor, Subconsulta e Condição.

4.5.3 Exemplos

O exemplo 1 lista os pedidos feitos nos três primeiros dias de março de 1992:

SELECT cód_pedido, data, cliente
FROM pedidos

```
WHERE data IN (DATETIME<01/03/1992>, DATETIME<02/03/1992>,
                                    DATETIME<03/03/1992>)
```

O exemplo 2 lista os vendedores que trabalham em escritórios que venderam acima da meta:

```
SELECT nome
FROM vendedores
WHERE escritório IN (SELECT escritório FROM escritórios WHERE vendas > meta)
```

4.6 Predicado quantificado

4.6.1 Sintaxe

predicado_quantificado : expressão comparador [ALL|ANY|SOME] subconsulta
comparador : = | <> | < | <= | > | >= | != | !> | !<

4.6.2 Regras

O predicado quantificado compara um valor com uma coleção de valores retornado pela subconsulta. Os tipos da expressão e do resultado da subconsulta devem ser comparáveis.
O predicado ALL é verdadeiro se o resultado da subconsulta for vazio ou se a comparação entre a expressão e cada valor resultante da subconsulta for verdadeiro. O predicado é falso se a comparação entre a expressão e algum valor resultante da subconsulta for falso.
O predicado ANY (ou SOME) retorna verdadeiro se o resultado da comparação entre a expressão e algum dos valores retornados na subconsulta for verdadeiro. O predicado é falso se o resultado da subconsulta for vazio ou se as comparações entre a expressão e todos os valores da subconsulta forem falsos.
O valor do predicado é não conhecido se não for verdadeiro nem falso.
Veja também Expressão, Subconsulta e Predicado de comparação.

4.6.3 Exemplo

O exemplo 1 lista os vendedores que tem pelo menos um pedido que representa mais de 20 por cento de sua cota:

```
SELECT nome
FROM vendedores
WHERE (0.2 * cota) < ANY (SELECT preco
                          FROM pedidos
                          WHERE pedidos.vendedor = vendedores.cod_vendedor)
```

O exemplo 2 lista os escritórios onde todos os vendedores venderam mais que metade da meta do escritório.

```
SELECT escritório, cidade
FROM escritorios
WHERE (0.5 * meta) < ALL (SELECT vendas
                          FROM vendedores
                          WHERE escritorios.escritorio = vendedores.escritorio)
```

4.7 Predicado EXISTS

4.7.1 Sintaxe

predicado_exists : EXISTS subconsulta

4.7.2 Regras

O predicado EXISTS é verdadeiro se o conjunto de valores retornado pela subconsulta não é vazio.
Veja também Subconsulta e Condição.

4.7.3 Exemplo

A consulta a seguir lista os nomes dos clientes que fizeram um pedido de mais de 200 unidades:

```
SELECT clientes.nome
FROM clientes
WHERE EXISTS (SELECT *
              FROM pedidos
              WHERE clientes.cod_cliente = pedidos.cliente
              AND pedidos.quantidade >= 200)
```

5 Cláusulas

5.1 Cláusula FROM

5.1.1 Sintaxe

cláusula_FROM :
FROM tabela [identificador_de_correlação]
[, tabela [identificador_de_correlação]]...

5.1.2 Regras

A cláusula FROM especifica uma tabela derivada de uma ou mais tabelas e a descrição de seu resultado é a concatenação das descrições das tabelas nela especificadas, na ordem em que seus nomes aparecem.
Um identificador de correlação é usado para identificar a tabela a ele associada. Caso a tabela não possua um, ela é dita exposta. Nomes de tabela expostos e/ou identificadores de correlações em uma cláusula FROM não podem ser iguais. Quando se deseja fazer referência a mais de uma instância uma mesma tabela ao mesmo tempo, o uso de identificadores de correlação se torna necessário para eliminar possíveis ambigüidades. O identificador de correlação pode ser usado mesmo sem que haja necessidade explícita, apenas para tornar a identificação da tabela mais legível no contexto.
O escopo dos nomes de correlação e dos nomes de tabelas expostos de uma cláusula FROM é a subconsulta mais interna ou a especificação de consulta que contém a cláusula FROM. Uma cláusula FROM define um escopo para um nome de tabela nela especificado somente se o nome de tabela estiver exposto.
Veja também Especificação de consulta, Tabela e Subconsulta.

5.1.3 Exemplo

Quando a cláusula FROM contém mais de uma tabela, ela especifica uma junção. O exemplo abaixo mostra o nome dos vendedores e a cidade onde trabalham a partir de uma junção.

```
SELECT nome,cidade
FROM vendedores, escritorios
WHERE vendedores.escritorio = escritorios.escritorio
```

O uso de identificadores de correlação é, por exemplo, necessário no caso de uma junção de uma tabela com ela mesma. O exemplo a seguir lista todos os pares de pedidos feitos no mesmo dia.

```
SELECT primeiro.pedido#, segundo.pedido#
FROM pedidos primeiro, pedidos segundo
WHERE primeiro.data = segundo.data
```

5.2 Cláusula WHERE

5.2.1 Sintaxe

cláusula_WHERE : WHERE condição

5.2.2 Regras

O resultado da cláusula WHERE é uma tabela contendo linhas sobre a qual o resultado da condição é verdadeiro. Quando a cláusula está inserida numa especificação de consulta, as linhas são oriundas das tabelas especificadas na cláusula FROM precedente. Se a cláusula está inserida num comando DELETE ou UPDATE, ela se refere à tabela indicada no comando.
Cada especificação de coluna diretamente contida na condição deve ser uma referência não ambígua a uma coluna da tabela original ou deve ser uma referência externa.
Se uma expressão diretamente contida na condição de seleção for uma função, então a cláusula WHERE deve estar contida em uma cláusula HAVING e a referência à coluna na função deve ser uma referência externa.
Veja também Condição, Função, Especificação de consulta, Cláusula FROM e Cláusula HAVING.

5.3 Cláusula GROUP BY

5.3.1 Sintaxe

cláusula_GROUP_BY : GROUP BY coluna [, coluna] …

5.3.2 Regras

A cláusula GROUP BY retorna uma tabela agrupada pelos valores das colunas listadas na cláusula, chamadas de colunas de grupamento. Em cada grupo, todos os valores de cada coluna de grupamento são iguais.

Quando uma condição ou uma expressão é aplicada a um grupo de linhas, uma referência a uma coluna de grupamento é uma referência ao valor desta coluna no grupo.
Para o GROUP BY, dois valores nulos de uma coluna são considerados iguais entre si e diferentes de todos os valores não nulos desta coluna.
Veja também Especificação de consulta.

5.3.3 Exemplo

O exemplo abaixo apresenta a média das vendas de cada vendedor. A coluna vendedor é a coluna de grupamento e a tabela retornada pela cláusula GROUP BY tem tantas linhas quanto o número de vendedores.

```
SELECT vendedor, AVG(preço_total) AS Media_de_Vendas
FROM pedidos
GROUP BY vendedor
```

5.4 Cláusula HAVING

5.4.1 Sintaxe

```
cláusula_HAVING : HAVING condição
```

5.4.2 Regras

O resultado da cláusula HAVING é a tabela agrupada contendo os grupos resultantes da cláusula precedente que aplicando-se a condição, o resultado é verdadeiro.
Se a cláusula precedente não for GROUP BY, a cláusula HAVING considera o resultado desta como um único grupo contendo toda a tabela resultante da cláusula. Neste caso não há coluna de grupamento.
Veja também Especificação de consulta.

5.4.3 Exemplo

O exemplo a seguir mostra o número dos vendedores que venderam mais de $200000.00 e o valor médio de suas vendas.

```
SELECT vendedor, AVG(preço_total)
FROM pedidos
GROUP BY vendedor
HAVING SUM (preço_total) > 200000.00
```

5.5 Cláusula SELECT

5.5.1 Sintaxe

```
cláusula_SELECT : SELECT [ALL | DISTINCT] lista_SELECT
lista_SELECT : * | item_lista [ , item_lista ]…
item_lista :     expressão [AS identificador_de_coluna] | tabela.*
```

5.5.2 Regras

A cláusula SELECT especifica as colunas da tabela a ser retornada numa especificação de consulta. Um novo identificador de coluna pode ser atribuído às colunas da tabela resultante com o uso da cláusula AS. O novo identificador pode ser usado para referenciar colunas da tabela resultante na cláusula ORDER BY.
A cláusula SELECT é aplicada sobre a tabela resultante das cláusulas FROM e opcionalmente das cláusulas WHERE, GROUP BY e HAVING.
Se a cláusula GROUP BY tiver sido aplicada, a tabela é dita agrupada.
A cláusula SELECT * é equivalente a uma lista SELECT que contenha todas as colunas que constituem a tabela retornada pela cláusula FROM na ordem em que a tabela foi definida. O item tabela.*, da mesma forma, especifica todas as colunas na tabela indicada.
Se a cláusula SELECT for aplicada a uma tabela não agrupada e a lista SELECT incluir uma função, então o resultado da especificação de consulta é uma tabela de uma única linha contendo em suas colunas os valores resultantes da avaliação das funções correspondentes aplicadas à tabela original.
Se a lista SELECT não incluir funções, cada expressão é avaliada para cada linha da tabela original. A tabela resultante tem a mesma cardinalidade (número de linhas) da primeira e suas colunas contêm os valores derivados das expressões correspondentes.
Se a cláusula SELECT é aplicada a uma tabela agrupada em um ou mais grupos, a tabela resultante tem cardinalidade igual ao número de grupos e cada linha corresponde à aplicação das expressões à cada grupo.

Se a opção DISTINCT é especificada, o resultado da especificação de consulta retornará a tabela definida pela cláusula SELECT com todas as linhas duplicadas eliminadas. Duas linhas são consideradas duplicadas se os valores em todas as colunas correspondentes forem iguais. Dois valores nulos são considerados iguais e são diferentes de quaisquer outros valores.
Se o resultado das cláusulas subsequentes for uma tabela agrupada, as referências às colunas presentes em cada expressão devem ser referências a colunas de grupamento (especificadas em uma cláusula GROUP BY) ou estar dentro de uma função.
Se o resultado das cláusulas subsequentes não for uma tabela agrupada e alguma expressão incluir uma função, então todas as referências a uma coluna devem estar numa função e o resultado da especificação de consulta é uma tabela de uma única linha.
Veja também Especificação de consulta e Comando SELECT dinâmico.

5.6 Cláusula ORDER BY

5.6.1 Sintaxe

cláusula_ORDER_BY :
ORDER BY especificação_de_ordem [,especificação_de_ordem]…
especificação_de_ordem : {coluna | inteiro} [ASC|DESC]

5.6.2 Regras

A cláusula ORDER BY ordena as linhas de uma tabela.
As especificações de ordem são referências a colunas da tabela resultante ou números inteiros que indicam a posição da coluna na definição da tabela. Os números inteiros devem ser maior que zero e menor ou igual ao grau da tabela.
As linhas são ordenadas pela primeira especificação de ordem. A seguir, as linhas são reordenadas por cada uma das demais especificações de ordem dentro do conjunto de linhas cuja especificação de ordem anterior têm valores iguais para as colunas especificadas.
Se a opção DESC é especificada, a tabela é ordenada em ordem decrescente. Caso contrário, a cláusula indica uma ordem ascendente.
Valores nulos são menores que todos os valores não nulos de uma coluna.
Veja também Cursor.

6 Comandos SQL

6.1 Comando SELECT dinâmico

6.1.1 Sintaxe

comando_SELECT_dinâmico :
expressão_de_consulta
[cláusula_ORDER_BY | cláusula_FOR_UPDATE]
cláusula_FOR_UPDATE : FOR UPDATE OF { * | coluna [, coluna]… }
expressão_de_consulta :
{ especificação_de_consulta |
(expressão_de_consulta) }
[UNION [ALL]
{ especificação_de_consulta |
(expressão_de_consulta) }]…
especificação_de_consulta :
cláusula_SELECT
cláusula_FROM
[cláusula_WHERE]
[cláusula_GROUP_BY]
[cláusula_HAVING]

6.1.2 Regras

O comando SELECT dinâmico retorna a tabela definida pela expressão de consulta, opcionalmente ordenada pela cláusula ORDER BY. A tabela será dita atualizável, se a especificação de cursor for atualizável e não for declarada a cláusula ORDER BY.

Se a cláusula ORDER BY não for especificada, as linhas da tabela poderão ser recuperadas em qualquer ordem, com resultado não previsível.
Se a opção ALL não for especificada com o operador UNION, serão eliminadas do resultado as linhas duplicadas.
A cláusula FOR UPDATE tem efeito apenas quando se trata de um SELECT a ser usado numa declaração dinâmica de cursor. Seu uso é obrigatório quando o cursor for ser usado para atualizações posicionadas na tabela. Neste caso somente as colunas especificadas na cláusula poderão ser atualizadas com o comando UPDATE.
Veja também Cursor, Cláusulas, Expressão de Consulta e Especificação de consulta.

6.2 Comando INSERT

6.2.1 Sintaxe

comando_INSERT :
INSERT INTO tabela
[(identificador_de_coluna [, identificador_de_coluna]…)]
{ lista_de_valores | expressão_de_consulta }
lista_de_valores : VALUES ({expressão | NULL} [, {expressão | NULL}]…)
expressão_de_consulta :
{ especificação_de_consulta |
(expressão_de_consulta) }
[UNION [ALL]
{ especificação_de_consulta |
(expressão_de_consulta) }]…
especificação_de_consulta :
cláusula_SELECT
cláusula_FROM
[cláusula_WHERE]
[cláusula_GROUP_BY]
[cláusula_HAVING]

6.2.2 Regras

Este comando insere uma ou mais linhas em uma tabela atualizável. A tabela não pode aparecer em nenhuma cláusula FROM na expressão de consulta.
Nenhuma coluna da tabela pode aparecer mais de uma vez na lista de colunas. Se a lista de identificadores de coluna for omitida, ficam implicitamente especificadas todas as colunas da tabela na ordem em que aparecem na definição da tabela.
O número de valores relacionados na cláusula VALUES ou o grau da tabela (número de colunas) resultante da expressão de consulta deve ser igual ao tamanho da lista de colunas implícita ou explicitamente especificados. A lista de valores na cláusula VALUES não deve conter funções de grupamento ou especificações de coluna.
Cada nova linha da tabela contém os valores explicitamente especificados no comando e valores nulos para as colunas não identificadas na lista.
Se o tipo de um valor a ser inserido for cadeia de caracteres de tamanho menor que a cadeia da coluna na tabela, o valor será completado com caracteres brancos. Se for de tamanho maior, nenhuma linha será inserida e o parâmetro SQLCODE indicará o erro. O mesmo acontece se o tipo da coluna for numérico exato e não houver uma representação do valor a ser inserido sem que haja perda de dígitos significativos de sua parte inteira.
Se uma coluna é do tipo DATETIME, o valor a ela correspondente deve ser do mesmo tipo com precisão de mesmo campo inicial. Se sua precisão for diferente da precisão do valor correspondente à coluna, os campos excedentes são truncados ou os campos adicionais são iniciados com o valor 1 para os campos mês e dia e com o valor 0 para os campos hora, minuto e segundo.
A cardinalidade da tabela resultante da expressão de consulta é o número de linhas a serem inseridas. Se esta tabela estiver vazia, o parâmetro SQLCODE receberá o valor 100 e não será feita nenhuma inserção.
Se o valor a ser inserido numa coluna em cuja definição não seja permitido valores nulos for nulo, nenhuma linha será inserida e o parâmetro SQLCODE indicará o erro.
Veja também Tabela, Valor , Especificação de consulta e SQLCODE.

6.2.3 Exemplos

O comando abaixo insere na tabela de clientes a empresa Arruda Associados com o número 1111 e com o vendedor de contato especificado por 101. O crédito deste cliente ainda não foi definido e recebe o valor nulo.

```
INSERT INTO clientes (cód_cliente, nome, contato)
VALUES (1111, 'Arruda Associados', 101)
```

O comando abaixo insere uma linha na tabela produtos. Valores para todas as colunas são especificados na ordem em que as colunas foram definidas.

```
INSERT INTO produtos
VALUES ('CAT', '_w234', 'Parafuso tipo B2', 10.50, 500)
```

6.3 Comando DELETE selecionado

6.3.1 Sintaxe

```
comando_DELETE_selecionado :
    DELETE FROM tabela
    [ cláusula_WHERE ]
```

6.3.2 Regras

O comando DELETE selecionado é usado para eliminar uma ou mais linhas de uma tabela atualizável. A cláusula WHERE identifica as linhas da tabela a serem eliminadas (todas sobre as quais a condição é verdadeira). Se não for especificada uma cláusula WHERE, todas as linhas da tabela são eliminadas.
A cláusula WHERE não pode conter uma subconsulta que faça referência à tabela a ser atualizada.
Se nenhuma linha for eliminada, o parâmetro SQLCODE retornará o valor 100.
Veja também Cláusula WHERE, Tabela , DELETE POSICIONADO e SQLCODE.

6.3.3 Exemplo

O exemplo abaixo apaga todos os pedidos feitos no dia 28 de fevereiro de1992.

```
DELETE pedidos
WHERE data = DATETIME <28/02/1992>
```

6.4 Comando UPDATE selecionado

6.4.1 Sintaxe

```
comando_UPDATE_selecionado :
    UPDATE tabela SET atribuição [ , atribuição ]…
    [ cláusula_WHERE ]
atribuição : identificador_de_coluna = { expressão | NULL }
```

6.4.2 Regras

O comando UPDATE selecionado atualiza uma ou mais linhas de uma tabela selecionadas pela condição da cláusula WHERE. Se esta condição não for especificada, todas as linhas da tabelas serão modificadas.
A tabela deve ser atualizável e não pode aparecer em nenhuma cláusula FROM de qualquer subconsulta presente na cláusula WHERE.
A avaliação das expressões é feita antes da alteração do valor de qualquer coluna da linha. As expressões de atribuição não devem conter funções. Cada coluna só deve aparecer uma única vez na cláusula SET.
Um valor nulo só será atribuído a uma coluna se sua definição assim o permitir. Caso contrário, nenhuma linha será atualizada e o parâmetro SQLCODE indicará o erro.
Se o tipo de um valor a ser atualizado for cadeia de caracteres de tamanho menor que a coluna da tabela, o valor será completado com caracteres brancos. Se for de tamanho maior, nenhuma linha será atualizada e o parâmetro SQLCODE indicará o erro. O mesmo acontece se o tipo da coluna for numérico exato e não houver uma representação do valor a ser atualizado sem que haja perda de dígitos significativos de sua parte inteira.
Se o tipo de uma coluna é DATETIME de precisão diferente da precisão do valor correspondente, então os campos excedentes são truncados ou os campos adicionais são iniciados com o valor 1 para os campos mês e dia e com o valor 0 para os campos hora, minuto e segundo.
Veja também Tabela, Cláusula WHERE , Expressão e SQLCODE.

6.4.3 Exemplos

A atualização a seguir aumenta em 20 por cento o crédito de todos os clientes da tabela.

```
UPDATE clientes
```

SET crédito = crédito * 1.2

A próxima atualização remaneja os vendedores do escritório 023 para o escritório 123, atualizando, também, o nome do gerente.

UPDATE vendedores
SET gerente = 'João Marinho', escritório = 123
WHERE escritório = 023

6.5 Comando DELETE posicionado

6.5.1 Sintaxe

comando_DELETE_posicionado :
DELETE FROM tabela
WHERE CURRENT OF identificador_de_cursor

6.5.2 Regras

Este comando elimina da tabela a linha onde o cursor está posicionado.
O cursor deve estar aberto.
A tabela deve ser atualizável e deve ser a mesma referida na especificação de consulta que define o cursor.
O cursor passa a se posicionar imediatamente antes da linha seguinte à eliminada ou no fim da tabela se não houver linha após a eliminada.
Veja também Tabela, Cursor, Especificação de consulta e Comando DELETE selecionado.

6.6 Comando UPDATE posicionado

6.6.1 Sintaxe

comando_UPDATE_ posicionado :
UPDATE tabela SET atribuição [, atribuição]…
WHERE CURRENT OF identificador_de_cursor
atribuição : identificador_de_coluna = { expressão | NULL }

6.6.2 Regras

Este comando altera na tabela os valores das colunas na linha onde o cursor está posicionado.
O cursor deve estar aberto.
A tabela deve ser atualizável e deve ser a mesma referida na especificação de consulta que define o cursor.
A avaliação das expressões é feita antes da alteração do valor de qualquer coluna da linha. As expressões de atribuição não devem conter funções. Cada coluna só deve aparecer uma única vez na cláusula SET.
Um valor nulo só será atribuído a uma coluna se sua definição assim o permitir. Caso contrário, a linha não será atualizada e o parâmetro SQLCODE indicará o erro.
Se o tipo de um valor a ser atualizado for cadeia de caracteres de tamanho menor que a coluna da tabela, o valor será completado com caracteres brancos. Se for de tamanho maior, a linha não será atualizada e o parâmetro SQLCODE indicará o erro. O mesmo acontece se o tipo da coluna for numérico exato e não houver uma representação do valor a ser atualizado sem que haja perda de dígitos significativos de sua parte inteira.
Se o tipo de uma coluna é DATETIME de precisão diferente da precisão do valor correspondente, então os campos excedentes são truncados ou os campos adicionais são iniciados com o valor 1 para os campos mês e dia e com o valor 0 para os campos hora, minuto e segundo.
Veja também Tabela, Cursor, Expressão , Comando UPDATE selecionado e SQLCODE.

6.7 Comando COMMIT

6.7.1 Sintaxe

comando_COMMIT : COMMIT WORK

6.7.2 Regras

Este comando termina normalmente uma transação, efetuando todas as alterações feitas no banco de dados e fechando todos os cursores abertos.
Veja também Transação e Cursor.

6.8 Comando ROLLBACK

6.8.1 Sintaxe

comando_ROLLBACK : ROLLBACK WORK

6.8.2 Regras

O comando ROLLBACK termina a transação corrente. Todas as alterações feitas na base de dados durante a transação são canceladas.
Todos os cursores abertos são automaticamente fechados.
Veja também Transação e Cursor.

6.9 Comando LOCK TABLE

6.9.1 Sintaxe

comando_LOCK_TABLE : LOCK TABLE tabela IN { SHARE | EXCLUSIVE } MODE

6.9.2 Regras

O comando LOCK TABLE faz uma reserva de tabela até o final da transação.
Quando uma transação faz uma reserva com a opção SHARE, é feita uma reserva compartilhada e outras transações podem ler dados da tabela, porém somente a transação que fez a reserva pode atualizar a tabela.
Já quando a reserva é feita com a opção EXCLUSIVE, nenhuma transação diferente da que fez a reserva pode fazer qualquer reserva de linhas da tabela, seja para atualização ou para leitura.
No momento em que a reserva é feita, a tabela não pode estar aberta.
Veja também Tabela e Transação.

6.10 Comando CREATE TABLE

6.10.1 Sintaxe

comando_CREATE_TABLE :
CREATE TABLE tabela
(item_def_tabela [, item_def_tabela]…)
item_def_tabela :
identificador_de_coluna tipo_de_dado [restrição_de_coluna]… |
restrição_de_unicidade |
restrição_referencial |
especificação_de_índice
restrição_de_unicidade :
especificação_de_unicidade
(identificador_de_coluna [, identificador_de_coluna]…)
especificação_de_unicidade : PRIMARY KEY | UNIQUE
restrição_referencial : FOREIGN KEY
(identificador_de_coluna [, identificador_de_coluna]…)
especificação_de_referência
restrição_de_coluna :
NOT NULL |
UNIQUE |
PRIMARY KEY |
INDEX |
especificação_de_referência
especificação_de_referência : REFERENCES tabela [ação_gatilho]
ação_gatilho : regra_UPDATE [regra_DELETE] |
regra_DELETE [regra_UPDATE]
regra_UPDATE : ON UPDATE { CASCADE | SET NULL }
regra_DELETE : ON DELETE { CASCADE | SET NULL }
especificação_de_ índice: INDEX
(identificador_de_coluna [, identificador_de_coluna] …)

6.10.2 Regras

O comando CREATE TABLE define uma nova tabela cujo nome deve ser diferente de qualquer outro definido no mesmo esquema. Deve-se especificar o nome e o tipo de dados de todas as colunas da tabela, sendo que duas colunas de uma tabela não podem ter o mesmo nome.

Não podem ser definidas colunas do tipo VARCHAR.
A restrição NOT NULL na definição de uma coluna impede a existência de valores nulos em qualquer de suas linhas. Esta restrição é considerada por todos os comandos SQL.
A definição de uma tabela pode conter uma restrição de unicidade UNIQUE ou PRIMARY KEY, obrigando cada linha a ser criada a ter, nas colunas especificadas , uma combinação de valores única, diferente de todas as outras combinações presentes na tabela. Duas combinações são consideradas iguais se o valor de cada coluna na primeira linha não é nulo e é igual ao valor da coluna correspondente na segunda linha.
Apenas uma especificação de unicidade PRIMARY KEY pode ser definida na tabela. A restrição NOT NULL será automaticamente considerada para todas as colunas desta chave.
Se existe uma especificação de unicidade UNIQUE ou PRIMARY KEY em uma definição de coluna, então é implícita uma restrição de unicidade formada pela especificação de unicidade e o nome da coluna.
A especificação de uma restrição referencial está subordinada às seguintes regras:

- a definição da tabela de referência deve conter uma especificação de unicidade PRIMARY KEY;
- cada nome de coluna da restrição referencial deve identificar uma coluna da tabela e deve ser diferente de qualquer outro na mesma definição;
- a lista de colunas deve ser de tamanho igual à lista de colunas da PRIMARY KEY da tabela de referência;
- as colunas correspondentes nas duas listas devem ser do mesmo tipo.

Se é especificada uma regra DELETE (ou UPDATE) SET NULL, então nenhuma coluna da lista de colunas que define a restrição referencial deve ser restrita à valores não nulos.
Se, na definição de uma tabela, duas restrições referenciais fazem referência à mesma tabela e as listas de colunas que definem as restrições não são disjuntas, então em nenhuma delas pode ser especificada uma regra UPDATE e somente a regra DELETE CASCADE pode ser especificada.
A restrição referencial é verificada automaticamente para cada comando SQL. Ela é satisfeita se, alguma coluna da restrição tem valor nulo, ou todas as colunas da restrição contem valores não nulos e estes são iguais aos valores das colunas de referência correspondentes, em alguma linha da tabela de referência (linha casada).
Se uma regra DELETE é especificada em uma restrição referêncial, então quando uma linha da tabela de referência é eliminada:

- se é especificada a regra DELETE CASCADE, todas as linhas casadas são eliminadas;
- se é especificada a regra DELETE SET NULL, em todas as linhas casadas as colunas que definem a restrição recebem o valor nulo.

Se uma regra UPDATE é especificada em uma restrição referêncial, então quando, em uma linha da tabela de referência, é alterado o valor de alguma coluna da chave primária:

- se é especificada a regra UPDATE CASCADE, então, em todas as linhas casadas, a coluna correspondente na definição da restrição recebe o mesmo valor;
- se é especificada a regra UPDATE SET NULL, então, em todas as linhas casadas, a coluna correspondente na definição da restrição recebe o valor nulo.

Se existe uma especificação de referência em uma definição de coluna, então é implícita uma restrição referencial formada pela mesma coluna e a especificação de referência.
Uma especificação de índice define sobre a tabela um índice.
Se existe uma restrição de coluna INDEX em uma definição de coluna, então é implícita uma especificação de índice cuja chave é formada pela coluna.
Veja também Tabela e Tipo de Dado.

6.10.3 Exemplo

A seguir apresenta-se a definição da tabela clientes. A coluna cod_cliente é a chave primária da tabela.

```
CREATE TABLE clientes
(cod_cliente INTEGER NOT NULL,
nome CHARACTER (30) NOT NULL,
vendedor INT,
credito NUMERIC (10,2),
PRIMARY KEY (cod_cliente))
```

6.11 Comando CREATE INDEX

6.11.1 Sintaxe

comando_CREATE_INDEX :
CREATE [UNIQUE] INDEX identificador_de_índice
ON tabela (identificador_de_coluna[,identificador_de_coluna]…)

6.11.2 Regras

O comando CREATE INDEX define um índice sobre colunas de uma tabela anteriormente definida. O identificador do índice deve ser único em um esquema. Cada coluna só pode aparecer uma vez na definição do índice e deve identificar uma das colunas da tabela.
A opção UNIQUE impede a existência de linhas duplicadas para os valores das colunas componentes do índice. Se esta opção é especificada, nenhuma coluna deve permitir valores nulos.
Veja também Tabela.

6.12 Comando DROP TABLE

6.12.1 Sintaxe

comando_DROP_TABLE : DROP TABLE tabela

6.12.2 Regras:

O comando DROP TABLE elimina uma tabela anteriormente definida.
Veja também Tabela .

6.13 Comando DROP INDEX

6.13.1 Sintaxe

comando_DROP_INDEX : DROP INDEX identificador_de_índice

6.13.2 Regras

O comando DROP INDEX elimina um índice anteriormente definido no esquema.
Veja também CREATE INDEX.

7 Interface nativa de programação

7.1 Introdução

A interface nativa de programação é uma biblioteca de funções através das quais uma aplicação se conecta a uma base de dados, efetua pedidos para a execução de comandos SQL e recupera dados. Por exemplo o utilitário tsqlwi utiliza a interface nativa para executar comandos SQL. Por sua vez o driver ODBC do TSQL traduz a interface ODBC para a interface nativa e desta forma programas ODBC se conectam com uma base OpenBase.
Como esta interface possui alguma complexidade e que a maioria dos usuários do TSQL desenvolvem aplicações em linguagens de maior nível que a linguagem C como Visual Basic, Delphi e outras, a documentação da interface nativa não é apresentada neste manual. Para obte-la deve-se entrar em contato com a Tecnocoop.
No entanto, apresentamos a seguir uma interface simplificada implementada sobre a interface nativa. Ela está sujeita a algumas restrições:

- somente um cursor pode ser aberto de cada vez;
- o cursor só pode se movimentar uma linha para frente a cada leitura;
- valores de tipo BLOB não podem ser manipulados.

7.2 Inferface simplificada

7.2.1 tsqlDBEXTNAME

int tsqlDBEXTNAME(char datasource, char *hostname, int instancia)

Cria uma conexão com uma base. No caso da conexão já existir com o número da instância especificado, ela se torna a corrente.
O valor retornado pela função é igual a menos um (–1), em caso de erro, ou maior ou igual a zero (>=0) identificando o número da conexão.

Parâmetros de entrada:
datasource: Cadeia que especifica uma base como descrito no Apêndice 3.
hostname: Nome da máquina onde reside a base. Pode ser também especificado no parâmetro datasource. O parâmetro hostname tem prioridade.
instancia: Permite diferenciar duas ou mais conexões com a mesma base.

7.2.2 tsqlPREPARE

int tsqlPREPARE(char *statement, int *sqlcode)

Prepara (compila) um comando SQL.
O valor retornado pela função é igual a menos um (–1), em caso de erro, ou igual a zero (0).

Parâmetro de entrada:
statement: Cadeia que contém o comando SQL. O parâmetro deve conter um dos comandos SQL listados abaixo:

comando_COMMIT |
comando_DELETE_selecionado |
comando_DELETE_posicionado |
comando_INSERT |
comando_ROLLBACK |
comando_SELECT_dinâmico |
comando_UPDATE_selecionado |
comando_UPDATE_posicionado |
comando_CREATE_TABLE |
comando_DROP_TABLE |
comando_CREATE_INDEX |
comando_DROP_INDEX

Parâmetro de saída:
sqlcode: Código de erro. Zero se sucesso.

7.2.3 tsqlDESCRIBE

int tsqlDESCRIBE(struct SQLDA *sqlda, int *sqlcode)

Retorna informações sobre o comando SQL preparado.
O valor retornado pela função é igual a menos um (–1), em caso de erro, ou igual a zero (0).

Parâmetro de entrada:
sqlda: Estrutura que contem informações sobre o comando preparado. Se o número de valores associados ao comando (SQLD) for zero o comando não é um comando SELECT. A utilização da estrutura pela rotina tsqlDESCRIBE é descrita a seguir:

```
struct SQLDA {
short SQLN;
short SQLD;
struct SQLVAR *SQLVAR;
}
struct SQLVAR {
char *SQLDATA;
long *SQLIND;
short SQLTYPE;
short SQLNULL;
short SQLLEN;
short SQLSCALE;
struct {
```

```
short SQLNAMEL;
char SQLNAMEC[18];
} SQLNAME;
        }
```

O campo SQLN deve especificar o número de estruturas SQLVAR. O campo SQLD retorna zero se o comando preparado não for um comando SELECT .
Se o comando for um SELECT, SQLD indica o número de colunas da tabela por ele especificada. Se SQLD for maior que SQLN, não serão retornadas as informações sobre as colunas da tabela. Caso contrário serão retornadas nas primeiras estruturas SQLVAR a descrição das colunas correspondentes na tabela.
No campo SQLTYPE é retornado um valor numérico indicando o tipo da coluna segundo a tabela abaixo.

Tipo da Coluna	Valor Numérico
CHARACTER	1
NUMERIC	3
DECIMAL	3
INTEGER	4
SMALLINT	5
FLOAT	7 OU 8
REAL	7
DOUBLE	8
DATETIME	9
VARCHAR	12

No campo SQLNULL é retornado o valor 1 se a coluna puder ter valores nulos.
No campo SQLLEN é retornado o tamanho, em “bytes”, da coluna, em função de seu tipo:

- se o tipo é CHARACTER, é retornado o tamanho da coluna;
- se o tipo é NUMERIC ou DECIMAL, é retornada a precisão da coluna mais 1.
- se o tipo é INTEGER ou REAL, é retornado 4.
- se o tipo é SMALLINT, é retornado 2.
- se o tipo é FLOAT, é retornado 4 ou 8, segundo a sua precisão.
- se o tipo é DOUBLE, é retornado 8.
- se o tipo é DATETIME, é função da precisão da coluna da seguinte forma:
- 2 “bytes” para o campo dia (de 01 a 31).
- 1 “byte” para o separador “/” entre os campos dia e mês.
- 2 “bytes” para o campo mês (de 01 a 12).
- 1 “byte” para o separador “/” entre os campos mês e ano.
- 5 “bytes” para o campo ano (de 9999- a 9999, ajustado a esquerda e completado com caractes espaço).
- 1 “byte” para o separador “ “ entre o campo dia, mês ou ano e o campo hora.
- 2 “bytes” para o campo hora (de 00 a 23).
- 1 “byte” para o separador “:” entre os campos hora e minuto.
- 2 “bytes” para o campo minuto (de 00 a 59).
- 1 “byte” para o separador “:” entre os campos minuto e segundo.
- 2 “bytes” para o campo segundo (de 00 a 59).
- . se o tipo é VARCHAR, em SQLLEN é retornado o valor 0.

Em SQLSCALE é retornada a escala do tipo NUMERIC ou DECIMAL ou a precisão do tipo DATETIME. Esta é representada numericamente pelo valor numérico do campo inicial somado com o do final, se são diferentes. Os valores de cada campo são:

Ano	64
Mês	32
Dia	16
Hora	8
Minuto	4

Segundo	2

Em SQLNAMEC é retornado o nome da coluna. Se este tem menos de 18 caracteres, o nome é completado com caracteres brancos.
Em SQLNAMEL é retornado o tamanho do nome da coluna que será igual a 18, se a coluna tiver nome ou igual a 0, se a coluna não tiver nome.
A estrutura SQLDA encontra-se declarada no arquivo \usr\include\sqlda.h que pode ser incluído no programa de aplicação.

Parâmetro de saída:
sqlcode: Código de erro. Zero se sucesso.

7.2.4 tsqlEXECUTE

int tsqlEXECUTE(int *sqlcode)

Descrição
Executa o comando SQL preparado se o comando não é do tipo SELECT.

Parâmetro de saída:
sqlcode: Código de erro. Zero se sucesso.

Retorno:
0 : sucesso.
-1 : erro

7.2.5 tsqlIMMEDIATE

int tsqlIMMEDIATE(char *statement, int *sqlcode)

Descrição:
Prepara e executa um comando SQL. O comando não pode ser do tipo SELECT.

Parâmetro de entrada:
statement : cadeia que contém o comando SQL. Ver a descrição da chamada tsqlPREPARE para os comandos permitidos.

Parâmetro de saída:
sqlcode: Código de erro. Zero se sucesso.

Retorno:
0 : sucesso.
-1 : erro

7.2.6 tsqlOPEN

int tsqlOPEN(int *sqlcode)

Descrição:
Executa o comando SELECT preparado e associa implicitamente o cursor ao conjunto resultado. O cursor é posicionado imediatamente antes do inicio da primeira linha. Diz-se que o cursor está no estado aberto. O nome do cursor não é relevante nesta interface considerando que êle é único.

Parâmetro de saída:
sqlcode: Código de erro. Zero se sucesso.

Retorno:
0 : sucesso. O cursor foi aberto.
-1 : erro

7.2.7 tsqlFETCH

int tsqlFETCH(struct SQLDA *sqlda, int *sqlcode)

Descrição:
Move o cursor para a próxima linha do conjunto resultado e recupera a linha colocando os valores das colunas na estrutura apontada por sqlda. A rotina tsqlFETCH só pode ser chamada se o cursor estiver aberto.

Parâmetros de saída:
sqlcode: Código de erro. Zero se sucesso.
sqlda: Estrutura que contem os valores retornados.

```
struct SQLDA {
short SQLN;
short SQLD;
struct SQLVAR *SQLVAR;
}
struct SQLVAR {
char *SQLDATA;
long *SQLIND;
short SQLTYPE;
short SQLNULL;
short SQLLEN;
short SQLSCALE;
struct {
short SQLNAMEL;
char SQLNAMEC[18];
} SQLNAME;
}
```

O parâmetro corresponde ao campo SQLN e especifica o número de estruturas SQLVAR retornadas. SQLD corresponde ao grau da tabela e deve ser menor ou igual a SQLN. As sucessivas estruturas SQLVAR conterão as descrições das variáveis no programa para as quais retornarão os valores das colunas.
O campo SQLTYPE deve conter um valor numérico indicando o tipo da coluna segundo a tabela abaixo.

Tipo da Coluna	Valor Numérico
CHARACTER	1
NUMERIC	2
DECIMAL	3
INTEGER	4
SMALLINT	5
FLOAT	6
REAL	7
DOUBLE	8
DATETIME	9
VARCHAR	12

O campo SQLNULL é igual a 1 se o valor retornado pode ser nulo. O campo SQLLEN é equivalente ao tamanho da variável. O campo SQLSCALE contém a escala do tipo numérico caso a variável seja desse tipo. O campo SQLNAME não é usado.
SQLDATA aponta para a área da memória para onde é retornado o valor da variável. SQLIND aponta para a área da memória onde é retornada o valor –1 se SQLNULL for igual a 1 e se a coluna correspondente tiver valor nulo; o tamanho do valor da coluna se este for uma cadeia de caracteres de tamanho maior que o tipo da variável ou nos demais casos, retorna 0.

SQLDATA aponta para a área da memória para onde é retornado o valor da variável. SQLIND aponta para a área da memória onde é retornada a variável INDICATOR, se SQLNULL é igual a 1.

Os tipos das variáveis devem corresponder aos tipos das colunas correspondentes na tabela. Se ocorrer um erro durante a atribuição de valores às variáveis, o parâmetro sqlcode retornará um valor negativo.
Se o tipo de uma variável é DATETIME de precisão diferente da precisão da coluna correspondente, os campos excedentes são truncados ou os campos adicionais são iniciados com o valor 1 para os campos mês e dia e com o valor 0 para os campos hora, minuto e segundo.
Caso a tabela associada ao cursor esteja vazia ou o cursor estiver na última linha, o cursor será posicionado após a última linha, o valor 100 será retornado no parâmetro sqlcode e não serão atribuídos valores às variáveis do programa de aplicação.

sqlcode :
< 0 se erro
= 100 se não há mais linhas a retornar
= 0 caso contrário

Retorno:
0 : se sqlcode >= 0
-1 : se sqlcode < 0

7.2.8 tsqlCLOSE

int tsqlCLOSE(int *sqlcode)

Descrição:
Fecha o cursor . A chamada tsqlFETCH não pode mais ser executada até que um outro cursor seja aberto.

Parâmetro de saída:
sqlcode: Código de erro. Zero se sucesso.

Retorno:
0 : sucesso. O cursor foi fechado.
-1 : erro

7.2.9 tsqlKILL

int tsqlKILL(int num)

Descrição:
Termina uma conexão.

Parâmetro de entrada:
num : número da conexão.

Retorno:
0 : sucesso
-1 : erro ao fechar a conexão

7.2.10 tsqlERROR

char *tsqlERROR(char *endereco)

Descrição:
A rotina tsqlERROR gera a mensagem de erro relativa á última chamada efetuada.
A mensagem é terminada com o caracter ‘\0’ e possui um tamanho máximo de 120 bytes.

Parâmetro de entrada :

endereco : endereço da mensagem. Se for passado com o valor (CHAR *) 0L, a mensagem é construída em uma área interna da rotina, que é alterada a cada nova chamada.

Retorno:
Endereço da mensagem

Veja também SQLCODE.

8 O Catálogo SQL

Um sistema de gerenciamento de banco de dados deve ter acesso a um conjunto de informações sobre a estrutura de uma base para poder gerencia-la. Em uma base relacional estas informações são armazenadas no catálogo. O catálogo é geralmente um esquema composto de um conjunto de tabelas que o gerenciador do banco mantém para o seu próprio uso, e descreve as tabelas, colunas, índices, etc.
Como uma base OpenBase não possui fisicamente tabelas de um catálogo SQL, o TSQL simula internamente a execução de consultas a estas tabelas sendo que o identificador de autorização do esquema catálogo para o OpenBase é o identificador SYSSQL. Por exemplo a consulta **select * from SYSSQL.SYSTABLES** devolve as informações sobre todas as tabelas de uma base.

Todas as colunas de todas as tabelas do catálogo são restritas a conter somente valores não nulos.

8.1 SYSCOLUMNS

Armazena informações sobre colunas de tabelas. Cada linha de SYSCOLUMNS corresponde a uma coluna de uma tabela.
As colunas de SYSCOLUMNS são:

SCHNAME - CHAR (18)
= identificador de autorização do esquema

TNAME - CHAR (18)
= identificador da tabela

COLNO - SMALLINT
= identificador numérico da coluna na tabela

COLNAME - CHAR (18)
= nome da coluna

COLTYPE - CHAR (18)
tipo da coluna:
= “CHARACTER”,
= “DECIMAL”,
= “INTEGER”,
= “SMALLINT”,
= “REAL”,
= “DOUBLE PRECISION”,
= “DATETIME”,
= “CHAR VARYING”
= “BLOB” ou
= “CLOB”

SYSTYPE - SMALLINT
= 1 (cadeia de caracteres de tamanho fixo),
= 3 (numérico decimal de representação exata e ponto decimal fixo),
= 4 (numérico binário inteiro longo),
= 5 (numérico binário inteiro curto),
= 7 (numérico binário curto de representação não exata e ponto flutuante),
= 8 (numérico binário longo de representação não exata e ponto flutuante),
= 9 (data-hora)

= 12 (cadeia de caracteres de tamanho variável terminada pelo caracter '\0')
= 30 (cadeia binária de tamanho longo) ou
= 40 (cadeia de caracteres de tamanho longo)

CLENGTH - SMALLINT

= tamanho máximo para colunas do tipo cadeia de caracteres de tamanho limitado ou
= precisão para colunas do tipo numérico não binário ou
= precisão para colunas do tipo data-hora que é representada pela soma entre o start-field e o end-field:

64 (YEAR)
32 (MONTH)
16 (DAY)
8 (HOUR)
4 (MINUTE)
2 (SECOND)
1 (FRACTION)

SCALE - SMALLINT

= escala para colunas do tipo numérico não binário

NULLS - CHAR (1)

= "Y" (pode ter valor nulo) ou
= "N"

FIRSTKEY - CHAR (1)

= "Y" (indica se é primeira coluna da chave de algum índice) ou
= "N"

COLCARD - INTEGER

= número de valores distintos da coluna ou
= -1, se não é válido

HIGH2KEY - CHAR (8)

= contém o segundo maior valor da coluna ou
= " ", se não é válido

LOW2KEY - CHAR (8)

= contém o segundo menor valor da coluna ou
= " ", se não é válido

A chave primária de SYSCOLUMNS é composta pelas colunas TNAME, SCHNAME e COLNO.
O valor da coluna COLCARD só é valido quando ela é a única coluna de um índice.
Os valores HIGH2KEY e LOW2KEY não são válidos na presente versão.

8.2 SYSINDEXES

Armazena informações de cada índice, restrição de unicidade e restrição referencial definida sobre uma tabela.
As colunas de SYSINDEXES são:

SCHNAME - CHAR (18)

= identificador de autorização do esquema

TNAME - CHAR (18)

= identificador da tabela

OBID - INTEGER

= identificador numérico do índice

NAME - CHAR (18)

= identificador do índice ou restrição

TYPE - CHAR (1)

="P" (chave primária)
= "I" (índice criado por CREATE TABLE),
="N" (índice criado por CREATE INDEX)
= "U" (restrição de unicidade) ou
= "E" (restrição referencial)

UNIQUERULE - CHAR (1)

= "Y" (valores duplicados de chave não são permitidos),

= “N” (valores duplicados de chave são permitidos)

COLCOUNT - SMALLINT

= número de colunas que compõem a chave do índice

RSCHEMA – CHAR(18)

= identificador de esquema da tabela referida em restrição referencial

RNAME – CHAR(18)

= identificador de tabela da tabela referida em restrição referencial

ROBID – INTEGER

= identificador numérico do índice da tabela referida em restrição referencial

UPDATERULE – CHAR(1)

= indica a ação decorrente do comando update em restrição referencial:
“R” (RESTRICT),
“C” (CASCADE) ou
“S” (SET NULLS)

DELETERULE – CHAR(1)

= indica a ação decorrente do comando delete em restrição referencial:
“R” (RESTRICT),
“C” (CASCADE) ou
“S” (SET NULLS)

FIRSTKEYCARD – INTEGER

= número de valores distintos da primeira coluna da chave
= -1, se não é válido

FULLKEYCARD - INTEGER

= número de valores distintos da chave do índice ou
= -1, se não é válido

NLEAF - INTEGER

= número de folhas ou
= -1, se não é válido

NLEVELS - SMALLINT

= número de níveis ou
= -1, se não é válido

CLUSTERED - CHAR (1)

= “Y” (tabela está fisicamente ordenada pela chave),
= “N” (tabela não está fisicamente ordenada pela chave) ou
= “ “, se não é válido

A chave primária de SYSINDEXES é composta pelas colunas NAME ,SCHNAME e OBID.

8.3 SYSKEYS

Armazena informações sobre colunas que compõem chaves de índices e restrições de unicidade. Cada linha de SYSKEYS corresponde a uma coluna que compõe uma chave.
As colunas de SYSKEYS são:

SCHNAME - CHAR (18)

= identificador de autorização do esquema

TNAME - CHAR (18)

= identificador da tabela

OBID - INTEGER

= identificador numérico do índice

COLSEQ - SMALLINT

= número de ordem da coluna na definição do índice (maior ou igual a 1)

COLNO - SMALLINT

= identificador numérico da coluna na tabela

ORDERING - CHAR (1)

= “A” (ordem da coluna na chave é ascendente)
= “D” (ordem da coluna na chave é descendente) ou
=”U” (indefinido)

A chave primária de SYSKEYS é composta pelas colunas TNAME, SCHNAME, OBID e COLSEQ.

8.4 SYSTABLES

Armazena informações sobre tabelas . Cada linha de SYSTABLES corresponde a uma tabela .
As colunas de SYSTABLES são:

SCHNAME - CHAR (18)
= identificador de autorização do esquema
TNAME - CHAR (18)
= identificador da tabela
TYPE - CHAR (1)
= "T" (tabela definida pelo comando CREATE TABLE)
VIEWTYPE - CHAR (1)
uso futuro
COLCOUNT - SMALLINT
= número de colunas
SIZE - INTEGER
uso futuro
VTREE - BLOB
uso futuro
CARD - INTEGER
= número total de linhas da tabela ou
= -1, se não é válido
NPAGES - INTEGER
= número de páginas que armazenam linhas da tabela ou
= -1, se não é válido

A chave primária de SYSTABLES é composta pelas colunas TNAME e SCHNAME.

9 Utilitários

9.1 TSQLI

9.1.1 Sintaxe

tsqlI [nomeDB] [-X arquivo | -E arquivo] [-F arquivo]

9.1.2 Regras

Programa que oferece ao usuário uma interface interativa e amigável para a execução de comandos SQL. O utilitário possui:
.Um ambiente para desenvolvimento de comandos SQL
.Um ambiente para desenvolvimento de formulários.
.Um gerador de relatórios com vários níveis de quebras, com totalizações, médias e freqüências.
.Um gerador de telas para consultas e inserções de dados.
.Um gerador de telas de menu, cada opção deste pode ser associada a execução de um programa qualquer, um comando SQL, um submenu ou uma tela para consulta e atualização de dados.
."HELP" on line.
A descrição do utilitário se encontra no manual do usuário.
Opções
nomeDB
O parâmetro nomeDB especifica o diretório da Base de dados (Veja Base de dados e esquemas). O diretório pode ser especificado na forma de um caminho completo ou um caminho em relação ao diretório corrente. Se a opção não é fornecida, é considerada a Base "default"(subdiretório db do diretório corrente).
-X arquivo
Com esta opção o utilitário entra diretamente no modo comando, executando o comando SQL contido em "arquivo". O usuário que entra diretamente neste modo não tem acesso aos demais recursos do sistema.
-E arquivo
O utilitário entra diretamente em modo tela, com a definição da tela contida em "arquivo". O usuário que entra diretamente neste modo não tem acesso aos demais recursos do sistema.

-F arquivo
Com esta opção o usuário especifica como formulário corrente o formulário cuja definição está contida em arquivo. Se a opção não for especificada em conjunto com a opção -X, o resultado da execução do comando SQL será exibido segundo o formulário especificado.
Veja apêndice 4 para maiores informações sobre o TSQL.

9.2 TSQLMAN

9.2.1 Sintaxe

tsqlman [seção] [item]
seção : I | CO | E | P | CL | C | D | L | U

9.2.2 Regras

O utilitário tsqlman é um manual de referência do TSQL on-line. É possível acessar através dele o índice geral, sub-índices de cada capítulo e as entradas do manual propriamente ditas.
As opções são:
seção
Indica seção do manual a ser consultada. Se estiver seguida por item, indica seção ao qual o item pertence, senão mostra o sub-índice relativo àquela seção.
As seções correspondem aos seguintes submenus:

I : Índice	C : Comandos
CO : Conceitos	D : Declarações
E : Elementos	L : Linguagens
P : Predicados	U : Utilitários
CL : Cláusulas	

item
Indica item do manual a ser mostrado. Os possíveis itens estão especificados no índice e nos submenus. Na execução do manual, são válidos todos os comandos usados no pg do sistema. Tecle h na linha de comando (sinalizada por ":") para ver os comandos possíveis.

9.3 TSQLBD

9.3.1 Sintaxe

tsqlbd –b <caminho do banco> [-g<arquivo do esquema>] [-n<nível de acesso>]
[-s<código de segurança>]

9.3.2 Regras

O utilitário tsqlbd tem por objetivo gerar a partir de uma base OpenBase o fonte do esquema SQL correspondente de acordo com a sintaxe seguinte :

esquema : **CREATE SCHEMA**
[AUTHORIZATION identificador_de_autorização]
[definição…]
definição : **comando_CREATE_TABLE**

A construção do fonte obedece ás seguintes regras:

- As entidades e relações da Base *Open*BASE são vistas como tabelas SQL, nas quais os nomes das colunas são os nomes dos itens.
- O caracter'.' em um nome de item é convertido para o caracter '_' no nome da coluna correspondente.
- Se a base Openbase possuir sub-itens de um campo composto não explicitamente declarado o utilitário cria um nome para o sub-item como o texto do exemplo:

nome: clientes e
registro:

cod_cliente (0)	I2
nome	U30
data	U8
ano	U4 POS data

```
mes                U2  POS    data + 4
```

que gera no fonte do esquema SQL correspondente o seguinte texto :

```
CREATE TABLE clientes (
cod_cliente    INTEGER  NOT NULL ,
nome           CHAR(30) NOT NULL ,
ano            CHAR(4)  NOT NULL ,
mes            CHAR(2)  NOT NULL ,
coluna0        CHAR(2)  NOT NULL
PRIMARY KEY (cod_cliente) )
```

Os tipos de dados do esquema são definidos a partir dos tipos da Base *Open*BASE, segundo a seguinte correspondência:

Tipo OpenBase	**tipo sql**
I2	INTEGER
I4	DEC(10)
N	DEC(-)
S	DEC(-)
P	DEC(-)
C	DEC(-)
U , V	CHAR(-)
D2 , D4	DATETIME YEAR TO DAY
D7 , T8	DATETIME YEAR TO SECOND
L	CHAR
F4	REAL
F8	DOUBLE
B1	SMALLINT
B2	SMALLINT
B3	INTEGER
B4	INTEGER
B4 (> 4)	DEC(-)
M , O , Q	BLOB

Uma chave unívoca na Base OpenBASE é definido no esquema SQL por intermédio de uma <definição de restrição de unicidade>. Se a chave for o atributo determinante de uma entidade (chave primária), a <especificação de unicidade> será PRIMARY KEY. De outra forma, será UNIQUE.
No fonte, a declaração de um índice correspondente a uma chave de acesso na Base *Open*BASE, aparece logo após à definição da tabela a qual ele pertence. , sendo que o nome do índice é o nome do item que representa a chave de acesso.
A proteção especificada na base *Open*BASE através das palavras de nível, não é traduzida no fonte do esquema.

As opções do utilitário são:
-b caminho do banco
Especifica o banco OpenBase.
-g arquivo do esquema
Opção que especifica o arquivo no qual é gravado o fonte do esquema com comandos SQL.
-n nível de acesso
Opção que especifica o nível de acesso do banco
-s código de segurança
Opção que especifica o código de segurança do banco

9.4 BDCNFG

9.4.1 Sintaxe

bdcnfg

9.4.2 Regras

O **bdcnfg** é um programa conversacional que configura o tamanho da memória compartilhada entre as diversas aplicações. Esta memória contém a tabela de bloqueios e em determinados ambientes de execução pode ser necessário aumenta-la.
Este utilitário também é utilizado para configurar determinados parâmetros do sistema *Open*BASE que não são relevantes para o **TSQL**.

9.5 BDLICE

9.5.1 Sintaxe

bdlice [-t]

9.5.2 Regras

O utilitário bdlice lista a tabela de bloqueios dos usuários **TSQL** e *Open*BASE.
Cada entrada na tabela contém o nome da base, a identificação do processo, a identificação do arquivo, a identificação da página bloqueada no arquivo e o tipo do bloqueio (compartilhado ou exclusivo). Este utilitário informa as páginas bloqueadas das diversas bases de dados em um determinado instante.
As opções são:
-t
Mostra também o número da última transação.

9.6 BDSGBD

9.6.1 Sintaxe

bdsgbd [-d] [-e] [-a arquivo]

9.6.2 Regras

O bdsgbd é um programa que deve ser executado em "background" antes de ser efetuado qualquer acesso a uma base TSQL ou *Open*BASE (nohup bdsgbd <opções> &). Ele é fundamental para o gerenciamento destas bases.
As opções são:
-d Indica gravação em diário.
-e Indica que o arquivo diário deve ser esvaziado.
-a Indica a designação do arquivo diário para outro arquivo que não o arquivo /usr/tsgbd/tsdic/diario. O arquivo diário deve ser um arquivo em disco.

10 Apêndice 1: Instalação

a) Introdução.

Para poder executar comandos SQL sobre uma base *Open*BASE é necessário instalar o servidor TSQL na máquina que contém a base e instalar na máquina cliente os ambientes clientes.
No WINDOWS os ambientes clientes são:

- programa **tsqlwi** : ambiente interativo para execução de comandos SQL
- driver **ODBC**
- interface nativa TSQL para desenvolvimento de aplicações em C.

No UNIX são :

- programa **tsqlI** : ambiente interativo não gráfico para execução de comandos SQL.
- interface nativa TSQL para desenvolvimento de aplicações em C.

b) Instalação propriamente dita dos diversos módulos:

- **Servidor TSQL no ambiente WINDOWS :**

Executar o programa setup.exe. Se o diretório de instalação não for especificado durante a instalação será considerado o diretório c:\tsql. Esta operação também instala os utilitários e os recursos para o desenvolvimento de aplicações em C.

- **Servidor TSQL no ambiente UNIX :**

Os arquivos contidos na mídia devem ser copiados para um diretório(/usr/tmp por exemplo). Um dos arquivos é o arquivo insTSQL que ativado em modo super-usuário efetua a instalação propriamente dita. No caso de uma reinstalação, a ativação de insTSQL com o parâmetro -R abrevia o tempo de instalação. O servidor é instalado no diretório /usr/lib/tsql.
Esta operação também instala os utilitários do sistema incluindo o programa tsqlI e os recursos para o desenvolvimento de aplicações em C.

- **Cliente TSQL no ambiente WINDOWS**

A execução do programa setup.exe instala o programa tsqlwi e o ambiente C para o desenvolvimento de aplicações. O diretório pode ser especificado durante a instalação.

- **ODBC no ambiente WINDOWS**

Executar o programa setup.exe contido no primeiro disco de instalação. O driver ODBC para o sistema TSQL é representado pela DLL tsql32.dll.

O uso do servidor TSQL pressupõe que o OpenBase , na sua versão básica, esteja instalado na máquina servidora.

11 Apêndice 2: Ativação do servidor Tsql

A comunicação com o servidor TSQL é baseada no protocolo TCP/IP.
Para que uma ligação cliente/servidor possa ser efetuada é preciso que o programa xsqlserv que se encontra no diretório de instalação do servidor TSQL esteja ativado. No caso do WINDOWS a ativação é realizada com a simples execução do programa. No caso do UNIX a ativação deve ser efetuada em modo super-usuário executando o seguinte comando “nohup /usr/lib/tsql/xsqlserv &” onde /usr/lib/tsql é o diretório default de instalação. Este comando pode ser inserido em algum arquivo da máquina(por exemplo no LINUX o arquivo /etc/rc.d/rc.local) para ser sempre executado na partida.
Após a ativação é recomendado verificar se o processo xsqlserv está ativo. Se não estiver, o arquivo xsqlserv.log que se encontra no diretório de instalação poderá conter alguma informação sobre o ocorrido.
O número default do serviço oferecido pelo programa xsqlserv é 6010. Se este número estiver sendo utilizado por algum outro serviço, a ativação do xsqlserv não terá sucesso. Neste caso é necessário atribuir um outro número ao serviço de nome tsqlserv. No ambiente UNIX isto é realizado inserindo-se a seguinte linha no arquivo /etc/services: **tsqlserv <numserv>/tcp <comentário>** onde numserv é um número não utilizado por outro serviço. No ambiente WINDOWS isto pode ser feito de diversas formas. A solução utilizada para o UNIX também se aplica mas nem sempre funciona para toda e qualquer configuração do NT. Neste caso deve-se verificar a documentação relativa aos serviços TCP/IP. Uma outra forma de especificar o número do serviço quando necessário é através do arquivo de configuração do TSQL cuja descrição se encontra no **apêndice 5**.
O número do serviço deve ser o mesmo para todas as máquinas que utilizam e/ou oferecem o serviço na rede. Desta forma se o número do serviço do servidor TSQL não puder ser 6010, é necessário definir também o novo número em todas as máquinas clientes.
No caso em que a aplicação e a base OpenBase estejam na mesma máquina não é necessário ativar o programa xsqlserv.

12 Apêndice 3: Referência a uma base

Em geral a referência á uma base é efetuada através da especificação de uma cadeia que define a base propriamente dita e alguns parâmetros. A cadeia possui a seguinte sintaxe:

<parâmetro> = <valor>; <parâmetro>=<valor>; ……

<parâmetro> é um dos seguintes identificadores de três letras:

DSN	arquivo do dicionário OpenBase	obrigatório
SEC	código de segurança	default : 1
LEV	nível de acesso	opcional
HST	máquina onde se encontra a base	opcional
CWD	diretório absoluto em relação ao qual são definidos os arquivos da base	default : diretório do dicionário
COL	COL=atomic ou COL=all	default : atomic
ISL	nível de isolamento	default : 1

Exemplo:

```
DSN=c:/temp/dic;SEC=10;LEV=senha;
```

É imperativo especificar o parâmetro CWD quando um item nome de um esquema OpenBase for especificado na forma de um caminho relativo. Caso contrário o TSQL poderá não visualizar certas tabelas da base. Desta forma quando CWD não é especificado, os itens nome de tabela devem ser especificados na forma do caminho completo de um arquivo a partir da raiz ou na forma do nome do arquivo quando o arquivo reside no diretório da base.

O parâmetro COL especifica a visibilidade dos campos do tipo POS para a SQL. No modo atomic somente os subcampos do último nível são visíveis sendo que todos os subcampos de um campo devem ser nomeados. No modo all todos os campos são visíveis.
Exemplo:

```
data      U8
    ano   U4 POS data
    mes   U2 POS data + 4
    dia   U2 POS data + 6
```

No exemplo, para COL=atomic, somente os campos ano, mes e dia são visíveis para a SQL. Se o campo ano não tivesse sido especificado, somente o campo data seria visível tendo em vista que todos os subcampos de um campo devem ser nomeados para serem visíveis.
Se COL=all, todos os campos data, ano, mes e dia são visíveis. Esta modalidade não é recomendada para a execução de comandos SQL do tipo INSERT considerando que os valores de data e mes por exemplo são superpostos .Em geral ela só deve ser utilizada para comandos de leitura (SELECT). O modo atomic é fortemente recomendado no caso do acesso ODBC. O modo all pode ser utilizado para recuperar dados em casos muito específicos.

O parâmetro ISL permite definir o nível de isolamento que tem efeito somente quando o bloqueio não é de banco (Veja capítulo 1).

O parâmetro HST especifica a máquina na qual reside a base. Se ele não for especificado a base é considerada residente na máquina local e a conexão entre a aplicação e o sistema TSQL não será do tipo cliente/servidor. No tsqlwi e no ODBC não é necessário especificar o parâmetro HST tendo em vista que eles possuem formas específicas para definir a máquina.

No caso mais específico do programa tsqlwi apresentado no **apêndice 6**, a cadeia aqui descrita não deve ser especificada para a conexão com o servidor considerando que os parâmetros da cadeia são fornecidos através os campos de uma interface teoricamente auto explicativa.

13 Apêndice 4: ODBC

Para efetuar uma ou mais conexões ODBC entre uma base OpenBase e uma aplicação é necessário que o servidor TSQL esteja instalado na máquina que contém a base e que o driver ODBC do TSQL(OpenBase Tsql Driver) esteja instalado na máquina da aplicação(Veja o apêndice 1).

Por outro lado, a base deve ser registrada na máquina cliente através a execução do programa administrador de bases ODBC que se encontra no painel de controle do Windows . Neste programa, após a escolha da opção "adicionar" e de selecionar o driver "OpenBase Tsql Driver", uma caixa de diálogo é exibida para o preenchimento dos seguintes campos :

- **Nome da base de dados**(Data Source Name) : É o nome com o qual a base é cadastrada para o ODBC.
- **Caminho da base de dados**(Database Path) :
  Como descrito no apêndice 3, é a cadeia que define a base propriamente dita assim como certos parâmetros de abertura. Exemplo :
  DSN=c:/temp/dic;SEC=10;LEV=senha;
  Por motivos de segurança os parâmetros SEC e LEV podem ser omitidos na cadeia. Neste caso, se seus valores não forem do tipo "default" eles deverão ser fornecidos durante a execução da aplicação através a caixa de diálogo que requer a identificação do usuário(nível de acesso) e a passwd(código de segurança) . Outros parâmetros podem ser especificados e estão documentados no apêndice 3.
- **Nome da máquina servidora** (Host Name):
  É o nome da máquina onde reside a base (Veja arquivo hosts). Se a máquina não for especificada a conexão será local.

No caso de haver algum problema na conexão deve-se verificar se as condições de funcionamento do servidor TSQL descritas no apêndice 2 estão respeitadas.

14 Apêndice 5: Configuração

Determinadas características de funcionamento do ambiente TSQL podem ser configuradas através a especificação de parâmetros. O TSQL procura a definição dos parâmetros segundo a ordem de prioridades definidas a seguir:

- Ambiente UNIX:
  1. No arquivo cujo nome está definido na variável de ambiente TSQLCFG
  2. No arquivo /etc/xsqlserv.cfg
  3. No arquivo xsqlserv.cfg do diretório default de instalação(/usr/lib/tsql).
- Ambiente Windows:
  1. No arquivo cujo nome está definido na variável de ambiente TSQLCFG
  2. Nos valores da chave do registry: HKEY_LOCAL_MACHINE\Software\Tecnocoop Sistemas\TSQL server\V8.1
  3. No arquivo xsqlserv.cfg do diretório default de instalação(c:\tsql).

Em um arquivo cada definição de parâmetro deve ocupar uma linha.

Parâmetros que podem ser definidos :

TSQLPATH=<diretório de instalação>
Não implementado.

TSQLPORT=<número do serviço TSQL>
Se for atribuído algum nome á TSQLPORT , este número se sobreporá ao número default(6010) e á qualquer número que tiver sido atribuído ao serviço tsqlserv no arquivo /etc/services. O mesmo número deve ser utilizado no servidor e nos clientes.

TSQLMEMSM=<tamanho da área de sort em kbytes>
O tamanho da área de sort pode ser aumentado no caso em que uma consulta com a cláusula GROUP BY ou ORDER BY não tiver um desempenho satisfatório.

TSQLTIMEOUT=<número de segundos>

Define o intervalo máximo em segundos que uma conexão pode permanecer aberta sem que haja qualquer troca de informações entre a aplicação e o servidor. Após o intervalo a comunicação é fechada pelo servidor. O intervalo default é de 3600 segundos.

XSQLTRACE= S | N
IMPNODOS = S

Quando é atribuído o caracter S ao parâmetro XSQLTRACE e o caracter C ao parâmetro IMPNODOS, o TSQL grava no arquivo xsqlserv.trace do diretório de instalação as informações sobre a estratégia adotada por ele para a execução dos comandos SQL. Estas informações são utilizadas pelo suporte para avaliação da qualidade da otimização dos comandos SQL.

15 Apêndice 6: tsqlwi e tsqlI

O tsqlwi é um ambiente gráfico, interativo e amigável para a execução direta de comandos SQL no Windows. Cada resultado é exibido em uma janela correspondente. Êle mantém um cadastro de bases que permite identificar uma base com simplicidade. Como o ambiente é auto explicativo, a sua descrição neste manual seria redundante.

Por sua vez, o tsqlI é um programa que oferece ao usuário uma interface interativa e amigável não gráfica no UNIX para a execução de comandos SQL.O utilitário possui:

- Um ambiente para desenvolvimento de comandos SQL
- Um ambiente para desenvolvimento de formulários.
- Um gerador de relatórios com vários níveis de quebras, com totalizações, médias e freqüências.
- Um gerador de telas para consultas e inserções de dados.
- Um gerador de telas de menu, cada opção deste pode ser associada a execução de um programa qualquer, um comando SQL, um submenu ou uma tela para consulta e atualização de dados.

15.1 1. Para entrar no sistema:

Para acessar o TSQL de forma interativa digite:
tsqlI "DSN=.....;.....;....." [opções]
Veja o apêndice 3 (Referência a uma base) para a especificação da cadeia inicial. A cadeia deve ser digitada entre aspas.
opções indicam o modo de acesso a interface;
Se a base não for remota a seguinte forma também pode ser utilizada para ativar o tsqlI.
tsqlI nome [–s código de segurança] [–n nível de acesso] [opções]
onde nome especifica o nome do arquivo do dicionário. Pode ser o caminho completo ou o caminho em relação ao diretório corrente.
opções indicam o modo de acesso a interface;
As opções podem ser:
-E arquivo
Com esta opção, o tsqlI entra direto em modo de tEla com a tela contida em "arquivo". Como podem ser definidas telas de menu e de manipulação de dados, esta opção permite o uso da interface simulando uma aplicação. O usuário que entra diretamente neste modo não tem acesso aos outros recursos do sistema.
-X arquivo
Com esta opção, o tsqlI entra direto no modo comando executando o comando contido em "arquivo". Novamente, o usuário que entra no sistema diretamente neste modo, não tem acesso aos demais recursos do sistema.
-N nível
Com esta opção o usuário especifica os níveis de isolamento 0,1 ou 2 (Veja manual de referência), sendo que o nível "default" é 1. O nível com o qual foi ativado o tsqlI é indicado no menu OPÇÕES.
No nível 0 nenhum bloqueio de registros é efetuado e desta forma o tsqlI não permite operações de escrita, só permitindo portanto a execução de comandos SQL do tipo SELECT.

No caso de uma base *OpenBASE* com bloqueio de banco, se um usuário ativar o tsqlI com nível diferente de zero, o banco ficará bloqueado por esse usuário durante todo tempo da exibição do relatório correspondente a execução de uma consulta.

15.1.1 1.1 Menu Principal

O tsqlI possui uma interface amigável ao usuário, composta de menus e submenus que guiam o usuário pelos diversos recursos do ambiente. Em cada uma das telas, há um menu de opções na linha superior do vídeo. Para retornar ao ambiente anterior, usa-se sempre a tecla ESC ou opção específica no menu.
O menu principal, no qual se chega ao chamar o tsqlI sem opções oferece três alternativas: modo de Comando, modo de tEla, e modo de Opções. Cada um desses modos será explicado em itens próprios a seguir.

15.2 2. Modo de Comando

O modo de Comando permite a edição e execução de procedimentos SQL. Oferece, também, a possibilidade de salvar e restaurar a área em edição.
O editor possui uma área de edição de 80 colunas e permite rolamento vertical. Os comandos de edição seguem o padrão do N'office e podem ser encontrados no item 5 deste capítulo.
As teclas PgUp e PgDn, no editor, recuperam os vinte últimos comandos executados com sucesso.
Durante a execução de um comando SQL, aparece um asterisco ("EM EXECUÇÃO") piscando na linha superior da tela. Nesta linha são colocadas, também, as mensagens de erro da SQL, quando for o caso.

15.2.1 2.1 Formulários

Ao ser executada um consulta SQL, a resposta aparece em um relatório. Se nada for especificado em contrário, é usado um formulário padrão, onde aparecem as colunas dispostas lado a lado entituladas pelo nome da coluna, se houver, ou pelo conteúdo da cláusula AS do comando SELECT. A opção Formulários permite que seja criado, ativado e desativado outros tipos de relatório.
Ao entrar no modo formulário, para editar um formulário, é necessário indicar seu nome (pode já existir ou não). A seguir são especificadas as características gerais do formulário:

- número de linhas por página.
- número de colunas por linha.
- margens esquerda e direita.
- número de linhas do cabeçalho.
- úmero de linhas do cabeçalho de cada coluna.

A tecla PgDn permite desviar para o próximo menu sem necessidade de deslocar o cursor até o último item das características gerais.
Estando as características definidas, aparece um novo menu, através do qual pode-se definir, sempre de forma interativa:

- o cabeçalho do relatório (Relatório);
- o título e conteúdo de cada coluna do relatório (Detalhe). Na operação de inserção de uma coluna, a coluna é inserida antes da corrente;
- a definição de quebras do relatório (Quebra).Para se remover uma quebra deve-se digitar no campo de quebra um "*".A informação da existência de uma quebra é exibida no Detalhe da coluna correspondente.
- a definição de totais finais (tOtais).

Após a edição do formulário este deve ser ativado (Ativa). A partir daí as seleções feitas geram relatórios no formato especificado pelo relatório. Se existe um formulário ativado, a função Ativa mostra o seu nome. Para desativar o formulário escolha a opção Desativa.
A opção Gera permite gerar em um arquivo o formulário padrão correspondente a um determinado comando SQL de seleção. Para isto deve-se ligar a opção GERA antes da execução do comando. A geração de um formulário padrão pode ser o ponto de partida para a criação de um formulário não padrão.

15.3 3. Modo de tela

O tsqlI oferece um gerador de telas para consulta e inserção de dados e de telas de menu.
Uma tela deve ser previamente editada em um editor de textos externo e chamada a partir da opção "tEla" do menu principal.

Existem três tipos distintos de telas. Em todas elas, as linhas 1 a 20 do arquivo devem conter o desenho da tela que pode ser livremente definida pelo usuário.
As cadeias “XX/XX/XX” e “XX:XX:XX” especificam a posição da tela onde será colocada, respectivamente, a data e a hora corrente.
O primeiro caracter underline (“_”) de uma seqüência de caracteres underline indica a posição do primeiro caracter da informação a ser colocada na tela. O conteúdo desta informação varia conforme o tipo de tela em que se está trabalhando.
O primeiro caracter percente (“%”) de uma seqüência de caracteres percente indica a primeira posição de uma mensagem a ser colocada na tela. Esta opção só existe nas telas de definição de menu.

15.3.1 3.1 Tela tipo menu - 1ª forma

Esta forma de tela permite a criação de menus. São definidas opções que podem chamar outras telas, comandos SQL e programas executáveis do sistema.
As linhas de 1 a 20 definem o desenho da tela. Na linha 21 deve ser colocado um caracter arroba (“@”) seguido de uma lista de opções do menu. Cada item da lista define uma opção do menu a ser posicionada nos grupos de underline seguindo uma precedência da esquerda para a direita, de cima para baixo.
A sintaxe de um item de opção é:

nome_arquivo ([código,] “opção”, “mensagem”)

nome_arquivo
Especifica o nome do arquivo a ser executado a partir desta opção. Como já foi dito, uma opção pode executar um arquivo executável, um comando SQL (previamente editado num editor de textos ou no próprio modo de edição de comandos do tsqlI), ou uma outra tela (submenu ou tela de dados).
código
Indica o conteúdo do arquivo. O código “X” indica programa executável, o código “P” indica um procedimento SQL e a ausência de código indica tela.
opção
Define o conteúdo da opção que aparecerá no menu da tela. Uma opção poderá ser selecionada com o uso das setas ou teclando a primeira letra maiúscula da opção. Esta aparecerá destacada no vídeo.
mensagem
Indica a mensagem que será mostrada a partir do 1º caracter percente (“%”), quando o cursor indicar esta opção. Só é permitida a definição de uma seqüência de caracteres “%” na tela.

15.3.1.1 Exemplo:

XX/XX/XXXX:XX:XX
Menu Principal

._____
._____
._____
._____
._____

%%%%%%%%%%%%%%%%%%%%

tela02.t (“Clientes”, “Cadastro de Clientes”)
com01.s (P, “produtos”, “Consulta Produtos”)
tela04.t (“Pedidos”, “Cadastro de Pedidos”)
nx(X, “Editor”, “N’Office”)
sh(X,”Shell”, “Sistema”)

15.3.2 3.2 Tela tipo menu - 2ª forma

O tsqlI oferece, ainda, uma outra forma de definição de menu. Neste modo o desenho da tela (nas linhas de 1 a 20) deve incluir o texto referente as opções do menu. Uma opção neste tipo de menu só pode ser selecionada por cursor que aparecerá nas posições indicadas pelo caracter underline (“_”).
Na linha 21 do arquivo de tela deve aparecer o caracter “#” seguido de uma lista de opções. Cada uma delas contém o nome do arquivo, código e mensagem semelhante ao primeiro modo de tela:
nome_arquivo [([código,] [“mensagem”])]
Estas opções são análogas as do item 3.1. Nome_arquivo especifica o arquivo a ser executado que pode ser um programa executável (código X), um procedimento SQL (código P) ou uma tela (código nulo).

15.3.3 Exemplo:

XX/XX/XXXX:XX:XX
[_]Consulta tabela

[_]Insere e Seleciona Pedidos
%%%
 # com04.s (P, "Mostra todos os pedidos")
tela03.t ("Tela de inserção e seleção de pedidos")

15.3.4 3.3 Tela de dados

Uma tela de dados pode ser definida para inserir e consultar tabelas da base. O tsqlI traduz internamente as ações do usuário em comandos SQL para o TSQL.
Neste caso o desenho da tela nas linhas 1 a 20 devem ser em número proporcional ao número de opções de forma a tela possuir um ou mais registros da tabela ao mesmo tempo.
A partir da linha 21 do arquivo de definição da tela deve ser colocado o nome da tabela seguido de uma lista de itens.
Cada item define o nome de uma coluna da tabela e opcionalmente uma máscara, um número de ordem para a seleção:

nome_coluna (["máscara",][ordenação])

Ver máscara no capítulo "máscaras de edição".
A sintaxe de ordenação é On onde n é o número de ordem pela qual a seleção será ordenada.
Quando este tipo de tela é executada ficam disponíveis (se o usuário corrente tiver estes privilégios) as opções de inserção e seleção.
A seleção pode ser feita por um ou mais campos digitando as condições necessárias no campo referente a coluna. Esta condição deve seguir a mesma sintaxe das condições de seleção de uma cláusula WHERE da consulta SQL. O nome da coluna não é digitado.
Para que a seleção seja feita por mais de um campo as teclas PgDn e PgUp vão para o campo seguinte onde pode ser a colocada nova condição. Para concatenar condições um "AND", um ponto e vírgula ";" ou se nada é colocado indicam um conector AND SQL. Um "OR" ou uma vírgula "," indicam um conector OR SQL.

15.3.5 Exemplo:

XX/XX/XXXX:XX:XX
Cadastro de Clientes
Código Clientes Crédito

_____ _____ _____
_____ _____ _____
_____ _____ _____
_____ _____ _____
_____ _____ _____
_____ _____ _____
_____ _____ _____

clientes cod cliente ("9999")
nome ("@s20,01)
credito ("$ZZ.ZZZ.ZZ9,99")

15.3.6 Exemplo:

XX/XX/XXXX:XX:XX
CADASTRO DE PEDIDOS
Código: _ _ _ _ _ _ _ _ Data: _ _ _ _ _ _ _ _

Fabricante: _ _ _ _ _ _

Produto:_ _ _ _ _ _ _ _ Quantidade: _ _ _ _ _

Código do Cliente:_ _ _

Preço:_ _ _ _ _ _ _ _

Código do Vendedor:_ _ _

pedidos **pedido# (“999999”)**
data(“@Dyd”)
fabricante (“XXXX”)
produto (“XXXXX”)
quantidade (“z9”)
cliente (“9999”)
preco total (“zzz zzz,99”)
vendedor (“999”)

15.4 4. Opções do Ambiente

Nesta opção, é possível definir algumas opções do ambiente.
As opções são:

- Definição do esquema a ser acessado. Seu valor inicial é definido pela variável de sistema SCHEMA ou, na sua ausência, pelo nome do usuário corrente (login).
- Definição do número de linhas por página do relatório. O valor inicial é 20 (tamanho ideal para o vídeo), mas deve ser modificado para se adequar ao tamanho do papel na impressora ou outro dispositivo de saída.
- Definição do tipo de saída. Pode ser em vídeo, em arquivo, num dispositivo qualquer do sistema (/dev/…) ou num pipe.
- Definição da saída. Conforme a opção anterior pode ser nulo, nome de um arquivo, de um dispositivo, etc.
- Definição de transação. O valor (F) faz com que o tsqlI feche a transação após cada comando executado. O valor (I) faz com que a transação seja fechada quando um próximo comando é submetido para execução. Com a opção N, é deixado a cargo do usuário, através da execução do comando COMMIT da SQL, o fechamento da transação no momento em que lhe convier.
- Confirmação ao atualizar dados via tela formatada. Esta opção indica se deve ou não ser pedida confirmação na entrada de dados via tela de dados.
- Inicializa confirmação. Esta opção seta o valor default, se houver, para os pedidos de confirmação de atualização de dados. O valor inicial pode ser Sim, Não ou nulo (não há valor default).

15.5 5. Comandos de Edição

Podem ser usados os mesmos comandos do N’office (a lista não deve estar completa):

Setas:	locomovem cursor
^E:	sobe cursor
^X :	desce cursor
^D :	cursor para direita
Bsp :	cursor para esquerda
^H :	volta cursor apagando o caracter anterior
^G :	apaga caracter corrente
^Y :	apaga linha
^T :	apaga palavra
^F :	vai para palavra seguinte
^A :	vai para a palavra anterior
^QF :	procura palavra
^L :	procura próxima ocorrência da palavra
^QLn :	vai para n-ésima linha
^N:	insere nova linha
^V :	entra em modo de inserção
^C e ^R :	Recuperam os últimos 20 comandos executados com sucesso. ^C percorre a lista

	dos comandos para a frente e ^R para traz.

Observação: ^P pode ser utilizado no lugar de ^Q.

15.6 6. Formatos de máscara:

Para cadeia de caracteres:

9	indica a posição de um algarismo (0 a 9).
#	indica a posição de um algarismo ou branco.
A	indica a posição de uma caracter alfabético ou branco.
N	indica a posição de um caracter alfanumérico ou branco.
X	indica a posição de um caracter qualquer.
!	converte para maiúsculas.

@Dij data, i e j especificam as precisões inicial e final.

Podem ser:

A	para ano
m	para mês
D	para dia
H	para hora
N	para minuto
S	para segundo

@Sn define o tamanho da janela de edição (se o campo for maior, faz scroll horizontal.

Para tipo numérico:

Z	indica posição de um algarismo, suprimindo zeros a esquerda.
#	indica posição de um algarismo ou branco.
9	indica a posição de um algarismo.
*	indica a posição de algarismo e substitui zeros a esquerda por *
$	indica posição de algarismo, substituindo zeros a esquerda por brancos, exceto o último zero a esquerda.
-	idem acima, porém substituí último zero a esquerda por "-", caso o número seja negativo.
@f	indica posição da vírgula não fixa.
@in	coloca n vezes o caracter i.

7. A tecla F3 (ou seqüência ^B)

Através da tecla F3, ou a seqüência ^B, o usuário pode em qualquer ponto do tsqlI executar comandos via a shell do sistema, como por exemplo listar os arquivos do diretório corrente. Este recurso é extremamente útil levando-se em conta que o usuário pode possuir formulários, consultas e relatórios armazenados em arquivos comuns.

16 Apêndice 7: Mensagens de erro

Os códigos de erro são retornados no parâmetro SQLCODE após a execução de cada comando SQL. Na lista abaixo, a coluna mais à esquerda fornece o código numérico do erro, a coluna mais à direita a descrição do erro e a coluna do centro a constante simbólica para aplicações em C. As constantes simbólicas estão definidas no arquivo /usr/include/sqlcode.h.

16.1 Erros graves de processamento:

-512 xFTCANCSIG	Processador TSQL cancelado
-513 xFTALLOC	Erro em alocação dinâmica de memória
-514 xFTOVERPSPR	Estouro da pilha sintática em um percurso
-515 xFTGCMEMNS	Memória alocada pela geração de código
-516 xFTOVERPS	Estouro da pilha sintática
-517 xFTNPRECUAR	O analisador sintático não pode recuar
-518 xFTSAIARQ	Erro na geração de arquivo
-519 xFTBADF	Handle de propriedades invalido

16.2 Erros de uso (parâmetros de ativação):

-520 xKKFONTE	Não conseguiu abrir arquivo fonte
-521 xKKNOMEDB	Não se posicionou na base de dados
-522 xKKINVALDSN	Nao é uma base de dados valida
-523 xKKPERM	Parametro de segurança inválido
-524 xKKPASSREM	Usuário não cadastrado ou passwd invalido
-710 xKKINVALARG	Argumento de chamada de funcao invalido
-711 xKKNOSPC	Area em memoria insuficiente para o valor
-712 xKKDEMONS	Data de demontração vencida

16.3 Erros léxicos em comando SQL:

-525 xLXIDTERMU	Caracter '_' termina identificador
-526 xLXIDDOISU	Caracter '_' duplicado no identificador
-527 xLXIDGRANDE	Identificador muito grande
-528 xLXCARINV	Caracter inválido
-529 xLXEOFEMCAD	Fim de arquivo em cadeia de caracteres
-700 xLXCADNBIT	Caracter invalido em cadeia de bits
-701 xLXCADNHEXA	Caracter invalido em cadeia hexa de bits
-702 xLXBINNHEXA	Caracter invalido em cadeia binaria
-703 xLXBINNPAR	Par incompleto em cadeia binaria

16.4 Erros sintáticos em comandos SQL:

-530 xSTERRO	Erro Sintático.

16.5 Erros semânticos em comandos SQL:

-531 xSMCURSORDUP	Cursor já declarado
-532 xSMFARITCHAR	Tipo CHAR em função SUM ou AVG
-533 xSMFFUNCAO	Função como argumento de função
-534 xSMNDIGBINV	Precisão de tipo FLOAT inválida
-535 xSMLINGNAO	Linguagem não disponível
-536 xSMPROCDUP	Procedimento já declarado
-537 xSMCURSORND	Cursor não declarado
-538 xSM2OPENCUR	Já existe procedimento OPEN para o cursor
-539 xSMARITCHAR	Tipo invalido em operacao aritmetica
-540 xSMNTABINV	Especificação de nome de tabela inválido
-541 xSMNPARMDUP	Nome de parâmetro duplicado

-542 xSMSEMCODE	Parâmetro SQLCODE não declarado
-543 xSMTCHARINV	Tamanho inválido para o tipo CHARACTER
-544 xSMTDECINV	Tamanho inválido para o tipo NUMERIC
-545 xSMESCINV	Escala inválida
-546 xSMTPINVLNG	Tipo de dado inválido para a linguagem
-547 xSMCODEDUP	Parâmetro SQLCODE duplicado
-548 xSMTABNCUR	Tabela não identificada na declaração do cursor
-549 xSMCOLQNRES	Coluna qualificada não resolvida
-550 xSMCOLNQNRES	Coluna não qualificada não resolvida
-551 xSMCOLNQAMBG	Coluna não qualificada ambígua
-552 xSMCOLOBAMBG	Coluna na cláusula ORDER BYambígua
-553 xSMCOLOBNSEL	Coluna na cláusula ORDER BY não na SELECT
-554 xSMPARNRES	Parâmetro não declarado
-555 xSMCOLGEMF	Coluna de grupamento argumento de função
-556 xSM2ESCOPF	Colunas de escopo diferente referidas na mesma função
-557 xSMCOLSNG	Coluna nas cláusulas SELECT ou HAVING e não na cláusula GROUP BY
-558 xSMCOL2SET	Coluna repetida na cláusula SET
-559 xSMFWOUSET	Função na cláusula WHERE ou SET
-560 xSMPARSGOB	Parâmetro em cláusula GROUP BY, ORDER BY ou SET
-561 xSMPAR2INTO	Parâmetro duplicado na cláusula INTO
-562 xSMXTABINS	Referência inválida à tabela em INSERT
-563 xSMIDAUTINV	Não é o identificador de autorização do esquema ou módulo
-564 xSMPKEY2	PRIMARY KEY repetida
-565 xSMCOLNDECL	Coluna não definida em CREATE TABLE
-566 xSMVARCINV	Coluna de tipo VARCHAR em restrição
-567 xSMID2LISTA	Identificador repetido em uma lista
-568 xSMNCOL2TAB	Coluna duplicada em CREATE TABLE
-569 xSMNULONPERM	Coluna restrita a valores não nulos
-570 xSMTPATCOLNC	Atribuição a coluna, tipo compatível
-571 xSMOVERFLOW	Transbordo na conversão de constante
-572 xSMXTABUPDD	Referência inválida à tabela em comando DELETE ou UPDATE
-573 xSMCURSOPEN	Cursor sem comando OPEN
-574 xSMTCURNATUA	Tabela do cursor é não atualizável
-575 xSMNORDERINV	Número de ordem na cláusula ORDERBY inválido
-576 xSMNMAXTAB	Excedido o número máximo de tabelas
-577 xSMGS2SNEX	Grau da lista SELECT diferente de 1 em subconsulta que não é EXISTS
-578 xSMPARINDIN	Tipo ou escala de parâmetro INDICATOR
-579 xSMFECSSEMG	Coluna simples e função sem cláusula GROUP BY
-580 xSMFDIST2	Mais de uma função DISTINCT no escopo
-581 xSMAUTHINV	Identificador de autorização inválido
-582 xSMGDIFUNION	Tabelas de graus diferentes em UNION
-583 xSMTPNCHAR	Valor não é do tipo cadeia de caracteres em predicado LIKE
-584 xSMPATNCHAR	Padrão não é de tipo cadeia de caracteres

-585 xSMESCNCHAR	ESCAPE não é do tipo cadeia de caracteres
-586 xSMTPPRNCOMP	Tipos incompatíveis em predicado
-587 xSMFUNCWHERE	Função sobre coluna na cláusula WHERE
-588 xSMFSEMCOL	Funcao(não COUNT(*)) sem coluna
-589 xSMASTSGBYH	SELECT *sem cláusula GROUP BY e com cláusula HAVING
-590 xSMGINTOINV	Grau da cláusula INTO inválido
-591 xSMGINSNCOMP	Graus diferentes no comando INSERT
-592 xSMGCVWNCOMP	Graus diferentes no comando CREATE VIEW
-593 xSMXPARSPEC	Especificação de parâmetro não permitida
-594 xSMGEEINV	Grantee igual a Grantor
-595 xSMDDINCUR	Cursor dinâmico referido em comando posicionado
-596 xSMDNDINCUR	Cursor referido em FETCH dinâmico
-597 xSMDALGINV	Variável SQLDA com linguagem do módulo incompatível
-598 xSMDANSMALL	Variável SQLDA de tipo não SMALLINT
-599 xSMSTATNCHAR	Variável comando de tipo não caracter
-600 xSMLCOLCRV	Lista de colunas obrigatória em comando CREATE VIEW
-601 xSMCOL2INS	Coluna referida mais de uma vez em cláusula INSERT
-602 xSMXREVOKE	Não pode especificar colunas em REVOKE
-603 xSMTPUNNCOMP	Tipos incompatíveis em operação UNION
-604 xSMLOCKVIEW	Comando LOCK TABLE sobre visão
-605 xSMLOCKMODE	Usuário não tem privilégio necessário ao modo de bloqueio sobre a tabela
-606 xSMDTQUAINV	Qualificador ao tipo DATETIME inválido
-607 xSMXCOLSPEC	Especificação de coluna não permitida
-608 xSMXFUNSPEC	Especificação de função não permitida
-609 xSMFREENLOC	Parametro não LOCATOR em comando FREE
-610 xSMNAODATA	Data (DD/MM/AAAA) inválida
-611 xSMIDERANV	Identificador de era inválido
-612 xSMDTVNV	Valor em constante data-hora invalido
-613 xSMPARDININV	Uso invalido de parametro dinamico
-614 xSMPINCHAR	Tipo de argumento de funcao POSITION
-615 xSMULNCHAR	Tipo de argumento de funcao UPPER ou LOWER
-616 xSMTRNCHAR	Tipo de argumento de funcao TRIM
-617 xSMTMTPNV	Tamanho invalido para tipo
-618 xSMQCDTNV	Qualificador inválido para constante DATETIME
-619 xSMFEXTEND	Tipo do argumento da função EXTEND não é DATETIME
-620 xSMTPATPARNC	Tipos incompatíveis em atribuição à variável
-621 xSMCCNCHAR	Tipo do operando de concatenação inválido
-622 xSMLENCHAR	Tipo do argumento da função LENGTH inválido
-623 xSMSBNCHAR	Tipo do argumento da função SUBSTRING não é cadeia de caracteres
-624 xSMSBPINNUM	Posição inicial em função SUBSTRING de tipo não numérico ou escala não zero
-625 xSMSBTRNNUM	Tamanho em função SUBSTRING de tipo não numérico ou escala não zero

-626 xSMNOMECOM	Não existe o comando associado ao cursor
-627 xSMMODNINT	Tipo ou escala de argumento de funcao MOD
-628 xSMCNFUCDIN	Coluna não está na cláusula FORUPDATE
-629 xSMSETNULL	SET NULL em chave restrita a não nulos
-630 xSMFORKT2REF	Erro na dupla referência à tabela em comando CREATE TABLE
-695 xSMPARFOTINV	Tipo de parametro em FETCH ORIENTATION
-696 xSMFONOTNEXT	FETCH diferente NEXT mas cursor sem SCROLL
-697 xSMLOCNOTLOB	AS LOCATOR para parametro não LARGE OBJECT
-698 xSMCCTPNCP	Tipos incompativeis em concatenacao
-699 xSMTPINVPRED	Tipo invalido em predicado
-700 xSMTPINVGRBY	Coluna de grupamento de tipo invalido
-701 xSMTPINVGRBY	Tipo invalido em clausula ORDER BY
-702 xSMTPINVUD	Tipo invalido em UNION DISTINCT
-703 xSMTPINVSD	Tipo invalido em SELECT DISTINCT

16.6 Erros na execução de comandos SQL:

-631 xINCHINVCAST	Cadeia de caracteres invalida em CAST
-632 xINOBJINVAL	Objeto do módulo não compatível
-633 xINMODINVAL	Módulo não está válido para execução
-634 xINCODINT	Código interpretável não compatível
-635 xINTUPLASS	Cardinalidade de comando SELECT INTO ou subconsulta maior que 1
-636 xINCURFEC1	Cursor fechado em DELETE ou UPDATE
-637 xINCURFEC2	Cursor fechado em CLOSE ou FETCH
-638 xINNAOPOS	Cursor não posicionado em comando DELETE ou UPDATE
-639 xINOPEN	Cursor já aberto em comando OPEN
-640 xINVNULLINV	Atribuição de valor nulo inválida
-641 xINUNDO	Erro no cancelamento das alterações
-642 xINOVERFLOW	Transbordo em operação ou atribuição
-643 xINDIVZERO	Divisão por zero
-644 xINCHAVDUP	Chave duplicada
-645 xINESCFIMPAD	ESCAPE último caracter do padrão
-646 xINCPRENSEL	Comando preparado não é SELECT
-647 xINCPRESEL	Comando preparado é SELECT
-648 xINDAFETCHNV	SQLDA de FETCH dinâmico inválido
-649 xINNTEMCPRE	Comando não existe ou não preparado
-650 xINCOLNDECL	Coluna de tabela não identificada em CREATE INDEX ou GRANT
-651 xINVARCINV	VARCHAR em CREATE INDEX
-652 xINNMAXMOD	Ultrapassado o limite de módulos em execução
-653 xINNPODEEXM	Usuário não pode executar o módulo
-654 xINMODEMEX	Não pode invalidar módulo em execução
-655 xINLEITINT	Erro na leitura do código do módulo
-656 xINMAXOPEN	Limite máximo de arquivos abertos

-657 xINERRSORT	Erro não esperado no módulo SORT-MERGE
-658 xINMAXSORT	Limite máximo de ordenações ativas
-659 xINTPDANCOMP	Tipos incompatíveis em FETCH dinâmico
-660 xINCVINPARAM	Erro na conversão de entrada de parâmetro
-661 xINMEMNOK	Endereço de parâmetro inválido
-662 xINOBJNOK	Objeto do módulo foi corrompido
-663 xINSBPINVAL	Posição inicial de SUBSTRING inválida
-664 xINSBTRNVAL	Tamanho do resultado da função SUBSTRING inválido
-665 xINCOMOPEN	Comando preparado está associado a cursor aberto
-666 xINFORKDEF	Chave estrangeira incompatível com a chave primária da tabela referida
-667 xINDEADLOCK	Esgotado o tempo de espera por objeto reservado a outro usuário
-668 xINFORKREF	Violação de restrição referencial
-669 xINNOLOGIN	Banco OpenBASE não está disponível para (ou por) uso exclusivo
-670 xINPKG	Pacote de software não instalado
-671 xINTIMEOUT	Tempo de espera esgotado
-672 xINPERM	Violacao de permissao de acesso
-673 xINEOF	Fim de arquivo
-688 xINCOMTSQL	Erro de comunicação com servidor TSQL
-691 xINTAMCHAVE	Limite do tamanho de chave de índice
-692 xINWCONOK	Violação de cláusula WITH CHECK OPTION
-693 xINGRANT	Usuario não pode atribuir privilégio
-694 xINREVOKE	Usuario não pode retirar privilégio
-708 xINNOUSEFUL	Base de dados não esta consistente
-709 xINOVTABLOCK	Estourou a tabela de bloqueios

16.7 Erros nas consultas ao Dicionário de Dados:

-674 xDDNAOTEMTAB	Não existe a tabela ou visão
-675 xDDJATEMTAB	Nome de tabela já existe
-676 xDDVISDEPTAB	Visão depende de tabela em DROP TABLE
-677 xDDJATEMMOD	Já existe o módulo
-678 xDDJATEMIND	Nome de índice já existe
-679 xDDNAOTEMMOD	Módulo não existe
-680 xDDNAOPRCMOD	Usuário não tem permissão para recompilar módulo
-681 xDDMODNAOPRC	O módulo não pode ser recompilado
-682 xDDNAOTEMIND	Índice não existe
-683 xDDJATEMVIS	Nome de visão já existe
-684 xDDNAOTEMPRV	Usuário não tem o privilégio sobre a tabela referida
-685 xDDNEXPRVUPC	Usuário não tem o privilégio UPDATE sobre a coluna referida
-686 xDDNEXPRVREF	Usuário não tem o privilégio REFERENCES sobre a tabela referida
-687 xDDTABDEPTAB	Tabela depende de tabela em DROP TABLE