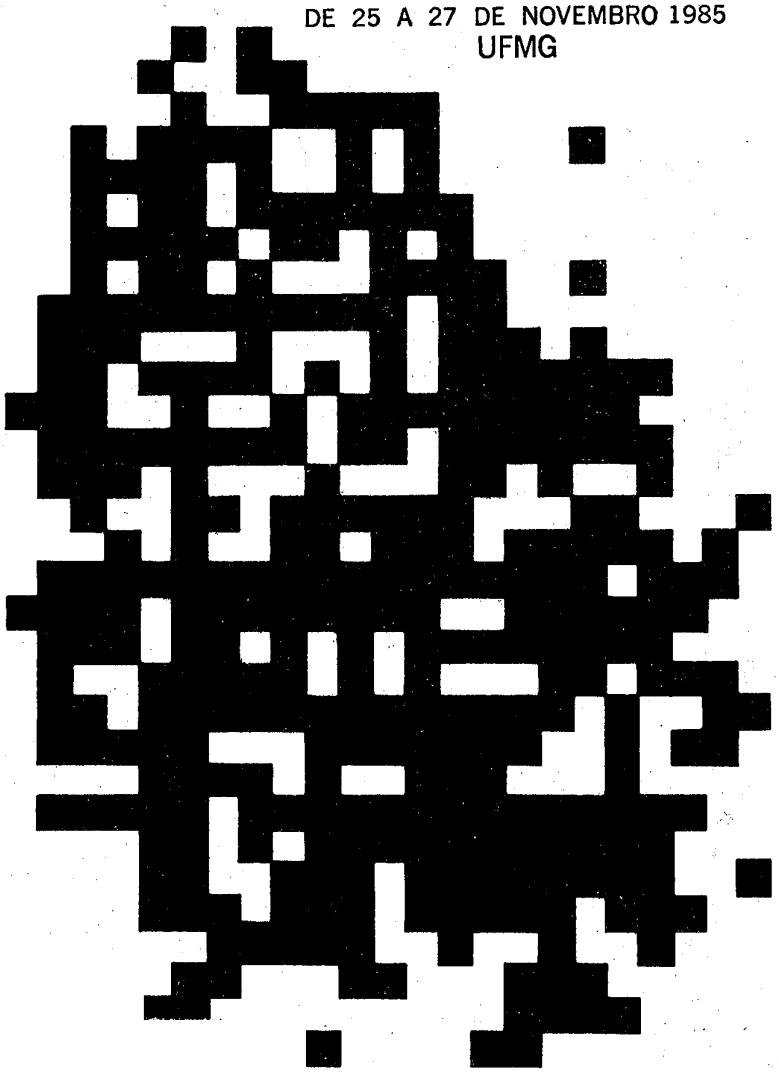
V SIMPÓSIO

SOBRE DESENVOLVIMENTO DE SOFTWARE BÁSICO

DE 25 A 27 DE NOVEMBRO 1985
UFMG



ANAIS

5º SIMPÓSIO SOBRE DESENVOLVIMENTO DE SOFTWARE BÁSICO

BELO HORIZONTE 25 A 27 DE NOVEMBRO DE 1985

A N A I S

PROMOÇÃO: SOCIEDADE BRASILEIRA DE COMPUTAÇÃO - SBC
, COMISSÃO ESPECIAL PARA LINGUAGENS E SISTEMAS DE PROGRAMAÇÃO
UNIVERSIDADE FEDERAL DE MINAS GERAIS - UFMG

PATROCÍNIO: CONSELHO NACIONAL DE DESENVOLVIMENTO CIENTÍFICO E TECNOLÓGICO - CNPq

A Implementação de uma Linguagem de Consulta Compatível com SQL

Autores: Rosana de Saldanha da Gama Lanzelotte (1)
Manoel José Prazeres (2)
Sonia Regina Raddi de Araujo (3)

Palavras-Chave: Linguagem de consulta, bancos de dados, SQL

RESUMO

O presente trabalho descreve a implementação de uma linguagem de consulta compatível com o SQL (Structured Query Language), projetada originalmente pela IBM, e utilizada, em geral, para consultas a bancos de dados.

ENDEREÇOS DOS AUTORES

(1) Rosana de Saldanha da Gama Lanzelotte

PUC-RJ - Departamento de Informática
R. Marques de S. Vicente, 225
22453 - Rio de Janeiro

(2) Manoel José Prazeres

COPPE - UFRJ

(3) Sonia Regina Raddi de Araujo

COBRA - Computadores e Sistemas Brasileiros S.A.
Divisão de Desenvolvimento de Software
Av. GB 8 - Eixo do Centro Metropolitano, nº 447
22700 - Jacarepaguá - Rio de Janeiro

A Implementação de uma Linguagem de Consulta Compatível com SQL

1. Introdução

O presente trabalho descreve o projeto de implementação de uma linguagem de consulta compatível com a linguagem SQL-Structured Query-Language (1). Essa linguagem surgiu durante o projeto do Sistema R, desenvolvido pela IBM durante os anos 74 a 81, e que originou os produtos SQL/DS e DB2, comercializados por essa empresa.

Existem hoje diversas implementações de SQL, desenvolvidas por fabricantes e casas de software americanas, o que faz com que a linguagem esteja em vias de se tornar um padrão "de facto", como linguagem de acesso a bancos de dados, Já existe um comitê ANSI estudando, inclusive, a sua padronização.

A linguagem SQL oferece uma visão relacional dos dados, e permite a criação, manipulação e consulta a tabelas, ou relações. Pode-se dizer que é baseada no cálculo relacional (2), o que a torna fácil de ser aprendida e utilizada. Em cálculo relacional, uma consulta é expressa em termos de que dados se deseja obter, e não como obtê-los. O usuário não precisa "navegar" para chegar até o conjunto de dados desejado, como usualmente acontece em um programa escrito em linguagem convencional. Por esse motivo, é de grande importância a qualidade do processador da linguagem de consulta, pois este deve assegurar um tempo de resposta adequado e compatível com o dispendido por um programa convencional para obter os mesmos dados.

Apesar de alguns pontos de contato com as metodologias de construção de compiladores, um processador de linguagem de consulta requer técnicas específicas de otimização, de modo a atender aos requisitos de desempenho, sem exigir do usuário informações adicionais sobre qual a melhor maneira de obter os dados pedidos na consulta.

2. Objetivos do projeto

2.1. A linguagem SQL

A linguagem SQL constitui, em primeiro lugar, uma poderosa ferramenta de recuperação de dados. O seu comando de consulta é bastante simples de ser usado, mesmo por não-especialistas.

O seu formato é

```
SELECT    nomes de colunas
FROM      nomes de tabelas
WHERE     condições
```

e o resultado da sua execução é um conjunto de tuplas que satisfaz às condições.

O usuário enxerga os dados na forma tabular, como propõe o modelo relacional, em que cada tabela é constituída por um conjunto de linhas (tuplas) e de colunas, como a seguir:

NroEmp	NomeEmp	NroDept	Função	Gerente	Salario	Comissoes
12	Jose	14	Programador	30	100000,00	1000,00
6	João	12	Digitador	32	50000,00	1000,00
17	Celso	10	Perfurador	31	40000,00	1000,00
14	Luiz	15	Operador	30	50000,00	5000,00
7	Clarice	12	Digitador	32	70000,00	500,00

Apesar de simples, o comando SELECT é bastante poderoso, e, além de possibilitar todas as operações da álgebra relacional, engloba:

. operadores aritméticos + , - , / , *
. operadores relacionais e lógicos
. operador booleano IN (lista de valores)
. operador booleano BETWEEN valor1 AND valor2
. operador booleano LIKE "pattern", para pesquisas de cadeias.
. funções intrínsecas: AVG, SUM, MAX, MIN, COUNT
. ordenação opcional do resultado, por quaisquer colunas, ascendente ou descendente
. quebras por grupos, através das cláusulas GROUP BY e HAVING
. aninhamento de consultas
. operadores EXISTS, ALL, ANY

Com esses recursos, é possível implementar qualquer consulta aos dados de uma ou mais tabelas.

Além do comando de consulta, a SQL contém também:
. comandos de definição de dados, que permitem a criação e eliminação de tabelas, índices, visões e sinônimos, e a extensão de tabelas;
. comandos de manipulação, que permitem a inserção, eliminação e atualização de conjuntos de tuplas;
. comandos de controle, para possibilitar o controle de acesso aos dados pelos usuários;

A SQL prevê ainda recursos para a programação de transações.

2.2. Classes de uso.

A linguagem SQL atende às necessidades de três classes distintas de usuários:
(i) usuários finais, não especialistas, que fazem consultas "ad hoc";
(ii) programadores, que desenvolvem aplicações que manipulam os dados;
(iii) administradores de dados, que são responsáveis pelas definições das tabelas, controle de acesso, etc.

Os usuários finais (i) fazem consultas SQL e recebem respostas através de um terminal, interativamente. Necessitam, então, de uma interface interativa com SQL, à qual fornecerão um comando por vez. Receberão as respostas no próprio terminal.

Os programadores (ii) podem necessitar de recursos procedurais, tais como "loops" e desvios, que não estão disponíveis em SQL. As aplicações que requerem esse tipo de recurso são, em geral, do tipo "dirigidas por dados", ou seja, aquelas em que as decisões de processamento são tomadas em função dos dados de entrada. Para que tais aplicações possam ter acesso a tabelas SQL, é necessário prover a hospedagem de SQL em linguagens procedurais, tais como COBOL e C. O SQL hospedado deve dispor de todos os recursos já enumerados, além da possibilidade de tratar conjuntos de tuplas, tupla a tupla e de trocar dados com o programa através de variáveis.

2.3. Armazenamento das Tabelas.

Usualmente as implementações de SQL dispõem de um gerenciador de dados próprio, que se encarrega do armazenamento e acesso às tabelas, que são exclusivamente manipuladas via SQL.

O projeto aqui descrito propõe-se a implementar SQL como uma ferramenta de consulta a dados organizados sob quaisquer formas. Em particular, os arquivos convencionais criados através de programas COBOL e outras linguagens poderão também ser enxergados como tabelas, através do SQL, desde que se associem a eles as descrições de tabelas adequadas. Essa abordagem amplia o escopo de utilização da linguagem, que deixa de ser apenas uma ferramenta restrita ao gerenciador de bancos de dados. Através dessa facilidade, um centro de processamento de dados poderá começar a usar SQL gradualmente, sem a necessidade de migrar de uma só vez toda a sua base de dados para um sistema de gerência de banco de dados.

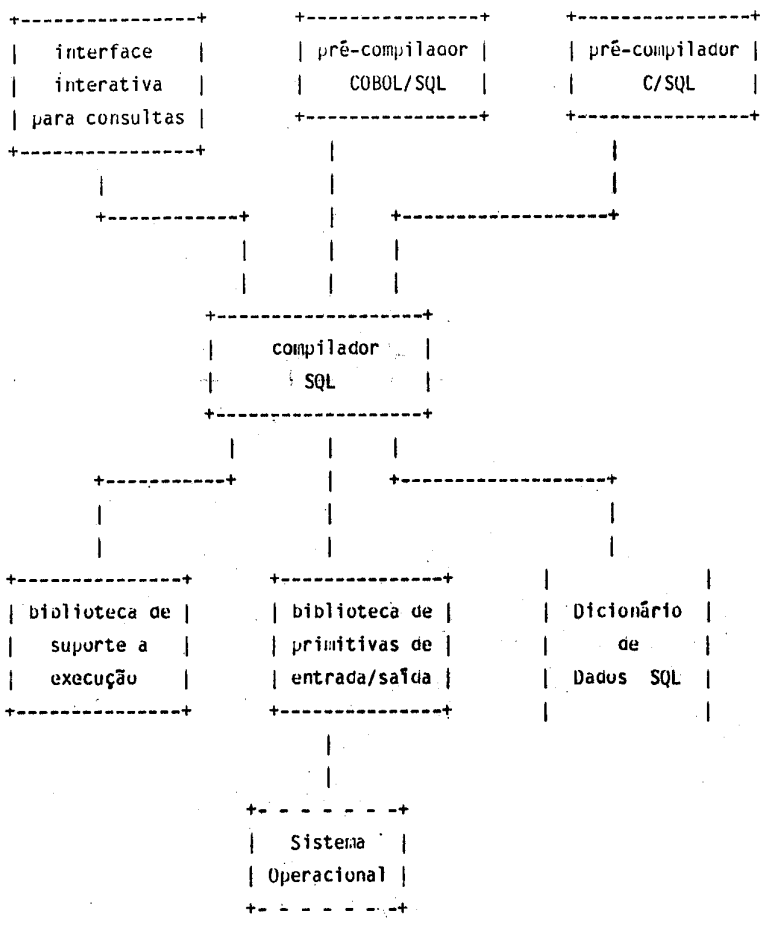
Para dotar o SQL dessa facilidade, é necessário agregar um comando para associar uma descrição de tabela a um arquivo do sistema operacional, cuja sintaxe é semelhante à do comando de criação de tabelas:

```
ASSOCIATE  TABLE  nome
    (nomecoluna 1    tipo ,
     nomecoluna 2    tipo ,
        .              .
        .              .          )
 TO   "identificação do arquivo no S.O."
```

3. Descrição do Projeto.

O projeto de implementação da linguagem SQL é composto dos módulos apresentados a seguir:

. usuário final
. administrador de dados . programadores



3.1. Interface Interativa para consultas.

Esse módulo interage com o usuário e possibilita a realização de consultas "ad hoc", além da geração de relatórios formatados e, eventualmente, entrada de dados em tabelas.

Essa interface atende também ao administrador de dados, que pode usar comandos de definição de dados e controle de acesso interativamente, além de consultar o dicionário de dados e gerar relatórios.

3.2. Pré-compiladores SQL.

Os pré-compiladores SQL aceitam como entrada um programa escrito em linguagem hospedeira, que contém comandos SQL, e geram duas saídas:

- um módulo objeto, contendo o código gerado para cada comando SQL hospedado;
- novo programa fonte, em que os comandos SQL foram substituídos por comandos da linguagem hospedeira que ativam o módulo objeto gerado.

Os pré-compiladores ativam o compilador SQL a cada comando hospedado, e este gera o código correspondente, que é armazenado no módulo objeto. Este se subdivide em diversas seções, cada uma correspondente a um comando SQL hospedado.

Os pré-compiladores são compostos de 2 módulos:

- "front-end", que é dependente da linguagem hospedeira;
- "back-end", que é dependente do SQL.

O "back-end" é comum a todos os pré-compiladores.

3.3. O Compilador SQL.

Optou-se pela compilação da SQL, ao invés da interpretação. A experiência da implementação da SQL pela IBM mostrou que a compilação é vantajosa, mesmo no caso das consultas "ad hoc" (3).

A compilação é usada para implementar tanto consultas "ad hoc" como programas. Essa abordagem unificada simplifica bastante o projeto, pois as mesmas técnicas podem ser usadas em ambos os casos.

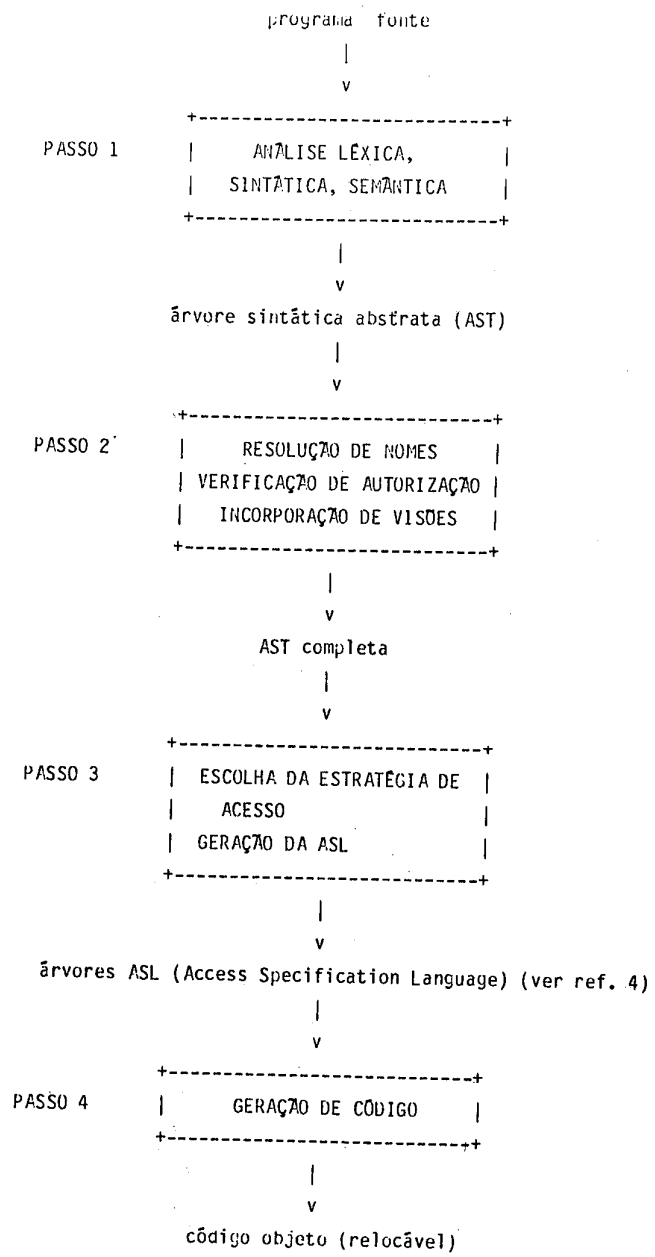
Durante a compilação são feitas as seguintes atividades:

. análise léxica e sintática
. análise semântica estática
. resolução de referências externas e autorização
. escolha do caminho de acesso
. geração de código.

Num sistema interpretativo todas essas atividades seriam feitas no momento da execução, o que acarretaria uma sobrecarga desnecessária.

O compilador SQL pode ser ativado a partir da Interface Interativa ou a partir dos pré-compiladores. Em ambos os casos, recebe um comando SQL por vez.

O compilador SQL é um compilador de quatro passos, em que são usadas duas representações intermediárias. O diagrama abaixo mostra a arquitetura do compilador:


programa fonte
PASSO 1
ANÁLISE LÉXICA,
SINTÁTICA, SEMÂNTICA
árvore sintática abstrata (AST)
PASSO 2
RESOLUÇÃO DE NOMES
VERIFICAÇÃO DE AUTORIZAÇÃO
INCORPORAÇÃO DE VISÕES
AST completa
PASSO 3
ESCOLHA DA ESTRATÉGIA DE ACESSO
GERAÇÃO DA ASL
árvores ASL (Access Specification Language) (ver ref. 4)
PASSO 4
GERAÇÃO DE CÓDIGO
código objeto (relocável)

3.3.1. PASSO 1: Análise Léxica, Sintática e Semântica Estática

A análise léxica é responsável pela tradução do comando fonte em um conjunto de elementos léxicos ("tokens").

A análise sintática será implementada usando-se o método RRP - LL (1) (5). Esse método é dirigido por tabela sintática.

A análise semântica estática do compilador SQL é feita através de chamadas de rotinas semânticas pelo algoritmo de análise sintática.

As ações semânticas, implementadas pelas rotinas semânticas, englobam as verificações de validade dos comandos SQL, levando-se em conta apenas o contexto do próprio comando.

Exemplos dessas ações seriam:
. exigir que nomes de colunas não estejam duplicados, na criação de tabelas;
. exigir que listas de usuários não contenham nomes repetidos;
. etc.

3.3.2. PASSO 2: Resolução de nomes, Verificação de Autorização e Incorporação de Visões

Durante esse passo, serão feitas todas as consultas ao dicionário de dados necessárias à compilação do comando SQL.

A resolução de nomes externos envolve consultas ao dicionário de dados para validar identificadores de tabelas, colunas, etc. Esse passo só é realizado para o comando SELECT, os de manipulação, e para os comandos ALTER e CREATE VIEW.

A verificação de autorização determina, através de consultas ao dicionário, se o usuário detém os privilégios necessários para executar o comando SQL. Esse passo só é realizado para o comando SELECT, os de manipulação e de definição.

Durante esse passo, a AST é percorrida para verificar se alguma das tabelas referenciadas é uma visão. Se fôr o caso, a AST que implementa a visão é in-

corporada, durante o passo 2, à AST que estiver sendo gerada para o comando. As AST que implementam as visões já se encontram na forma posterior ao passo 2 da compilação.

3.3.3. PASSO 3: Escolha da Estratégia de Acesso Geração da ASL

A entrada para esse passo é a AST completa gerada pelo passo 2. A AST completa já contém todas as informações, do catálogo do sistema e outras, necessárias à escolha da estratégia de acesso.

Essas informações compreendem:

- classificação dos predicados quanto a

    . fator de seletividade
    . tabelas referenciadas

- estatísticas referentes a cada coluna de tabela referenciada, que provem do Dicionário de Dados e foram incorporadas à AST pelo passo 2;

- índices existentes para cada tabela, juntamente com todas as informações inerentes aos mesmos, que provem do dicionário de dados e foram incorporados à AST pelo passo 2;

- ordens interessantes para resultados parciais, em função dos predicados de junção

A escolha da estratégia de acesso vai determinar, para cada nodo SELECT da AST:

- em que ordem as tabelas serão pesquisadas;
- qual caminho de acesso, dentre os disponíveis, será usado para cada tabela;
- qual método de junção será usado em cada caso;

Apenas dois métodos de junção são considerados, pois atendem de maneira ótima a mais de 90% dos casos (6).

O método usado para a escolha da estratégia de acesso é baseado em (7). Consiste da geração de árvores de custos de acesso pelos dois métodos de junção. Cada ramo das árvores corresponde à adição de mais uma tabela segundo um dado caminho de acesso. Ao fim de cada passo equivalente, os custos são comparados e são mantidos os ramos de menor custo que dão os mesmos resultados. O cálculo dos custos leva em conta os fatores de seletividade de cada predicado, a cardinalidade do resultado parcial e o caminho de acesso usado.

3.3.4. PASSO 4: Geração de Código

A geração de código fica extremamente facilitada pela estrutura da árvore ASL. Trata-se de percorrer os nodos e emitir o código correspondente, através de um processo dirigido por tabela. Todas as decisões quanto à estratégia de acesso já foram tomadas no passo 3.

Quase todos os nodos da ASL requerem a geração de chamadas a rotinas especializadas da biblioteca de suporte à execução. O código não é portanto, gerado em linha, e sim usando a técnica de código alinhavado (8). Alguns nodos requerem a alocação de memória que será usada durante a execução, como, por exemplo, os "buffers" para leitura e escrita de tuplas de uma tabela.

O código gerado pelo passo 4 é relocável, e será a entrada de um passo de ligação-edição, em que serão agregadas a ele as rotinas de biblioteca necessárias. Após a ligação, o código executável é armazenado em uma seção do módulo de acesso que estiver sendo construído. Caso o compilador SQL tenha sido ativado através da Interface Interativa, o código correspondente ao comando compilado poderá, então, ser executado, e os resultados serão mostrados no terminal.

3.4. Biblioteca de Suporte à Execução.

Essa biblioteca contém todas as rotinas requeridas pelo compilador SQL para implementar os diversos comandos. Inclui as rotinas aritméticas, de comparação, de conversão de tipos, etc.

3.5. Bilioteca de Primitivas de Entrada/Saída.

Essa biblioteca contêm todas as primitivas de entrada e saída para as organizações de arquivos existentes no Sistema Operacional, o que permite ao SQL manipular tais organizações.

O mesmo conjunto de primitivas é aplicável a uma organização específica para tabelas do banco de dados, cuja criação será possível quando da existência do gerenciador de bancos de dados.

3.6. Dicionário de Dados.

O dicionário de dados, que contêm todos os meta-dados do sistema é armazenado como um conjunto de tabelas, ou seja, constitui uma base de dados SQL.

A vantagem dessa abordagem é a unificação de manipulação do dicionário. Todos os recursos disponíveis para acesso às tabelas podem ser usados para o dicionário. Em particular, o usuário pode consultar o dicionário usando a própria SQL.

4. Conclusão

A linguagem SQL, apesar da sua facilidade de aprendizado e uso, ainda está aquém de uma interface totalmente interativa e amigável. Entretanto, pode servir de base a essas novas interfaces, já que é simples traduzir o resultado de interações com o usuário em um comando SQL. Dessa forma, pode-se utilizar todo o esforço de otimização do projeto SQL na implementação de ferramentas de recuperação de dados mais amigáveis ainda, como as que existem nos denominados "software integrados".

A implementação da linguagem SQL constitui, então, o ponto de partida para um projeto de um conjunto completo de facilidades para o armazenamento e recuperação de informações.

B I B L I O G R A F I A

(1) Chamberlin, D.D., et al. "SEQUEL 2: A unified approach to data definition, manipulation, and control". IBM J. Res. and Develop. 20,6 (Nov. 1976), 560-575 (also see errata in Jan. 1977 issue).

(2) Codd, E.F., "A relational model of data for large shared data banks". Comm. ACM 13,6 (June 1970), 377-387.

(3) Chamberlin, D.D., et al. "A History and Evaluation of System R". Comm. of ACM. VOL. 24, n. 10.1

(4) Lorie, R.A., and Nilsson, J.F. "An access specification language for a relational data base System". IBM J. Res. and Develop. 23,3 (May 1979), 286-298.

(5) Teles, A.A.S., De Simone, E. "Gerador de analisadores sintáticos RRP LL (1)". Anais do 8o SEMISH, julho 1981, 387-398.

(6) Blasgen, M.W., Eswaran, K.P. "On the Evaluation of Queries in a Relational Data Base System". IBM Res. Rep. RJ 1745, April 76.

(7) Selinger, P.G., et al. "Access path selection in a relational database management system". Proc. ACM SIGMOD Conf., Boston, Mass., June 1979, pp. 23-34.

(8) Bell, J.R. "Threaded Code", Comm. ACM, 16, 1973, 370-372.