Universidade Estadual de Campinas
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE TELEMÁTICA

Suporte para Áudio em Sistemas Relacionais

Autora: Ania Mayelín Montané Ramos

Orientador: Prof. Dr. Akebo Yamakami

Banca:

Profª. Drª. Marina Teresa Pires Vieira (DC/UFSCar)

Prof. Dr. Luiz Geraldo Pedroso Meloni (FEEC/UNICAMP)

Prof. Dr. Pedro Luis Dias Peres (FEEC/UNICAMP)

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos necessários para a obtenção do título de Mestre em Engenharia Elétrica.

Campinas, Outubro/2000

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

M762s

Montané Ramos, Ania Mayelin
Suporte para áudio em sistemas relacionais / Ania Mayelin Montané Ramos.--Campinas, SP: [s.n.], 2000.

Orientador: Akebo Yamakami
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Banco de dados relacionais. 2. Sistemas multimídia – Banco de dados. I. Yamakami, Akebo. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Resumo

Este trabalho propõe uma arquitetura genérica que visa o desenvolvimento de sistemas para o gerenciamento de áudio, sobre sistemas relacionais que suportem blocos binários. A arquitetura promove um tipo de dado para representar áudio nas relações, assim como interfaces apropriadas para gerenciá-lo, de maneira integrada aos dados convencionais. O objetivo principal do trabalho é oferecer uma plataforma comum que facilite o desenvolvimento de aplicações que integram áudio.

Adicionalmente, apresenta-se uma implementação da arquitetura e duas aplicações protótipo, utilizadas na validação das idéias propostas.

Abstract

This work defines a general architecture for audio management upon relational systems that support binary objects. The architecture promotes a data type to represent audio in relational tables and proper interfaces for integrated management of audio and conventional data. The main goal is to provide a common plataform to ease the work of audio application developers.

The work also describes an architecture implementation and two prototype applications that were used for validating the proposed ideas.

A Laurita

Agradecimentos

À FAPESP, que proporcionou o apoio financeiro.

A meu orientador, que aceitou trabalharmos juntos neste projeto.

A meus amigos *Sahudy* e *Luis Mariano*, que colaboraram na concepção e elaboração deste trabalho.

A meus amigos *Marta*, *Eduardo*, *Ana*, *Raul*, *Juan*, *Daynet*, *Odalys*, *Harold* e *França*, que deram-me alento e carinho.

A meus pais e irmã, que sempre estão do meu lado.

O que temos que aprender, aprendemos fazendo.
Aristotle, Ethics

Índice

1 Introdução **1**

1.1 Integração de dados multimídia aos SGBDs 2

1.2 Problema abordado 4

1.3 Trabalhos relacionados 5

1.4 Proposta do trabalho 6

2 Tópicos de Gerenciamento de Bancos de Dados **8**

2.1 Sistemas de Gerenciamento de Bancos de Dados 8

2.2 Modelo de dados relacional 12

2.3 Sumário 15

3 Arquitetura para o Gerenciamento de Áudio **16**

3.1 Motivações e objetivo 16

3.2 Princípios de projeto 17

3.3 Modelagem de dados áudio 18

3.4 Especificação 20

3.4.1 Componente MDD 21

3.4.2 Componente MMD 22

3.4.3 Componente Driver 23

3.5 Fluxograma de requisições 23

3.6 Utilização da arquitetura 24

3.7 Sumário 25

4 Uma implementação de AGA **26**

4.1 Princípios de projeto 26

4.2 Descrição do sistema 26

4.2.1 Tipo de dados para representar áudio 27

4.2.2 Interface do componente Driver 29

4.2.3 Conexão a um banco de dados 30
4.2.4 Definição de um banco de dados 32
4.2.5 Manipulação de um banco de dados 35
4.3 Modo de utilização do sistema 40
4.4 Extensões ao sistema 40
4.5 Sumário 42

5 Aplicações protótipo **43**
5.1 Implementação de um Driver 43
5.2 Interface Gráfica de SGA 45
5.3 Sistema de Recuperação de Músicas 48
5.4 Sumário 51

6 Conclusões e Trabalhos Futuros **52**

A Sintaxe do URL de um banco de dados em SGA **54**

B Domínios convencionais de SGA **55**

C Sintaxe de condições de manipulação em SGA **56**

Bibliografia **57**

Lista de Figuras

3.1 Abordagem AGA para o suporte de áudio em sistemas relacionais. 17
3.2 AGA - Componentes. 21
3.3 AGA - Fluxograma de uma requisição. 24

4.1 SGA - Relações entre as classes que permitem manipular áudio. 29
4.2 SGA - Relações entre as classes envolvidas na comunicação com um banco de dados. 31
4.3 SGA - Relações entre as classes envolvidas na definição de um banco de dados. . . 34
4.4 SGA - Tabela Auxiliar. 36
4.5 SGA - Relações entre as classes envolvidas na manipulação de um banco de dados. 38
4.6 SGA - Diagrama de utilização. 41

5.1 Relação entre SGA e o Driver desenvolvido. 44
5.2 Interface Gráfica de SGA - Interface principal. 45
5.3 Interface Gráfica de SGA - Menu Banco de Dados. 45
5.4 Interface Gráfica de SGA - Interface para criar uma tabela. 46
5.5 Interface Gráfica de SGA - Menu Tabela. 46
5.6 Interface Gráfica de SGA - Interface para consultar. 47
5.7 Interface Gráfica de SGA - Interface para examinar dados recuperados. 47
5.8 Sistema de Recuperação de Músicas - Interface principal. 49
5.9 Sistema de Recuperação de Músicas - Interface para examinar dados recuperados. . 50
5.10 Sistema de Recuperação de Músicas - Interface para mostrar informação de um áudio. 50
5.11 Sistema de Recuperação de Músicas - Interface para exportar um áudio. 50

Capítulo 1

Introdução

Nos últimos tempos assiste-se a uma explosão de sistemas de informação que incorporam dados multimídia tais como áudio, vídeo, imagens e animações. Esses sistemas cresceram rapidamente em uma variedade de áreas, incluindo entretenimento, medicina, educação e serviços. Eles fazem uma melhor fixação do conteúdo, têm maior riqueza de detalhes e conseguem uma comunicação mais eficiente que os sistemas tradicionais.

Os dados multimídia superam em complexidade o conjunto dos tipos de dados alfanuméricos (Grosky, 1994). Esses dados convencionais são utilizados para representar informação simbólica. Os dados multimídia expõem informação mais próxima à realidade física ou virtual. Eles são utilizados para representar informação áudio-visual.

Áudio, vídeo e animação são dados dependentes do tempo, conhecidos na literatura como dados contínuos e/ou dinâmicos. Isto é, a sua interpretação está associada a uma escala de tempo. Por exemplo: um sinal de áudio está formado por um conjunto de amostras seqüenciadas temporalmente; sua interpretação é obtida a partir da interpretação no tempo das amostras que o constituem. Esses dados contínuos impõem restrições temporais aos componentes *software* e *hardware* que os manipulam.

Os dados imagem, vídeo e gráficos, entre outros, apresentam relações espaciais. Isto é, objetos individuais entre esses dados se relacionam espacialmente. Por exemplo, em uma imagem com dois objetos (A e B) pode acontecer que A esteja contida em B ou A esteja à esquerda de B.

Os dados multimídia ocupam grandes volumes de espaço de armazenamento devido à alta densidade de informação presente. Freqüentemente, a eles são associadas técnicas de compressão, requerendo operações complexas para sua manipulação. Atualmente, o seu gerenciamento eficiente apresenta-se como um grande desafio.

Sabe-se que os Sistemas de Gerenciamento de Banco de Dados (SGBDs) são altamente recomendados para serem utilizados em conjunto com as aplicações que manipulam

informação, pois eles oferecem vantagens muito atraentes (Silberschatz et al., 1997), incluindo: transparência dos aspectos de armazenamento físico, independência e consistência dos dados, facilidades de consulta, acesso multiusuário e um repositório de dados para ser compartilhado por várias aplicações.

Considerando as vantagens dos SGBDs, é desejável integrar dados multimídia a esses sistemas.

1.1 Integração de dados multimídia aos SGBDs

Os modelos de bancos de dados tradicionais foram concebidos para estruturar bancos de dados alfanuméricos. O crescente aumento de aplicações que integram dados mais complexos vem estimulando o desenvolvimento de novas tecnologias de bancos de dados. Os modelos mais destacados são o relacional (Codd, 1990) e o orientado a objeto (Kim, 1990). Diversas pesquisas os tomam como base na busca de soluções para o gerenciamento dos novos tipos de informação.

Um dos desafios atuais dos SGBDs Relacionais (SGBDRs) e SGBDs Orientados a Objetos (SGBDOOs) é incluir a capacidade de gerenciar dados multimídia de maneira integrada aos dados convencionais. Considerando a complexidade dos dados multimídia, sistemas projetados para gerenciá-los devem incorporar mais funcionalidades que as oferecidas para os dados tradicionais (Grosky, 1998).

Diversas abordagens são utilizadas para integrar dados multimídia a esses sistemas, destacando-se o esquema de blocos binários e a introdução do paradigma de orientação a objeto (Campbell e Chung, 1996; Silberschatz et al., 1997).

Blocos binários

As estruturas propostas no modelo relacional não são adequadas para representar dados complexos, pois ele limita-se a representar valores atômicos e simples. O modelo foi estendido para suportar grandes blocos binários - conhecidos na literatura como *binary large objects (blob)* ou *long field* - que permitem representar dados complexos nas relações como valores atômicos, não estruturados.

Um *blob* é um campo sem tipo, freqüentemente de tamanho grande e variável. Quando um *blob* é requisitado, ele é simplesmente entregue à aplicação, a qual é responsável pelo seu processamento "inteligente". Quer dizer, a aplicação fica encarregada da interpretação dos blocos binários.

Os dados multimídia podem ser armazenados como atributos *blob* numa relação. A vantagem desse esquema é que oferece uma mínima integração dos dados tradicionais e

multimídia. Entretanto, um problema associado é que o sistema de gerenciamento não conhece a estrutura do objeto binário e, por isso, não pode interpretar nem manipular o seu conteúdo.

Modelo de dados orientado a objeto.

A utilização do paradigma de orientação a objeto em bancos de dados apresenta-se como uma solução às desvantagens do modelo relacional na modelagem de dados complexos. Os princípios de orientação a objeto (objeto, classe, encapsulamento, herança, polimorfismo, etc.) possibilitam uma modelagem mais rica das informações (Nierstrasz, 1989). Esse paradigma oferece um *framework* para os usuários definirem tipos de dados e modelarem flexivelmente seu universo de informação.

O modelo de banco de dados orientado a objeto integra características de sistemas de bancos de dados (persistência, concorrência, recuperação, facilidade de consulta, etc.) e princípios de orientação a objetos. Ele oferece mecanismos para definir novos tipos de dados que encapsulam sua estrutura e comportamento, assim como suporte para consultá-los (Atkinson et al., 1992).

Os princípios de orientação a objeto permitem uma melhor modelagem dos dados multimídia, pois facilitam definições mais ricas sobre seu formato e semântica. Logo, os sistemas baseados neste modelo conseguem uma melhor representação de sua estrutura, comportamento e operações associadas.

Apesar deste modelo apresentar-se como uma alternativa promissora para representar e manipular dados multimídia, os sistemas baseados nele não provêem inerentemente técnicas e métodos novos para o armazenamento e manipulação desses dados (Campbell e Chung, 1996; Pazandak e Srivastava, 1997). Adicionalmente, o desenvolvimento de SGBDOOs não tem apresentado grandes avanços, como conseqüência da falta de uma base teórica forte (Dittrich e Dittrich, 1995).

Modelo de dados relacional-objeto

As limitações do modelo relacional e seu mecanismo de blocos binários para representar dados complexos conduz a novas extensões no modelo relacional. O modelo relacional-objeto estende o modelo relacional, oferecendo um esquema de tipos mais completo. As novas extensões incluem a metodologia de orientação a objeto para criar tipos de dados, assim como extensões à linguagem de manipulação relacional para seu suporte. Dessa maneira, permite-se que um atributo de uma relação seja um tipo de dados complexo.

Os SGBDRs Objetos (SGBDROs) - também conhecidos como SGBDRs extensíveis - permitem a um usuário fazer as próprias extensões, incluindo definição de tipos de dados e

operações associadas, operadores, funções e métodos de acesso (Stonebraker e Brown, 1999).

As facilidades dos sistemas extensíveis podem ser utilizadas para definir estruturas que representem os diferentes tipos de dados multimídia e operações sobre eles. Esses dados podem ser armazenados no banco de dados utilizando o mecanismo de blocos binários.

Os sistemas extensíveis têm como vantagem principal herdarem as capacidades de manipulação de dados dos sistemas relacionais, já que preservam a base relacional, em particular, o acesso declarativo aos dados. Adicionalmente, eles integram o poder do paradigma de orientação a objeto para a modelagem de dados.

No entanto, esses sistemas requerem um forte suporte para manipular os dados complexos (Aoki, 1999). Várias questões ainda são objeto de pesquisa e amadurecimento, como por exemplo, o processamento e otimização de consultas, e o acesso rápido às extensões dos usuários. Além disso, o incremento no poder da linguagem de consulta traz como conseqüência uma perda de desempenho, já que, por exemplo, um nome de uma função definida por um usuário pode implicar várias operações de junção.

Por outra parte, o fato das aplicações terem que definir as extensões próprias para representar dados multimídia, limita a uniformidade quanto à modelagem desses dados.

1.2 Problema abordado

As pesquisas de gerenciamento de dados multimídia têm colocado maior ênfase em texto e imagem. A integração de áudio aos SGBDs têm recebido menor atenção, provavelmente, por sua dependência temporal que dificulta sua representação e manipulação. Além disso, a análise do conteúdo desses dados é difícil, limitando a recuperação por seu conteúdo (Yoshitaka e Ichikawa, 1999).

Certamente, as aplicações que pretendem o gerenciamento de dados áudio e convencionais podem ser beneficiadas com a infra-estrutura oferecida pelos sistemas relacionais. O modelo relacional possui um forte suporte teórico para a descrição da representação e manipulação de dados tradicionais. Para a maioria das aplicações não é conveniente perder a integração com os dados tradicionais, mesmo que o áudio seja uma informação fundamental.

As soluções oferecidas pelos sistemas relacionais quanto a integração de dados multimídia, incluindo áudio, apresentam desvantagens que foram citadas na seção anterior.

Logo, pode-se concluir que as aplicações precisam mais que um mecanismo de blocos binários para armazenar os dados áudio. Elas precisam de um tipo de dados para representar os dados áudio de seu universo de informação. O tipo de dados deve modelar adequadamente essa mídia, representando sua estrutura e operações.

As aplicações também requerem interfaces apropriadas para representar áudio nas relações e para manipulá-lo, utilizando as operações definidas sobre esses dados, de maneira integrada aos dados convencionais.

Adicionalmente, acredita-se que as aplicações podem beneficiar-se com um tipo de dados áudio genérico, cujo projeto esteja em função de um serviço e não do domínio das aplicações. Igualmente, as aplicações podem beneficiar-se com interfaces de propósito geral, para gerenciar áudio nos sistemas relacionais. Isso sem comprometer o desempenho desses sistemas.

1.3 Trabalhos relacionados

Atualmente, existem diversos SGBDROs protótipos e comerciais. Cada um deles oferece determinados mecanismos para a extensibilidade do sistema, permitindo a um usuário a definição de suas extensões para um domínio específico. Os sistemas incorporam as extensões para operar sobre os dados complexos.

Existem diversas abordagens direcionadas à incorporação de áudio, como as mostradas a seguir.

- PREDATOR (Seshadri, 1998; PREDATOR Development Group, 2000) é um sistema desenvolvido na Universidade de Cornell, New York, com propósitos de pesquisa. Ele promove a extensibilidade de tipos de dados baseados na tecnologia de tipos de dados abstratos "estendidos". O termo estendido refere-se à exposição da semântica dos tipos para ser utilizada na otimização de consultas (Seshadri et al., 1997).

  O processo de adicionar um novo tipo de dados não é simples. O desenvolvedor de um tipo deve definir os métodos para representar os dados na memória principal e no disco. Ele deve especificar meta-informação, se requerida. Além disso, ele deve registrar regras de otimização individuais.

  O sistema foi estendido com um tipo de dados para representar áudio. O tipo de dados limita-se a representar arquivos áudio no formato AU (formato da Sun) e não oferece métodos para processar o conteúdo de um áudio, nem para apresentá-lo.

- Informix Dynamic Server (Informix, 2000) permite a extensão do sistema através da criação de módulos chamados *DataBlades*. O desenvolvedor de um *DataBlade* deve estimar os custos das rotinas definidas e informá-los ao sistema. Essa informação é utilizada para otimizar as consultas.

  Informix Dynamic Server foi estendido com um módulo para o suporte de áudio. O *AIRDataBlade (Audio Information Retrieval DataBlade)* (Informix & Muscle Fish,

2000) permite a consulta baseada no conteúdo de áudio. Os arquivos são analisados automaticamente quando inseridos no banco de dados, construindo-se um vetor de características acústicas-perceptuais, que é armazenado no banco de dados. A busca por conteúdo de áudio é guiada através desse vetor. O módulo não provê suporte para apresentar e acessar temporalmente esses dados.

- DB2 Universal Database (IBM, 2000) permite estender o sistema através da criação de *Extenders*. O desenvolvedor de um *Extender* deve criar índices sobre os tipos de dados que define para facilitar a otimização de consultas. Ele deve colecionar estatísticas sobre os custos das funções que define e os métodos de acesso que incorpora para serem utilizadas na otimização.

  O sistema incorporou um *Extender* que possibilita consultar dados áudio. O *AudioExtender* mantém um conjunto de atributos relacionados ao formato de arquivos áudio (tipo de conteúdo, número de canais, taxa de amostragem, etc.). Esses atributos podem ser utilizados para consultar os dados. O *AudioExtender* também permite apresentar dados áudio. No entanto, ele não oferece operações que processem o conteúdo de áudio e que possam ser utilizadas em consultas. Os áudios são mantidos em um servidor de arquivos.

Cada um desses sistemas oferece soluções próprias para os usuários os estenderem. O fato de não existir um modelo relacional-objeto unificado vem dificultando sua aceitação.

Esses sistemas não têm ainda suporte eficiente para o processamento de consultas que envolvem dados complexos. A integração de bibliotecas de domínio específico (DataBlades, Extenders, etc.) afeta seriamente o processamento de consultas e o desempenho dos sistemas.

As extensões para integrar áudio propostas por esses sistemas mudam muito de um sistema para outro, pois não existe um modelo de dados áudio comum para gerenciá-los.

1.4 Proposta do trabalho

Este trabalho propõe AGA (Arquitetura Genérica para o Gerenciamento de Áudio), uma arquitetura de propósito geral, que permite definir sistemas de gerenciamento de dados áudio sobre sistemas relacionais que suportam blocos binários.

AGA promove um modelo de dados áudio genérico, para descrever esses dados. A arquitetura propõe o desenvolvimento de interfaces genéricas para representar esses dados nas relações e manipulá-los de maneira integrada aos dados convencionais. A generalidade do modelo e das interfaces propostas refere-se a sua independência do domínio das aplicações.

O propósito da arquitetura é fornecer uma plataforma comum para o desenvolvimento de aplicações que gerenciam áudio. A arquitetura visa a criação de ferramentas flexíveis e extensíveis, que facilitarão aos programadores o desenvolvimento dessas aplicações.

O trabalho apresenta como contribuições práticas o SGA (Sistema de Gerenciamento de Áudio), uma implementação da arquitetura proposta, e duas aplicações protótipo.

O trabalho enfoca o gerenciamento de dados áudio, porém as idéias propostas pela arquitetura para integrar áudio podem ser adaptadas para integrar outras mídias.

No próximo capítulo são expostos alguns tópicos de gerenciamento de bancos de dados. O Capítulo 3 descreve a arquitetura para o suporte de áudio, especificando sua estrutura e funcionalidade. No Capítulo 4 apresenta-se uma implementação da arquitetura proposta. O Capítulo 5 descreve duas aplicações protótipo que utilizam o sistema implementado, para efeito de validação das idéias propostas. O Capítulo 6 apresenta as conclusões e trabalhos futuros para melhorar e estender a arquitetura e o sistema implementado.

Capítulo 2

Tópicos de Gerenciamento de Bancos de Dados

Este capítulo apresenta brevemente alguns tópicos de gerenciamento de bancos de dados, considerados importantes no entendimento do trabalho. A Seção 2.1 expõe aspectos gerais relacionados a sistemas de bancos de dados. A Seção 2.2 descreve resumidamente o modelo de dados relacional.

2.1 Sistemas de Gerenciamento de Bancos de Dados

A tecnologia de banco de dados contribuiu no crescimento da utilização dos sistemas de computação. Os bancos de dados desempenham um papel importante na maior parte das áreas onde os computadores são utilizados, incluindo comércio, engenharia, medicina e educação. A seguir são apresentados aspectos relacionados a sistemas de bancos de dados segundo Elmasry e Navathe (1994) e Silberschatz et al. (1997).

Banco de dado e SGBD

Um **banco de dados** é uma coleção de dados relacionados coerentemente, representando um universo de informação determinado do mundo real. Eles são projetados, construídos e povoados com dados para um propósito específico. Podem ser de qualquer tamanho e de complexidade variável.

Um **Sistema de Gerenciamento de Banco de Dados (SGBD)** é um conjunto de programas que permite criar, manter e acessar um banco de dados. O objetivo primário de um SGBD é prover um ambiente conveniente e eficiente para armazenar e recuperar informações de um banco de dados.

Antes do advento dos SGBDs, o gerenciamento de dados era feito, tipicamente, através de sistemas de processamento de arquivos. Esses sistemas apresentam sérias desvantagens, tais como redundância e inconsistência dos dados, dificuldades para acessar os dados, problemas de integridade, anomalias por acesso concorrente e problemas de segurança. Os SGBDs têm como propósito oferecer as funcionalidades necessárias para solucionar esses problemas.

O gerenciamento de dados envolve a definição de estruturas para armazená-los e acessá-los, assim como o fornecimento de mecanismos para sua manipulação. Além disso, os sistemas de gerenciamento devem garantir a segurança (recuperação a falhas, acesso restrito) e consistência (dados corretos) do banco de dados.

Visão abstrata dos dados

Os sistemas de bancos de dados pretendem fornecer aos usuários uma **visão abstrata** dos dados. Eles ocultam certos detalhes do armazenamento e manutenção dos dados através da definição de vários níveis de abstração. O **nível físico** (nível menor de abstração) descreve como os dados são armazenados fisicamente. O **nível lógico** (nível intermediário) representa os dados e as relações entre eles em termos de estruturas relativamente simples. Por último, o **nível de visão** (nível maior de abstração) especifica somente parte do banco de dados, pois diferentes usuários podem precisar diferentes visões do banco de dados.

Um sistema de banco de dados tem diversos esquemas, de acordo com os níveis de abstração dos dados: **esquema físico**, **esquema lógico** e **subesquema**, respectivamente. Em geral, um SGBD suporta um esquema físico, um esquema lógico e diversos subesquemas. A **independência dos dados** é definida como a habilidade de modificar a definição de um esquema em um nível, sem afetar a definição de um esquema no nível superior.

Modelos de dados

A estrutura de um banco de dados é descrita por um **modelo de dados** que é uma coleção de ferramentas conceituais para descrever os dados, as suas relações, semântica e restrições. Existem vários modelos de dados e podem ser agrupados em: **modelos lógicos orientados a registro** e **modelos lógicos orientados a objeto**.

Os modelos lógicos orientados a registro são utilizados para descrever os dados nos níveis lógico e de visão. Esses modelos permitem especificar a estrutura lógica global do banco de dados e fornecer descrições de implementação. Eles recebem esse nome porque o banco de dados é estruturado como uma coleção de registros. Os modelos relacional, de redes e hierárquico são exemplos de modelos orientados a registro.

Os modelos lógicos orientados a objeto são utilizados também para descrever os dados nos níveis lógico e de visão. Eles recebem esse nome porque representam as entidades do mundo real como objetos com propriedades e comportamento. Esses modelos possibilitam representar os dados de maneira flexível e permitem especificar restrições aos dados. O modelo orientado a objeto é um exemplo de modelo lógico orientado a objeto.

Esquema e instância de um banco de dados

O **esquema de um banco de dados** é o projeto global do banco de dados, estruturado através de um determinado modelo. Uma **instância do banco de dados** é a coleção de informação armazenada no banco de dados em um instante determinado. Um banco é atualizado à medida que informações são inseridas, removidas ou modificadas.

Linguagens de bancos de dados

Um sistema de banco de dados fornece dois tipos diferentes de linguagens: **Linguagem de Definição de Dados (LDD)** e **Linguagem de Manipulação de Dados (LMD)**.

A LDD permite especificar o esquema de um banco de dados. As estruturas de armazenamento e métodos de acesso são especificados através de um tipo especial de LDD.

A LMD permite aos usuários recuperar informação e atualizar um banco de dados. Uma consulta (*query*) é uma instrução requisitando informação. A parte da LMD que envolve a recuperação de informação é conhecida como linguagem de consulta.

Existem dois tipos de LMD: procedimental (o usuário descreve o resultado desejado e especifica um conjunto de operações sobre o banco de dados para obtê-lo) e não-procedimental (o usuário descreve o resultado desejado sem especificar como será obtido).

Gerenciamento de transações

Um conceito importante dentro de um SGBD é **transação**. Define-se como uma coleção de operações (especificadas por um programador de aplicação) que desenvolvem uma função lógica simples sobre o banco de dados.

Uma transação deve ser uma unidade de **atomicidade** (a execução dela deve ser completada com sucesso ou não deve ser executada) e **consistência** (não deve violar as restrições de consistência do banco de dados). Adicionalmente, a transação deve ser executada com certo nível de **isolamento** (suas mudanças são visíveis a outras transações após encerrar sua execução) e ela deve ser **durável** (mudanças no banco de dados não devem ser perdidas por falhas posteriores à mudança).

O programador é responsável pela definição adequada das transações de modo que elas preservem a consistência dos dados. O sistema deve cuidar da atomicidade e durabilidade das transações. Em ausência de falhas, todas as transações podem completar suas execuções com sucesso. Porém, vários tipos de falhas podem acontecer e é provável que algumas transações não se completem. O SGBD é responsável pela deteção de falhas e recuperação adequada do banco de dados após uma falha.

Alguns SGBDs são projetados para suportar **concorrência** de transações (múltiplas transações acessando o banco de dados concorrentemente). A consistência de um banco de dados pode ser seriamente afetada caso essa concorrência não seja devidamente controlada. O SGBD é responsável por esse controle. O esquema de isolamento das transações é determinado pelo mecanismo de controle de concorrência.

Gerenciamento de armazenamento

Tipicamente, os bancos de dados requerem espaço de armazenamento grande, sendo armazenados em discos. Os SGBDs movem os dados entre os discos e a memória principal à medida que eles são requisitados. Devido a que o movimento de dados entre um disco e a memória principal é lento, é necessário que os sistemas estruturem os dados de forma tal que minimize o número de movimentos.

Os sistemas de bancos de dados devem fornecer a interface entre os dados armazenados (arquivos) em um banco de dados e os programas de aplicações que manipulam o banco de dados. Eles são responsáveis pela interação com o gerenciador de arquivos. Logo, os SGBDs são responsáveis pelo armazenamento, recuperação e atualização dos dados do banco de dados.

Estrutura global

Um sistema de banco de dados é particionado em módulos responsáveis pelas diferentes funcionalidades do sistema. Alguns desses módulos são resumidos a seguir.

- Interpretador da LDD: interpreta as instruções da LDD e armazena as informações que descrevem a estrutura do banco de dados.
- Compilador da LMD: traduz as instruções da LMD em uma linguagem compreensível pelo mecanismo de avaliação e execução de consultas; adicionalmente, o compilador tenta transformar uma requisição de um usuário em uma requisição equivalente, porém de execução mais eficiente.
- Mecanismo de avaliação e execução de consultas: executa as instruções geradas pelo compilador da LMD.

- Gerenciador de transações: deteta falhas e garante a recuperação adequada do banco de dados após uma falha; controla a execução de transações concorrentes para garantir a consistência dos dados.
- Gerenciador de arquivos: gerencia a alocação de espaço no disco e as estruturas de dados utilizadas para representar a informação armazenada no disco.
- Manipulador de *buffer*: movimenta os dados entre o disco e a memória principal.

Adicionalmente, diversas estruturas de dados formam parte dos sistemas de bancos de dados, tais como: arquivos de dados (contêm os dados do banco de dados); dicionário (contém as informações que descrevem o esquema de um banco de dados); e métodos de acesso (permitem o acesso rápido aos dados).

Hoje em dia, é comum as aplicações que manipulam grandes quantidades de informação utilizarem os SGBDs. É importante destacar que o modelo relacional e sua linguagem de gerenciamento contribuíram para o desenvolvimento e utilização desses sistemas (Du e Wolfe, 1997). A próxima seção descreve resumidamente esse modelo.

2.2 Modelo de dados relacional

O modelo relacional foi proposto por Codd em 1970. Ele é baseado em uma estrutura de dados simples e uniforme, e tem uma base matemática sólida (Date, 1986; Codd, 1990). Um banco de dados é dito relacional se sua estrutura for descrita através do modelo relacional.

Estrutura de bancos de dados relacionais

O modelo relacional apresenta os bancos de dados como tabelas. Cada tabela recebe um nome único e é formada por uma ou mais colunas. As colunas também são identificadas com um nome único, chamado de cabeçalho. Todos os valores em uma mesma coluna são do mesmo tipo de dados. Cada linha representa uma coleção de valores relacionados. Esses valores podem ser interpretados como fatos que descrevem o mundo real.

Neste modelo, as linhas de uma tabela são chamadas de **tuplas**, o cabeçalho de uma coluna é chamado de **atributo** e uma tabela é chamada de **relação**. O tipo de dados que descreve o conjunto de valores que podem aparecer em cada coluna é chamado de **domínio**. O domínio de uma coluna deve ser atômico (seus elementos são unidades indivísiveis).

Um **esquema de relação** descreve uma relação e o conjunto de atributos que a compõe. Cada atributo toma valores no seu domínio. Uma **instância de relação** é definida como um subconjunto do produto cartesiano dos domínios especificados no esquema de relação.

Dado que uma relação é definida como um conjunto de tuplas, todas as tuplas de uma relação devem ser diferentes (os elementos de um conjunto são distintos). Assim sendo, é definida como **chave primária** de um esquema de relação o atributo ou conjunto de atributos que identifica unicamente uma tupla em uma instância desse esquema de relação.

Freqüentemente, um banco de dados relacional é formado por várias tabelas relacionadas. **O esquema de um banco de dados relacional** é formado por um conjunto de esquemas de relações e um conjunto de **restrições de integridade** (mecanismos que garantem que as atualizações dos dados não afetam sua consistência). Uma **instância de um banco de dados** é um conjunto de instâncias de relações do banco de dados.

Operações e linguagens relacionais

Os usuários de um banco de dados relacional precisam de meios para manipulá-lo. O modelo relacional define operações sobre um banco de dados, que podem ser divididas em dois grupos: **operações de atualização** e **operações de recuperação.**

As operações de atualização definidas no modelo permitem fazer mudanças no banco de dados. Existem três operações básicas neste grupo: Inserir e Remover tuplas, assim como Modificar valores de atributos de tuplas. Sempre que operações de atualização forem aplicadas, as restrições especificadas sobre o esquema do banco de dados relacional não devem ser violadas.

As operações de recuperação permitem consultar relações para recuperar informação. O modelo define duas linguagens de consulta formais equivalentes: Álgebra Relacional e Cálculo Relacional.

A álgebra relacional é uma linguagem de consulta procedimental. As operações dessa linguagem podem ser divididas em dois grupos. Um grupo inclui as **operações básicas da teoria matemática de conjuntos**: União, Diferença, Interseção e Produto Cartesiano. O outro grupo inclui **operações desenvolvidas especificamente para bancos de dados relacionais**: Seleção, Projeção e Junção.

As operações união, diferença e interseção são definidas sobre duas relações com estruturas compatíveis (possuem igual número de atributos e cada par de atributos correspondente tem o mesmo domínio).

A união de duas relações produz uma relação que contém as tuplas da primeira relação e as tuplas da segunda relação. A diferença de duas relações dá como resultado uma terceira relação que contém as tuplas da primeira relação que não estão presentes na segunda relação. A interseção de duas relações gera uma terceira relação que contém as tuplas que estão presentes em ambas as relações. A operação do produto cartesiano de duas relações gera uma nova relação com todas as combinações possíveis entre as duas relações originais.

A seleção sobre uma relação permite selecionar tuplas dentre as pertencentes à relação. A projeção de colunas de uma relação gera uma nova relação com apenas os valores das colunas selecionadas. Por último, a operação junção de duas relações tem como resultado uma relação cujas tuplas são a combinação de tuplas relacionadas das duas relações originais.

O cálculo relacional é uma linguagem fundamentada no ramo da lógica matemática chamado de cálculo de predicados. Essa linguagem é baseada em construções não-procedimentais, utilizando descrições formais do resultado desejado. Existem duas variantes básicas nas quais o cálculo de predicado pode ser adaptado a uma linguagem para banco de dados relacionais: Cálculo Relacional de Tuplas e Cálculo Relacional de Domínios.

As linguagens relacionais, matemáticas, ofereceram os fundamento para o desenvolvimento de linguagens computacionais de gerenciamento de bancos de dados. Os sistemas comerciais utilizam linguagens amigáveis, tais como SQL (*Structured Query Language*), QBE (*Query-by-Example*) e Quel.

SQL estabeleceu-se como a linguagem padrão dos sistemas de bancos de dados relacionais. Essa linguagem utiliza uma combinação de construtores da álgebra relacional e do cálculo relacional. Embora seja referida como linguagem de consulta, SQL também inclui mecanismos para definir estruturas de dados, atualizar o banco de dados e especificar restrições de segurança. Por isso, SQL é considerada linguagem de definição de dados e linguagem de manipulação de dados.

QBE é baseada no cálculo relacional de domínios. As consultas são expressas através de exemplos (o usuário descreve os dados que deseja recuperar através de um exemplo). Quel é baseada no cálculo relacional de tuplas. Ela é similar a SQL em alguns aspectos, porém evita complexidades de SQL. QBE e Quel também incluem capacidades para definir um banco de dados, especificar restrições de integridade e atualizar os dados.

Extensões do modelo

O surgimento de aplicações que integram dados complexos (multimídia, espaciais, estruturas matemáticas, etc.) demanda o gerenciamento desses novos dados. O requerimento de domínios atômicos do modelo relacional limita a representação de dados complexos.

Um grande volume de pesquisa tem sido dedicado a estender o modelo com o objetivo de permitir a integração de dados complexos. As principais extensões descritas em diversas propostas incluem os mecanismos descritos a seguir (Silberschatz et al., 1997).

- Relações aninhadas: permite que o domínio de um atributo seja uma relação. Um dado complexo pode ser representado por uma tupla de uma relação aninhada.

- Blocos binários: permite armazenar dados complexos como atributos atômicos, não estruturados. A aplicação é responsável pela sua interpretação.
- Princípios de orientação a objeto: permite a criação de tipos de dados que encapsulam a estrutura (atributos) e o comportamento (métodos) de objetos complexos.
- Tipos coleções: permite definir atributos de tipo conjunto, listas ou arranjo de valores.

Os diferentes mecanismos propostos para estender o sistema de tipos do modelo relacional impõem a necessidade de estender as linguagens de definição e manipulação de bancos de dados para seu suporte.

2.3 Sumário

Este capítulo apresentou aspectos relacionados ao gerenciamento de bancos de dados. Expuseram-se de maneira resumida conceitos gerais de sistemas de bancos de dados. O modelo relacional foi exposto brevemente, resumindo a estrutura básica de um banco de dados relacional, as principais linguagens relacionais e algumas extensões do modelo para suportar a integração de dados complexos.

O próximo capítulo apresenta uma arquitetura para o suporte de áudio em sistemas relacionais.

Capítulo 3

Arquitetura para o Gerenciamento de Áudio

Este capítulo apresenta AGA (Arquitetura Genérica para o Gerenciamento de Áudio), uma arquitetura de propósito geral que visa o desenvolvimento de sistemas de gerenciamento que integram dados convencionais e áudio.

A primeira seção mostra as motivações e o objetivo da arquitetura proposta e faz uma descrição geral da mesma. A segunda seção expõe os princípios de projeto de AGA. Na terceira seção, aborda-se a modelagem de dados áudio. Por último, especifica-se a estrutura de AGA, assim como seu funcionamento.

3.1 Motivações e objetivo

A falta de modelos de dados para integrar áudio a bancos de dados dificulta o desenvolvimento de aplicações que incorporam essa mídia. Os sistemas relacionais oferecem o mecanismo de blocos binários para armazenar áudio, porém deixam às aplicações a responsabilidade da interpretação desses blocos. Os sistemas relacionais-objeto permitem que os usuários criem as próprias extensões para o suporte desses dados, mas ainda não suportam eficientemente a integração dessas extensões. Além disso, os mecanismos de extensibilidade não são simples e mudam muito de um sistema para outro, dificultando uma maior aceitação desses sistemas.

Como uma alternativa para integrar áudio a sistemas relacionais, é proposta AGA, uma arquitetura que tem como objetivo oferecer uma plataforma comum para os programadores desenvolverem, de maneira simples, aplicações que integram áudio e dados convencionais. AGA permite definir sistemas com funcionalidades para gerenciar bancos de dados relacionais que integram áudio.

Segundo ilustrado na Figura 3.1, AGA é definida como uma camada intermediária entre um SGBDR e uma aplicação. O único requerimento sobre o SGBDR é oferecer suporte para blocos binários de tamanho grande e variável, para facilitar o armazenamento dos dados áudio.

Sendo assim, aproveitam-se as vantagens oferecidas pelo SGBDR no gerenciamento de dados convecionais e seu mecanismo de armazenamento de blocos binários. Adicionalmente, oferecem-se, aos programadores de aplicações, funcionalidades necessárias para a definição e manipulação de bancos de dados relacionais que integram dados convencionais e áudio.

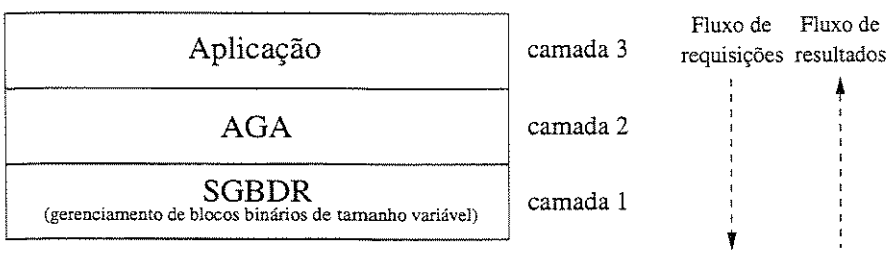

Figura 3.1: Abordagem AGA para o suporte de áudio em sistemas relacionais.

Na primeira camada está o SGBDR que oferece à camada superior suporte para dados convencionais e blocos binários. A segunda camada corresponde a AGA, que oferece interfaces à aplicação para a definição e manipulação de um banco de dados relacional que integra dados convencionais e áudio. Essa camada é responsável pela definição e manipulação de blocos binários (utilizando o SGBDR) para a integração de áudio. A terceira camada corresponde a uma aplicação, que gerencia seus dados através das funcionalidades oferecidas por AGA.

A arquitetura pretende oferecer aos desenvolvedores meios simples para criar aplicações que incorporam áudio. Para a aplicação, a forma real de armazenamento e recuperação de dados áudio será transparente. O programador somente considerará a existência de um tipo de dados adequado para representar áudio em um banco de dados relacional, e como tal vai manipulá-lo junto aos dados convencionais.

3.2 Princípios de projeto

Pretende-se definir uma arquitetura de propósito geral para sistemas de gerenciamento de áudio sobre SGBDRs. Os princípios de projeto de AGA são resumidos a seguir.

- Promover um modelo de dados áudio.

- Permitir o gerenciamento de dados convencionais e áudio de maneira integrada em bancos de dados relacionais.
- Ser independente do domínio das aplicações.
- Ser independente de um SGBDR determinado.

A integração de áudio em um banco de dados relacional impõe oferecer um tipo de dados que modele as informações áudio, assim como interfaces adequadas para gerenciar os atributos áudio, em junção com os atributos convencionais.

Para a arquitetura ser de propósito geral, deve-se modelar um tipo de dados áudio genérico, que represente a estrutura e o comportamento básicos de um áudio digital. Igualmente, é preciso oferecer interfaces com operações genéricas de gerenciamento de bancos de dados. A independência da arquitetura com respeito a um SGBDR específico possibilita definir uma plataforma comum para desenvolver aplicações que integram áudio.

3.3 Modelagem de dados áudio

AGA promove um modelo de dados áudio genérico, que permita representar as características que descrevem a estrutura dos dados, assim como seu comportamento lógico.

A modelagem das características de áudio deve considerar a existência de diversos formatos de representação de áudio. Esses formatos identificam os dispositivos de gravação e reprodução da mídia, e são caracterizados por vários parâmetros (Strawn, 1994; Gibbs e Tsichritzis, 1995), incluindo os listados a seguir.

- Taxa de amostragem: especifica a freqüência de amostragem do sinal analógico (amostras por segundo). Hoje em dia, é muito comum se utilizarem as taxas de amostragem de áudio 8 kHz e 44.1 kHz.
- Tamanho da amostra: especifica o número de bits utilizados para armazenar uma amostra do áudio. Amostras de 8 bits e 16 bits são utilizadas freqüentemente na digitalização de áudio.
- Codificação: especifica o método de representação digital dos dados. Existem diferentes métodos de codificação que permitem reduzir os custos de transmissão e armazenamento. Linear, ULAW, ALAW e MPEGLAYER3 são esquemas de codificação utilizados amplamente.
- Canais: especifica o número de canais utilizados na gravação/reprodução do áudio. Existem sinais mono (um canal), *stereo* (utilizam dois canais: esquerdo e direito) e sinais multicanais (utilizam mais de dois canais).

A modelagem do comportamento lógico de um áudio digital deve considerar as operações próprias da mídia, assim como os requerimentos das aplicações que incorporam esses dados.

A seguir são apresentadas algumas das operações próprias de áudio (Strawn, 1994; Gibbs e Tsichritzis, 1995).

- Apresentação: consiste em recuperar e apresentar rapidamente segmentos de áudio digital. Os problemas básicos associados à recuperação estão relacionados à localização e leitura dos dados, de maneira que se garanta um fluxo contínuo de amostras.

- Edição: envolve cortar, copiar e inserir segmentos de áudio. Existem diversas técnicas de edição, desenvolvidas com o objetivo de preservar a continuidade do sinal de áudio durante sua edição.

- Efeitos e filtragem: consiste na aplicação de efeitos especiais sobre áudio através de técnicas de filtragem digital. Por exemplo: redução de ruído, *stereoization* e aplicação de ambientes acústicos.

- Conversão de formatos: operações sobre áudio digital lidam com conversão de um formato a outro, ou alteração de parâmetros de codificação dentro de um formato. Por exemplo, compressão/descompressão de áudio.

As aplicações que incorporam áudio requerem um conjunto de operações para manipularem esses dados. Geralmente, elas baseiam-se nas operações próprias dessa mídia. Algumas dessas operações são listadas a seguir.

- Requisição de características: permite inquerir atributos que descrevem um áudio e seu formato. Por exemplo: inquerir a duração, taxa de amostragem, método de codificação, volume, etc.

- Requisição de semântica: permite inquerir informação semântica do conteúdo de um áudio, assim como estabelecer relações entre audios por sua semântica. Por exemplo: determinar similaridade entre dois áudios; determinar se um áudio está contido em outro áudio; reconhecer a fala e o falante de um audio que contém fala, etc.

  As operações de requisição de semântica baseiam-se em operações que extraem informação específica do áudio com o objetivo de determinar informação de seu conteúdo. Por exemplo: as propriedades acústico-perceptuais *brightness*, *pitch* e *loudness* são utilizadas para determinar similaridade entre áudios (Wold et al., 1996); a seqüência de diferenças relativas no *pitch* entre notas sucessivas pode ser utilizada para determinar similaridade entre melodias (Ghias et al., 1995); existem diversas técnicas de

processamento de fala que permitem o reconhecimento de fala (Bu e Chiueh, 2000) e a identificação do falante (Sukkor et al., 2000).

Em (Foote, 1999) apresenta-se uma visão geral do estado da arte em recuperação de informação do conteúdo de dados áudio.

- Captura, apresentação e acesso temporal: permitem controlar a captura e apresentação de áudio, assim como determinar o momento de apresentação dentro do áudio (*play, stop, record, stop record, forward, rewind, pause*).
- Modificação de áudio: permite modificar atributos de um áudio. Por exemplo: modificar formato, edição e mudança no volume do áudio.
- Exportação/Importação: permite exportar/importar um áudio para/de um arquivo de diferente formato.

Os dados áudio podem ser modelados utilizando o paradigma de orientação a objeto. Os princípios desse paradigma podem ser utilizados para definir uma superclasse, que represente as características e o comportamento comuns a todos os áudios, e diferentes subclasses, que representem as características e o comportamento de tipos de áudio específicos. Por exemplo, subclasses podem ser criadas para diferenciar arquivos de fala, música e efeitos de sons (Dionisio e Cárdenas, 1998).

A modelagem de dados áudio deve ser independente do domínio das aplicações. O modelo deve-se concentrar no projeto de um serviço amplo e de simples utilização.

3.4 Especificação

AGA está formada por diferentes componentes que projetam as funcionalidades necessárias para gerenciar áudio em um sistema relacional. A seguir, descreve-se sua estrutura e funcionamento.

A Figura 3.2 mostra a estrutura de AGA. Observa-se que a arquitetura permite a associação da aplicação a um SGBDR, onde são armazenados os dados.

AGA está formada por três componentes:

1. Módulo de Definição de Dados (MDD): oferece as funcionalidades para definir um banco de dados.
2. Módulo de Manipulação de Dados (MMD): oferece as funcionalidades para manipular um banco de dados.

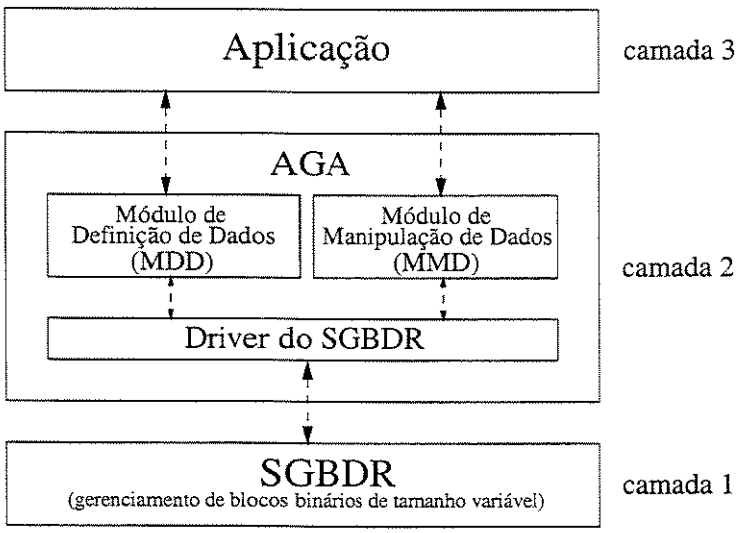

Figura 3.2: AGA - Componentes.

3. Driver do SGBDR: oferece as funcionalidades para estabelecer a relação entre AGA e o SGBDR utilizado pela aplicação.

3.4.1 Componente MDD

O MDD suporta a definição de um banco de dados relacional que integra dados convencionais e áudio. Este componente esboça as facilidades da Linguagem de Definição de Dados (LDD) de um SGBDR e adiciona funcionalidades para definir relações com atributos áudio.

O componente MDD oferece um tipo de dados para representar áudio em uma relação. Isto é, o novo tipo de dados constitui um domínio a ser utilizado na definição do esquema de um banco de dados relacional. Esse novo tipo de dados deve seguir os aspectos de modelagem de áudio apresentados na Seção 3.3.

O componente MDD esboça as operações de definição de um banco relacional, integrando atributos convencionais e áudio. Incluem-se neste componente as operações expostas a seguir.

- Criar uma relação: permite definir uma relação especificando seu nome, atributos, domínios e restrições de integridade. O novo domínio é utilizado para definir um atributo áudio.

- Remover uma relação: permite remover todas as tuplas de uma relação e seu esquema.

- Alterar uma relação: permite renomear uma relação, renomear um atributo de uma relação, adicionar um atributo e remover um atributo.

Uma implementação de AGA deve oferecer um tipo de dados que modele os dados áudio. Além disso, deve definir e desenvolver as operações propostas no componente MDD. Essas operações devem permitir a especificação de nomes de relações, nomes de atributos, domínio de atributos e restrições de integridade.

A implementação de AGA deve definir o conjunto de domínios de atributos suportado. Esse conjunto deve incluir os tipos de dados convencionais de qualquer sistema relacional, além do domínio para dados áudio.

3.4.2 Componente MMD

O componente MMD suporta a manipulação de áudio em junção com os dados tradicionais. Este componente esboça as facilidades da Linguagem de Manipulação de Dados (LMD) de um SGBDR e adiciona as funcionalidades adequadas para manipular um banco de dados relacional que integra áudio de acordo com as especificações do componente MDD.

A seguir são listadas as principais funcionalidades deste componente:

- Atualização: permite realizar operações de atualização definidas sobre relações (inserir uma tupla, remover uma tupla e modificar valores de atributos de uma tupla).
- Consulta: inclui operações de recuperação definidas sobre relações (seleção).
- Manipulação do resultado de uma consulta: permite a obtenção dos valores dos atributos de uma relação resultante de uma consulta. Tipicamente, os dados recuperados devem ser armazenados, visualizados ou processados com determinado propósito.

As operações que removem, modificam e recuperam tuplas devem permitir a especificação de condições sobre os dados (condições de manipulação: a operação é realizada sobre as tuplas que satisfazem as condições). Essas condições devem integrar aquelas sobre os dados convencionais e as baseadas nos dados áudio.

Os critérios para interrogar áudio incluem as operações que requisitam suas características, e aquelas que requisitam sua semântica (ver Seção 3.3).

Uma implementação de AGA deve definir e desenvolver as operações propostas no componente MMD. Essas operações devem permitir a especificação de nomes de relações, nomes de atributos, valores de atributos e condições de manipulação. A implementação de AGA deve definir uma sintaxe para expressar essas condições, de maneira que integre dados convencionais e áudio.

3.4.3 Componente Driver

O Driver permite estabelecer a relação entre AGA e o SGBDR utilizado pela aplicação. Ele oferece aos componentes MDD e MMD suporte para dados convencionais e blocos binários. Quer dizer, ele é responsável pela definição e manipulação desses dados, utilizando o SGBDR indicado pela aplicação.

Dado que nem todos os SGBDRs possuem a mesma linguagem de gerenciamento de dados, a função do Driver é traduzir as requisições dos componentes de definição e manipulação na linguagem do SGBDR, estabelecer a comunicação com o banco de dados e enviar as operações sobre ele. Por último, o Driver deve mapear os resultados dessas operações para entregá-los aos componentes MDD e MMD.

As funcionalidades deste componente são resumidas a seguir.

- Conexão a um banco de dados: permite estabelecer a conexão a um banco de dados.
- Operações de definição: permite criar, remover e alterar relações com atributos convencionais e blocos binários.
- Operações de manipulação: permite atualizar relações com atributos convencionais e blocos binários.
- Encerrar conexão: permite encerrar uma conexão a um banco de dados.

Em princípio, o Driver promove a independência da arquitetura proposta dos SGBDRs, pois ele é o único componente a ser mudado de um SGBDR para outro. Isto é, a introdução do Driver na arquitetura implica que uma mudança de SGBDR não causa mudanças nos outros componentes de AGA.

Uma implementação de AGA deve definir as operações propostas no componente Driver. Essas operações devem ser simples e genéricas, de maneira tal que possam ser facilmente implementadas para qualquer sistema relacional com blocos binários.

3.5 Fluxograma de requisições

O fluxo de uma requisição é mostrado na Figura 3.3. Os passos são detalhados a seguir.

1. A aplicação envia uma requisição ao MDD (ou MMD).
2. O MDD (ou MMD) processa a requisição recebida para determinar as operações que serão resolvidas pelo módulo e as operações que precisará enviar ao Driver do SGBDR.
3. O Driver do SGBDR converte as operações recebidas em requisições na linguagem do SGBDR base e as envia para este.

4. O SGBDR efetua as requisições sobre o banco de dados e envia os resultados ao Driver.

5. O Driver do SGBDR converte os resultados no esquema do MDD (ou MMD) e envia a este.

6. O MDD (ou MMD) envia os resultados finais à aplicação a partir dos resultados das operações resolvidas pelo módulo e pelo Driver.

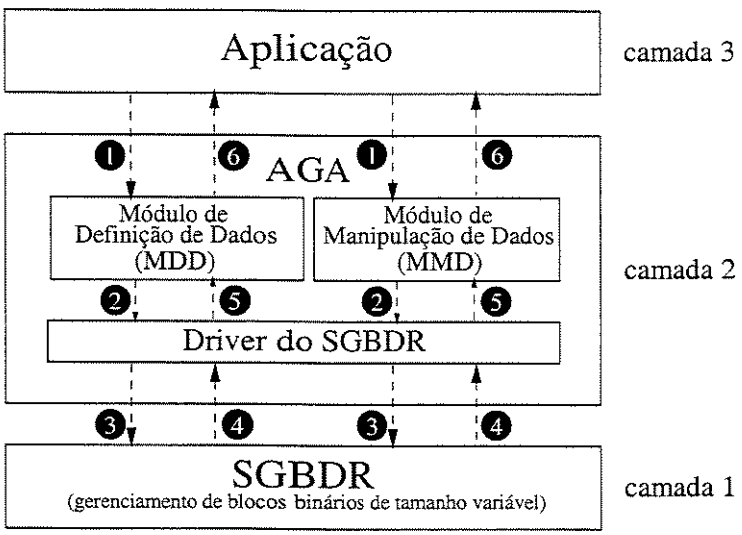

Figura 3.3: AGA - Fluxograma de uma requisição.

Como se pode observar, os componentes MDD e MMD gerenciam os dados convencionais e blocos binários utilizando as operações do Driver, toda vez que for preciso interagir com o SGBDR. O Driver é responsável pela comunicação com o banco de dados e pelo envio das operações que gerenciam os dados (operações que podem ser entendidas pelo SGBDR utilizado).

3.6 Utilização da arquitetura

AGA é utilizada para desenvolver sistemas de gerenciamento de bancos de dados relacionais que integram áudio e dados convencionais.

Uma implementação de AGA deve compreender o desenvolvimento de seus três componentes. O desenvolvimento dos componentes MDD e MMD implica definir e desenvolver

as operações propostas nesses componentes. O desenvolvimento do Driver implica definir as operações propostas nesse componente.

Uma aplicação utiliza uma implementação de AGA. Para isso, é preciso que exista um Driver para o SGBDR utilizado pela aplicação. Quer dizer, as operações do Driver devem ter sido implementadas para esse SGBDR. Adicionalmente, o desenvolvedor da aplicação deve conhecer o modelo de dados relacional e as interfaces dos componentes MDD e MMD.

3.7 Sumário

Este capítulo apresentou AGA, uma arquitetura que permite definir sistemas de gerenciamento de áudio e dados tradicionais. AGA é definida sobre os sistemas relacionais com suporte de blocos binários, permitindo aproveitar suas capacidades para gerenciar dados tradicionais e para armazenar dados áudio.

AGA promove um modelo de dados áudio genérico, para oferecer um tipo de dados que represente áudio nas relações. AGA propõe interfaces para definir e manipular relações que integram os domínios tradicionais e o novo domínio para áudio.

A arquitetura proposta é de propósito geral e flexível, pois engloba componentes que projetam funcionalidades independentes do domínio das aplicações e de um SGBDR específico. Dessa maneira, oferece uma plataforma comum para o desenvolvimento de aplicações que gerenciam áudio.

O próximo capítulo apresenta uma implementação simples da arquitetura proposta.

Capítulo 4

Uma implementação de AGA

Neste capítulo descreve-se SGA (Sistema de Gerenciamento de Áudio), uma implementação simples de AGA. O sistema é uma ferramenta que implementa algumas das funcionalidades da arquitetura proposta. A implementação de SGA compreendeu o desenvolvimento dos três componentes da arquitetura, guiado pelos princípios de projeto de AGA mais os novos princípios que foram definidos especificamente para a implementação.

4.1 Princípios de projeto

Vários princípios de projeto foram definidos para o desenvolvimento de SGA, resumidos a seguir.

- Ser independente de plataforma.
- Permitir o gerenciamento de bancos de dados relacionais que integram áudio na Web.
- Ser extensível e reutilizável.
- Ser simples de utilizar.

No desenvolvimento de SGA, de forma a atender os princípios de projeto listados acima, escolheu-se Java (Naughton, 1996; Deitel e Deitel, 1998) como linguagem de implementação. Java garante a independência de plataforma e possibilita o gerenciamento de bancos de dados relacionais na Web. Além disso, Java permite a extensibilidade e reusabilidade do sistema pelo seu paradigma de orientação a objeto.

4.2 Descrição do sistema

O desenvolvimento de SGA incluiu os itens expostos a seguir.

- Definição e implementação de um tipo de dados para representar áudio: modela as característica e o comportamento básicos de um áudio digital.
- Definição de uma interface que especifica as funcionalidades do componente Driver do SGBDR: especifica as funcionalidades a serem desenvolvidas por um Driver de um SGBDR específico.
- Definição e implementação de uma interface para a comunicação com um banco de dados: permite à aplicação solicitar/encerrar uma conexão a um banco de dados.
- Definição e implementação de uma interface que oferece as funcionalidades do componente MDD: permite definir um banco de dados relacional que integra áudio; oferece um novo domínio para representar áudio nas relações.
- Definição e implementação de uma interface que oferece as funcionalidades do componente MMD: permite manipular um banco de dados que integra áudio, assim como manipular o resultado das consultas.

As diferentes interfaces mencionadas nos itens listados acima foram desenvolvidas como um conjunto de classes. As seções seguintes oferecem detalhes dessas interfaces.

4.2.1 Tipo de dados para representar áudio

SGA define e implementa um tipo de dados para representar áudio através da classe `Audio`. Essa classe modela as características e o comportamento dessa mídia. A modelagem de áudio considerou os aspectos destacados na Seção 3.3.

A classe `Audio` define um conjunto de propriedades que descrevem suas características de formato, duração e volume, tais como:

- `contentType`: tipo de conteúdo (WAV, MP3, AU, etc.);
- `encoding`: método de codificação (Linear, ULAW, ALAW, MPEGLAYER3, etc.);
- `sampleSize`: tamanho de amostra (8 bits, 16 bits, etc.);
- `samplingRate`: freqüência de amostragem (8 kHz, 44.1 kHz, etc.);
- `channels`: número de canais (1, 2, etc.); e
- `volume` : volume do áudio.

A classe `Audio` define um conjunto de métodos que permitem manipular áudio, tais como:

- `getContentType`: obtém o tipo de conteúdo;
- `getEncoding`: obtém o método de codificação;
- `getDuration`: obtém a duração;
- `getVolume` e `setvolume`: obtém e modifica o volume, respectivamente;
- `play`, `stop`, `pause`, `forward` e `rewind`: apresenta, controla apresentação e permite acesso temporal; e
- `saveAs`: salva o arquivo com um novo formato.

Cada um desses métodos possui parâmetros que permitem passar a informação necessária para realizar as diferentes operações.

SGA modela um conjunto de eventos, que podem ser gerados pelos objetos da classe `Audio`, através da classe `AudioEvent` e suas subclasses `AudioStartEvent`, `AudioStopEvent`, `AudioDeviceUnavailableEvent`, `AudioVolumeSetEvent`, `AudioMediaTimeSetEvent` e `AudioErrorEvent`. Esses eventos notificam assincronamente mudanças no estado de um áudio. Os eventos podem notificar as informações resumidas a seguir.

- Apresentação/Fim de apresentação do áudio: inicio da apresentação do áudio (`AudioStartEvent`); fim da apresentação do áudio, porque foi solicitado explicitamente ou porque chegou o fim da mídia (`AudioStopEvent`).
- Recurso não disponível: o áudio não consegue obter o dispositivo de apresentação (`AudioDeviceUnavailableEvent`).
- Mudança no volume do áudio: o volume do áudio foi modificado utilizando o método `Audio.setVolume` (`AudioVolumeSetEvent`).
- Mudança no tempo da mídia: o áudio foi acessado temporalmente utilizando os métodos `Audio.forward` e `Audio.rewind`, mudando o momento a partir do qual será apresentado o áudio (`AudioMediaTimeSetEvent`).
- Erros nas operações sobre o áudio: sob ocorrência de funcionamento irregular, é informado que o objeto áudio é encerrado e não poderá ser referenciado novamente (`AudioErrorEvent`).

SGA define a interface Java `AudioListener` que permite responder aos eventos gerados por objetos da classe `Audio`. A interface define o método `audioUpdate` com essa finalidade.

Para os usuários da classe `Audio` receberem os eventos, eles devem se registrar como ouvintes desses eventos e implementar a interface `AudioListener`. A classe `Audio` define os

métodos `addAudioListener` e `removeAudioListener` para o registro e remoção de ouvintes dos eventos, respectivamente. Todos os eventos gerados por um objeto `Audio` são enviados a cada um dos ouvintes registrados.

A Figura 4.1 mostra as relações entre as classes `Audio`, `AudioEvent` e suas diferentes subclasses, e a interface `AudioListener`.

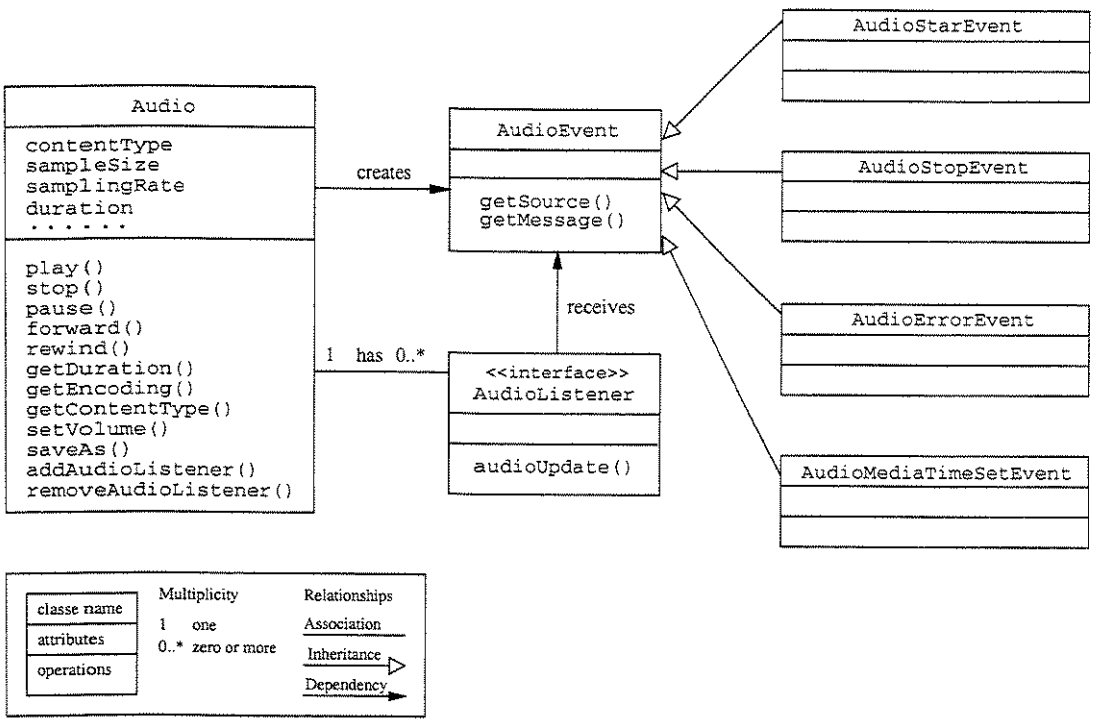

Figura 4.1: SGA - Relações entre as classes que permitem manipular áudio.

Os métodos `getSource()` e `getMessage()` da classe `AudioEvent` permitem obter o objeto que gerou o evento e a mensagem associada a esse evento.

Utilizou-se a API JMF (*Java Media Framework*) para desenvolver as funcionalidades relacionadas à classe `Audio`.

4.2.2 Interface do componente Driver

O componente Driver de AGA permite estabelecer a relação entre a arquitetura e o SGBDR da aplicação. Ele possibilita estabelecer e encerrar a comunicação com o banco de dados, e atende as requisições referentes a dados convencionais e blocos binários dos componentes MDD e MMD.

SGA especifica as funcionalidades de um Driver do SGBDR através da interface Java `AMSDriver` (acrônimo de *Audio Management System Driver*). Essa interface declara as operações para estabeler e encerrar conexões a bancos de dados e para gerenciá-los.

Em Java, uma interface é uma coleção de métodos sem suas implementações, que define um protocolo de comportamento. Uma classe que implementa uma interface adere-se a seu protocolo.

A interface `AMSDriver` define os métodos resumidos a seguir.

- `connect()` / `close()`: estabelece/encerra uma conexão a um banco de dados;
- `createTable()` / `dropTable()` / `addColumn()` / `renameTable()` / `renameColumn()`: cria / remove / altera relações;
- `insertRow()` / `deleteRow()` / `updateRow()` / `selectRow()`: insere uma tupla / remove tuplas / modifica atributos de tuplas / recupera tuplas; e
- `importBlob()` / `exportBlob()` / `removeBlob()`: importa / exporta / remove um objeto binário.

A interface `AMSDriver` especifica as funcionalidades que deve oferecer qualquer Driver de SGA. Ela garante a independência dos SGBDRs. Essas funcionalidades são simples, e podem ser implementadas para qualquer SGBDR com suporte de blocos binários. O desenvolvedor de um Driver para um SGBDR específico deve desenvolver uma classe que implemente as operações definidas na interface.

A interface do Driver pode ser enriquecida com operações de gerenciamento de dados convencionais e blocos binários mais complexas tais como recuperação de informação de várias relações.

As operações do Driver são chamadas pela interface definida para a comunicação com um banco de dados e pelas interfaces de definição e manipulação de SGA, que serão explicadas nas próximas seções.

4.2.3 Conexão a um banco de dados

Quando uma aplicação vai gerenciar dados através de SGA, primeiro precisa obter uma conexão ao banco de dados a utilizar. A aplicação deve especificar esse banco de dados, assim como o Driver do SGBDR (adequado ao banco de dados).

De maneira a permitir o gerenciamento de bancos de dados relacionais na Web, SGA estabelece a identificação de um banco de dados através de um URL. Ele inclui o identificador do Driver do SGBDR, assim como o nome e endereço na Internet (se for remoto) do

banco de dados. O identificador do Driver é determinado pelo desenvolvedor do mesmo. O apêndice A mostra a sintaxe do URL de um banco de dados em SGA.

SGA define a classe `AMSDriverManager` (acrônimo de *Audio Management System Driver Manager*) que constitui uma interface intermediária entre a aplicação e o Driver utilizado. Ela permite estabelecer a conexão ao banco de dados através do Driver especificado. O método `AMSDriverManager.getConnection` deve ser utilizado para solicitar uma conexão ao banco de dados. SGA representa uma conexão através da classe `MyConnection`.

A Figura 4.2 mostra as relações entre `AMSDriver`, `AMSDriverManager` e `MyConnection`.

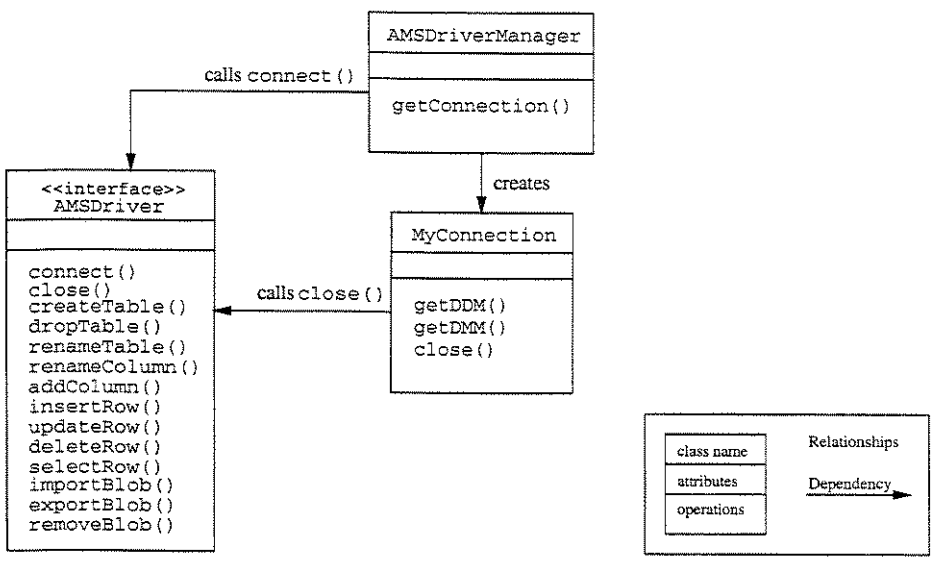

Figura 4.2: SGA - Relações entre as classes envolvidas na comunicação com um banco de dados.

`AMSDriverManager` estabelece a conexão ao banco de dados utilizando o método `connect()` do Driver do SGBDR especificado. A conexão é representada por um objeto da classe `MyConecction`. Ela pode ser encerrada utilizando seu método `close()` que finaliza a comunicação utilizando o método `close()` do Driver.

Cada conexão tem uma interface de definição e uma interface de manipulação associadas. Elas podem ser utilizadas para gerenciar os dados. Uma sessão de conexão permite o acesso a essas interfaces através dos métodos `getDDM` e `getDMM`, respectivamente.

Uma aplicação pode ter uma ou mais conexões a um banco de dados, assim como pode ter diferentes conexões a diferentes bancos de dados.

Exemplo: comunicação com um banco de dados

```
. . . . . .
String dbURL = "ams:pgsqlams:musicdb";
String driverName = "pgsqlams.AMSDriver";

//get connection to the database
MyConnection conn;
conn = AMSDriverManager.getConnection(dbURL, driverName, user, password);

//get data definition interface (DDM object)
DDM ddm = conn.getDDM();

//get data manipulation interface (DMM object)
DMM dmm = conn.getDMM();
. . . . . .
//close connection
conn.close();
. . . . . .
```

No exemplo, solicita-se uma conexão ao banco de dados local `musicdb` e o Driver utilizado é `pgsqlams.AMSDriver` (identifica um Driver implementado para PostgreSQL). O nome do usuário do banco de dados e seu *password* devem ser especificados.

4.2.4 Definição de um banco de dados

O componente MDD de AGA esboça as funcionalidades que permitem definir um banco de dados que integra atributos convencionais e áudio.

SGA define e implementa as funcionalidades do componente MDD através da classe `DDM` (acrônimo de *Data Definition Module*). Essa classe oferece diferentes métodos para criar, remover e alterar esquemas de relações com atributos convencionais e áudio.

Atributos convencionais

SGA definiu um conjunto de domínios convencionais a serem suportados nesta primeira versão de implementação, apresentados no apêndice B. O usuário da classe `DDM` pode utilizar esses domínios para representar informação tradicional em um banco de dados.

Atributos áudio

SGA permite a integração de áudio. Para isso, foi definido o domínio `Audio`, o que possibilita representar áudio nas relações. As características e o comportamento dos valores

que formam o novo domínio são modelados pela classe `Audio`.

Operações de definição

Nesta primeira versão de implementação, `DDM` inclui os métodos resumidos a seguir.

- `createTable()`: cria uma relação;
- `dropTable()`: remove as tuplas de uma relação e seu esquema;
- `addColumn()`: altera o esquema de uma relação, adicionando uma nova coluna;
- `renameColumn()`: altera o esquema de uma relação, renomeando uma coluna; e
- `renameTable()`: renomeia uma relação.

Esses métodos possuem um conjunto de parâmetros que permitem passar a informação necessária para realizar as diferentes operações, tais como nome de relação, nomes e domínios de atributos, e restrições de integridade.

A Figura 4.3 mostra as relações entre as classes envolvidas na definição de um banco de dados.

Uma conexão `MyConnection` permite obter a interface de definição `DDM` associada através do método `getDDM()`. Essa interface pode ser utilizada para definir um banco de dados que integra áudio. As funcionalidades oferecidas pela interface são implementadas utilizando as funcionalidades do Driver.

As definições convencionais são resolvidas facilmente chamando as operações do Driver. A definição de um atributo de domínio `Audio` é suportada por estruturas que envolvem definições convencionais e de blocos binários.

SGA cria uma tabela auxiliar com o objetivo de suportar a definição e manipulação de atributos de domínio `Audio`. Essa tabela armazena os dados áudio como blocos binários, além de informação sobre eles (metadados), incluindo atributos do formato e duração. As funcionalidades do Driver são utilizadas para definir e manipular essa tabela.

Exemplo: definição de uma relação

```
. . . . . .
//get connection to the database
MyConnection conn;
conn = AMSDriverManager.getConnection(dbURL, driverName, user, password);

//get data definition interface (DDM object)
DDM ddm = conn.getDDM();
```

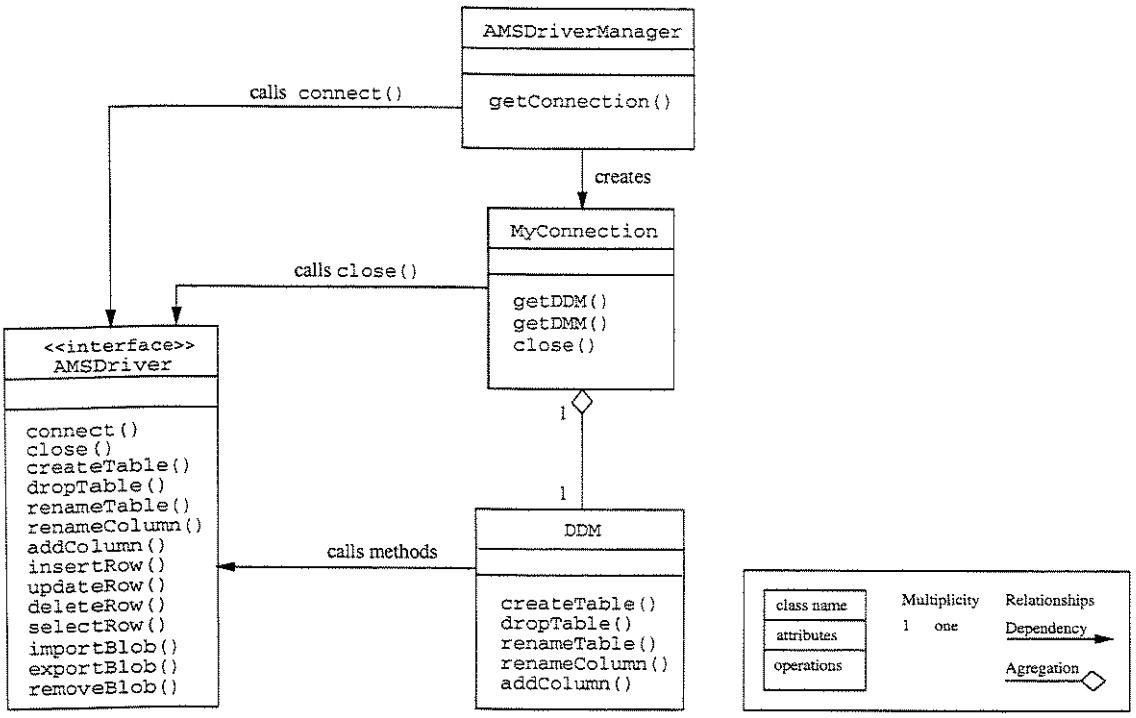

Figura 4.3: SGA - Relações entre as classes envolvidas na definição de um banco de dados.

```
//create table "songs"
Columns columns = new Columns();
Column column;
column = new Column("id", Types.INTEGER);
columns.addColumn(column);
column = new Column("songtitle", Types.TEXT);
columns.add(column);
column = new Column("audiodata", Types.AUDIO);
columns.addColumn(column);
String primaryKey = "id";
String tableName = "songs";
ddm.createTable(tableName, columns, primaryKey);
. . . . . .
//close connection
conn.close();
. . . . . .
```

No exemplo, utiliza-se o método `createTable` do objeto da classe `DDM`, associado à

conexão, para criar a relação `songs`. Ela é criada com dois atributos convencioanais `id` e `songtitle`, e o atributo `audiodata` de domínio `Audio`. Além disso, especifica-se que o atributo `id` é a chave primária da relação.

As classes `Column` e `Columns` foram criadas para permitir a representação de uma coluna e de várias colunas, respectivamente. A classe `Types` define os domínios suportados por SGA, incluindo o domínio `Audio` (`Types.AUDIO`).

4.2.5 Manipulação de um banco de dados

O componente MMD de AGA esboça as funcionalidades que permitem manipular um banco de dados que integra atributos convencionais e áudio.

SGA define e implementa as funcionalidades do componente MMD através da classe `DMM` (acrônimo de *Data Manipulation Module*). Essa classe oferece um conjunto de métodos que permitem atualizar e consultar um banco de dados relacional que integra áudio.

Operações de manipulação

Nesta primeira versão de implementação, a classe `DMM` inclui os métodos resumidos a seguir.

- `insertRow`: insere uma tupla;
- `deleteRow`: remove uma ou várias tuplas;
- `updateRow`: modifica atributos de uma ou várias tuplas; e
- `selectRow`: recupera tuplas.

Esses métodos possuem parâmetros que facilitam passar a informação necessária para realizar as operações de manipulação, tais como nome de relação, nomes e valores de atributos, e condições de manipulação.

Os métodos definidos na classe `DMM` permitem especificar condições sobre os dados. Somente as tuplas que satisfazem as condições especificadas são atualizadas ou recuperadas.

Esta primeira versão de SGA define uma sintaxe simples para formular essas condições, permitindo a integração de condições sobre atributos tradicionais e baseadas em atributos de domínio `Audio`. O apêndice C mostra a sintaxe definida.

As condições baseadas em áudio devem formar-se a partir dos nomes dos métodos definidos na classe `Audio` que interrogam características do áudio e dos métodos que interrogam aspectos semânticos desses dados. Esta primeira versão de SGA não implementou métodos que interroguem o conteúdo semântico de áudio.

Manipulação baseada em áudio

A tabela auxiliar de SGA contém informação que o sistema utiliza para resolver as requisições baseadas em áudio. Essa tabela é povoada da maneira explicada a seguir.

Quando um usuário insere um áudio em uma relação, o dado não é inserido propriamento nessa relação. Em lugar disso, SGA cria um *handle* para representar o áudio, que é armazenado na tabela do usuário. SGA armazena o áudio como tal na tabela auxiliar, junto com as características que descrevem seu formato e duração. Essa tabela também armazena o *handle* do áudio, permitindo a relação com a informação da tabela do usuário.

A Figura 4.4 mostra com um exemplo a tabela auxiliar de SGA.

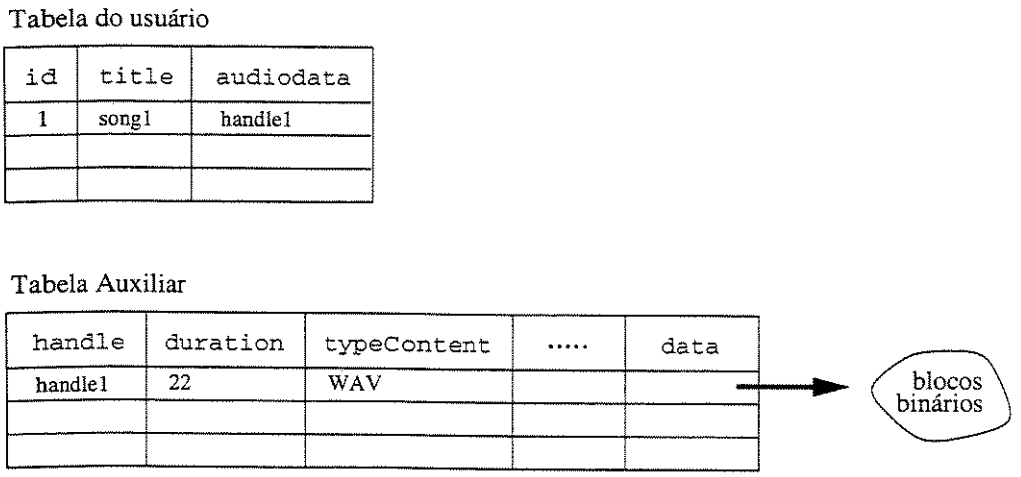

Figura 4.4: SGA - Tabela Auxiliar.

Um usuário insere um áudio indicando o arquivo que o contém. SGA obtém as informações que o descrevem criando um objeto `Audio` com o arquivo indicado. O áudio é processado apenas uma vez, quando é inserido no banco de dados.

À medida que novos métodos que extraiam informação semântica de áudio estejam disponíveis na classe `Audio`, as capacidades de SGA para a recuperação baseada no conteúdo de áudio serão estendidas.

Resultado das consultas

O resultado de uma consulta a um banco de dados é um conjunto de tuplas. SGA proporciona facilidades para manipular essa informação através da classe `MyResultSetManager`. Isto é, um objeto dessa classe contém as tuplas recuperadas pelo método `DMM.selectRow`.

A classe `MyResultSetManager` permite o acesso aos dados recuperados. Ela define um conjunto de métodos `getXXX` que pode ser utilizado para obter os valores dos atributos de cada tupla recuperada.

Por exemplo: o método `getInt` permite recuperar o valor de um atributo como um número inteiro; o método `getString` serve para recuperar um valor como uma cadeia de caracteres; etc. Na hora de acessar os resultados, é preciso indicar a tupla e o nome ou número do atributo cujo valor deseja-se recuperar.

O método `getAudio` deve ser utilizado para recuperar um valor de um atributo de domínio `Audio`. Uma vez obtido um objeto `Audio`, ele pode ser apresentado, acessado temporalmente, modificado, e suas características podem ser recuperadas, utilizando os métodos definidos nessa classe. Os eventos do objeto podem ser manipulados pelos ouvintes registrados.

A classe `MyResultSetManager` também contém os métodos `first` e `next` para o deslocamento entre as tuplas recuperadas.

SGA define e implementa a classe `MyResultSetMetaData` que contém um conjunto de métodos que fornece informação sobre os dados recuperados. Esse conjunto de métodos inclui `getRowCount` (obtém o número de tuplas recuperadas), `getColumnCount` (obtém o número de atributos recuperados), `getColumnName` (obtém o nome de um atributo recuperado) e `getColumnTypeName` (obtém o domínio de um atributo recuperado).

A Figura 4.5 mostra as relações entre as classes envolvidas na manipulação de um banco de dados.

Uma conexão `MyConnection` permite obter a interface de manipulação `DMM` associada através do método `getDDM()`. Essa interface pode ser utilizada para manipular um banco de dados que integra áudio.

Uma consulta ao banco de dados (método `DMM.selectRow()`) cria um objeto da classe `MyResultSetManager`. Os valores das tuplas recuperadas podem ser acessados utilizando os métodos dessa classe. Objetos da classe `Audio` podem ser recuperados com o método `getAudio`, e seus eventos `AudioEvent` podem ser manipulados através da interface `AudioListener`.

Cada objeto `MyResultSetManager` tem associado um objeto da classe `MyResultSetMetaData`, que permite obter informação dos dados recuperados.

As funcionalidades oferecidas pela interface da classe `DMM` são implementadas utilizando as funcionalidades do Driver.

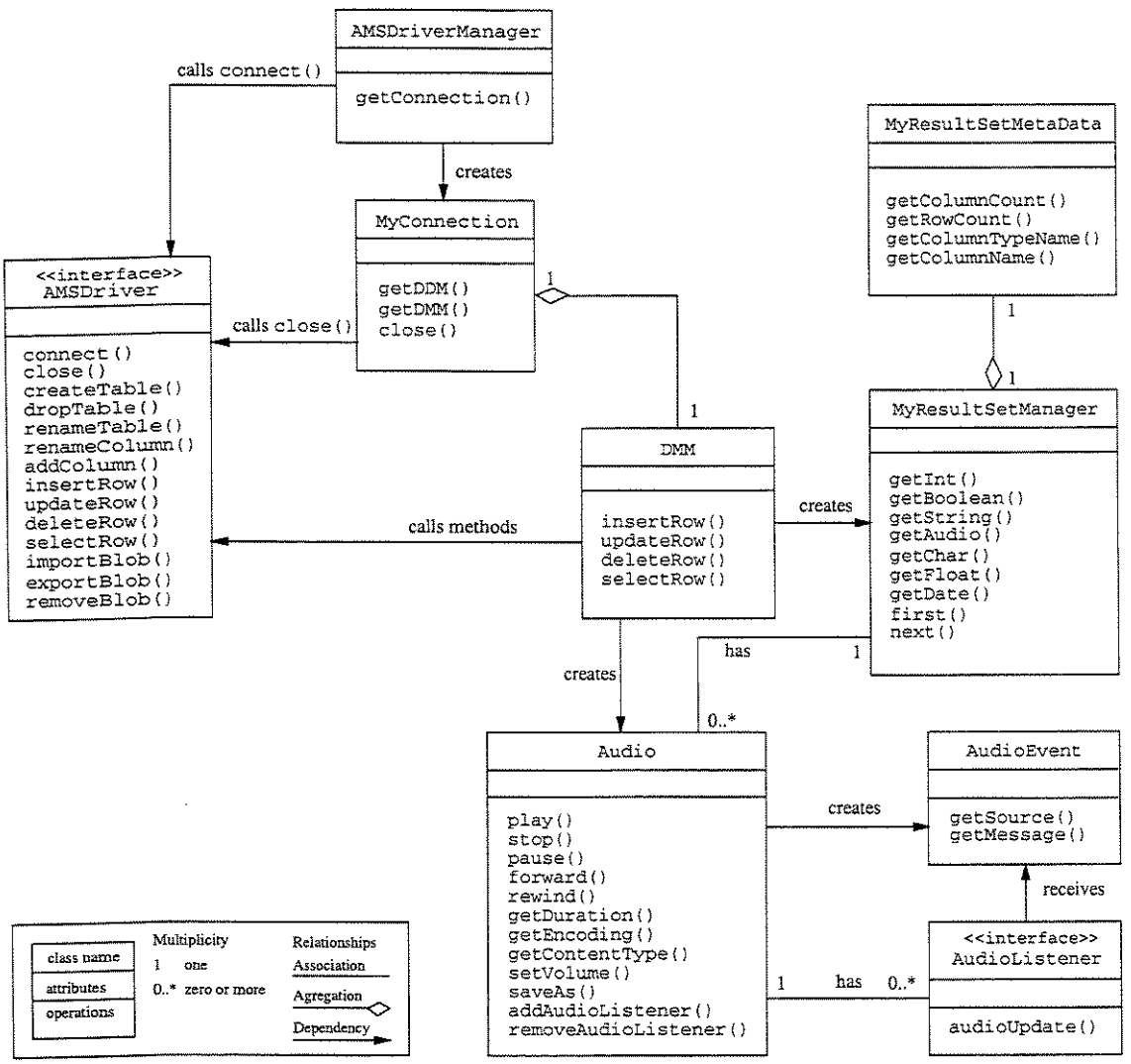

Figura 4.5: SGA - Relações entre as classes envolvidas na manipulação de um banco de dados.

Exemplo: inserção de uma tupla e consulta

```
. . . . . .
//get connection to the database
MyConnection conn;
conn = AMSDriverManager.getConnection(dbURL, driverName, user, password);

//get data manipulation interface (DMM object)
DMM dmm = conn.getDMM();

//insert a row
ColumnValue cv;
RowValues rowVal = new RowValues();
cv = new ColumnValue("id", 1);
rowVal.addColumnValue(cv);
cv = new ColumnValue("audiodata", "~/audios/song1.wav");
rowVal.addColumnValue(cv);
cv = new ColumnValue("songtitle", "Calice");
rowVal.addColumnValue(cv);
String  tableName = "songs";
dml.insertRow(tableName, rowVal);

//retrieve the row
String columns [] = new String[1];
columns[0] = "audiodata";
String whereCondition;
whereCondition = "songtitle='Calice' and getEncoding(audiodata)='ulaw';
MyResultSetManager myRSM = dmm.selectRow(columns, tableName, whereCondition);

//get retrieved audio to play
Audio au = myRSM.getAudio(1);
if ( myRSM.next() )
  au.play();
. . . . . .
//close connection
conn.close();
. . . . . .
```

No exemplo, primeiramente é inserida uma tupla na relação songs. O áudio é indicado pelo nome e localização do arquivo. As classes `ColumnValue` e `RowValues` permitem representar o valor de um atributo e os valores dos atributos de uma tupla, respectivamente.

Em seguida é feita uma consulta ao banco de dados. Especifica-se o atributo a recuperar

`audiodata` e as condições que devem satisfazer as tuplas recuperadas. O exemplo integra uma condição sobre um atributo convencional (`songtitle='Calice'`) e uma condição sobre um atributo de domínio `Audio` (`getEncoding(audiodata)='ulaw'`). Note que a condição sobre áudio utiliza o método `Audio.getEncoding()` que interroga o método de codificação de um áudio.

A consulta feita permitiu recuperar uma tupla. O objeto `Audio` dessa tupla é acessado utilizando o método `MyResultSetManager.getAudio()`. Posteriormente, ele é apresentado utilizando `Audio.play()`.

4.3 Modo de utilização do sistema

SGA é um sistema para desenvolvedores de aplicações que gerenciam áudio em junção com dados convencionais. O sistema é uma ferramenta de baixo nível e é projetado para ser a base de interfaces e ferramentas de gerenciamento de áudio amigáveis, orientadas a usuários finais.

A Figura 4.6 apresenta um diagrama que mostra como SGA pode ser utilizado.

Uma aplicação pode utilizar o sistema da maneira explicada a seguir.

1. Solicitar uma conexão ao banco de dados (especificar o URL do banco de dados).
2. Obter interfaces de definição e manipulação de dados.
3. Utilizar as operações das interfaces para o gerenciamento dos dados.
4. Manipular resultados das operações.
5. Encerrar conexão.

Note-se que uma aplicação pode utilizar SGA para gerenciar um banco de dados desde que exista um Driver implementado para o SGBDR utilizado.

4.4 Extensões ao sistema

SGA é uma versão simplificada da arquitetura proposta. O sistema pode ser estendido, adicionando novas funcionalidades.

O tipo de dados áudio proposto por SGA pode ser estendido com o objetivo de representar a estrutura e comportamento específicos de diferentes tipos de áudio, por exemplo:

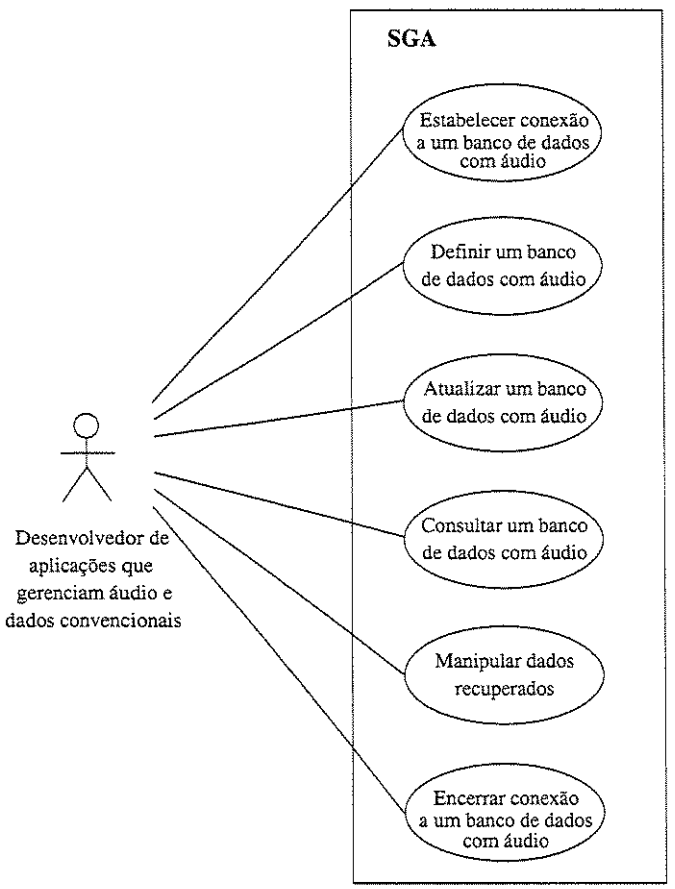

Figura 4.6: SGA - Diagrama de utilização.

música e fala. Para isso, é preciso estender a classe `Audio`. Métodos que extraiam informação semântica dos dados áudio devem ser incorporados para facilitar a busca por conteúdo dessa mídia.

As interfaces de definição e manipulação de SGA podem ser enriquecidas com novas operações para o gerenciamento de dados. Por exemplo, permitir a definição de outras restrições de integridade e permitir a recuperação de dados de várias tabelas.

4.5 Sumário

Este capítulo apresentou SGA, uma implementação simples de AGA. O sistema implementado é uma ferramenta de propósito geral para o desenvolvimento de aplicações que gerenciam áudio. Ele permite a integração de dados convencionais e áudio em sistemas relacionais e é independente de um SGBDR específico.

SGA foi implementado em Java com o objetivo de oferecer um sistema independente de plataforma, permitir o gerenciamento de bancos de dados relacionais na Web e possibilitar a sua extensibilidade e reusabilidade.

O próximo capítulo apresenta duas aplicações protótipo que utilizam SGA.

Capítulo 5

Aplicações protótipo

Este capítulo apresenta duas aplicações protótipo: Interface Gráfica para SGA e Sistema de Recuperação de Músicas. Ambas as aplicações foram desenvolvidas utilizando SGA, com o objetivo de validar o sistema. Foi necessário implementar um Driver para um SGBDR específico. A Seção 5.1 descreve o Driver implementado. As Seções 5.2 e 5.3 apresentam detalhes das aplicações.

5.1 Implementação de um Driver

Para uma aplicação utilizar SGA, precisa existir um Driver implementado para o sistema de gerenciamento correspondente a seu banco de dados. A aplicação Sistema de Recuperação de Músicas utiliza um banco de dados PostgreSQL. Por conseguinte, foi necessário desenvolver um Driver para esse sistema.

O SGBDR PostgreSQL (The PostgreSQL Development Group, 1998; Momjian, 2000) é um sistema relacional-objeto (extensível), com arquitetura cliente-servidor. O acesso a um banco de dados é através de consultas SQL. O sistema oferece suporte para blocos binários, permitindo armazenar informação de tamanho grande e variável.

Desenvolveu-se a classe `PgSQL_AMSDriver` que implementa as operações do Driver de SGA. Isto é, essa classe implementa todas as operações definidas na interface `AMSDriver` para bancos de dados PostgreSQL. Logo, ela oferece as funcionalidades para estabelecer/encerrar comunicação com um banco de dados PostgreSQL, assim como gerenciar seus dados convencionais e blocos binários. A Figura 5.1 mostra como se relaciona a classe com SGA.

O desenvolvimento do Driver foi muito simples. Não foram utilizadas as funcionalidades que fazem extensível o sistema PostgreSQL. Exploraram-se apenas suas capacidades para gerenciar dados convencionais e objetos binários. A interface JDBC de PostgreSQL foi

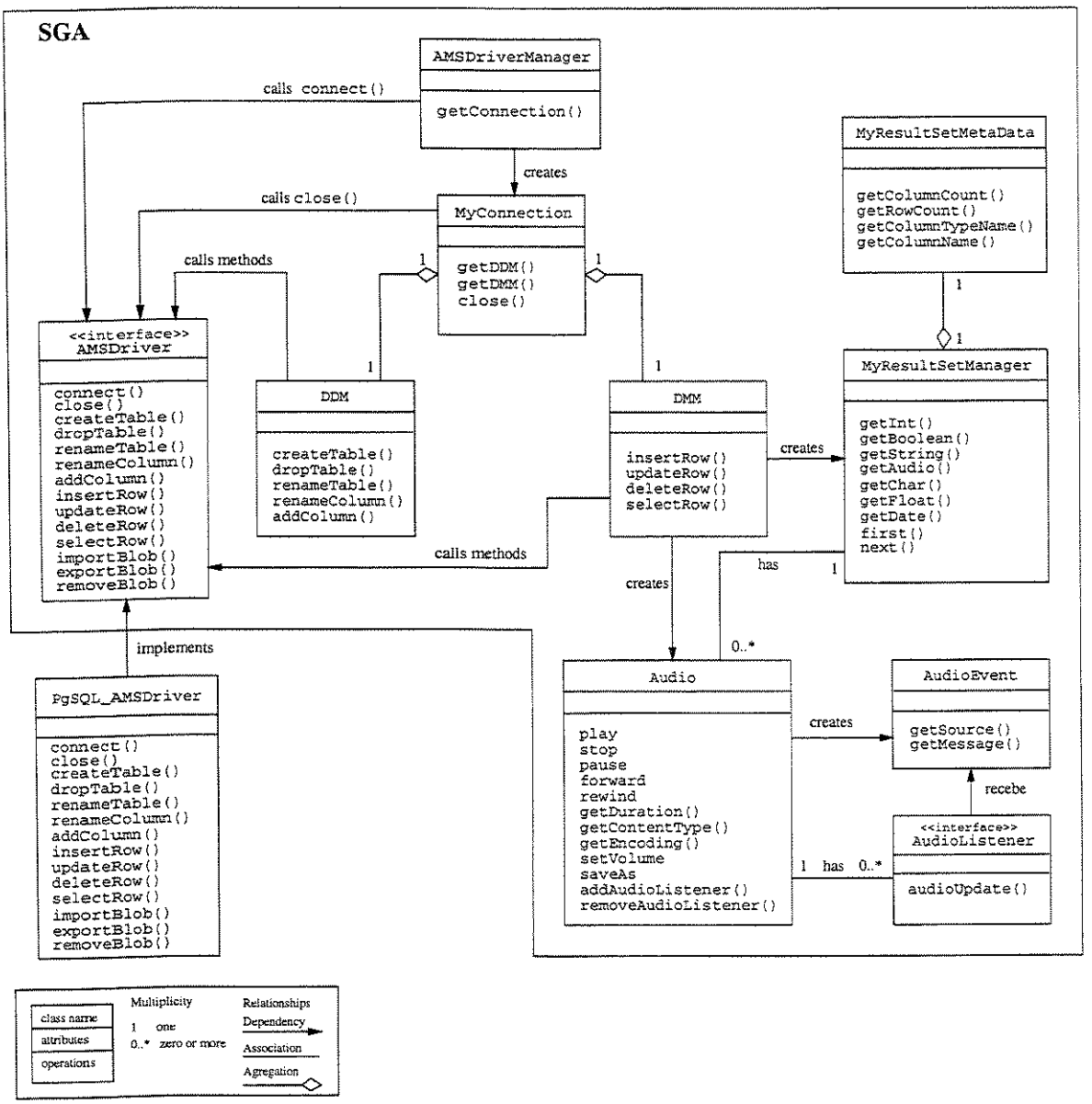

Figura 5.1: Relação entre SGA e o Driver desenvolvido.

utilizada como via de acesso a um banco de dados.

5.2 Interface Gráfica de SGA

A primeira aplicação desenvolvida constitui uma ferramenta visual para SGA. Essa ferramenta possibilita, de maneira gráfica, definir, povoar, atualizar e consultar um banco de dados que integra áudio. O usuário da aplicação indica o URL do banco de dados a gerenciar no chamado à aplicação.

A interface principal desta aplicação apresenta uma barra de menu, como ilustrado na Figura 5.2.

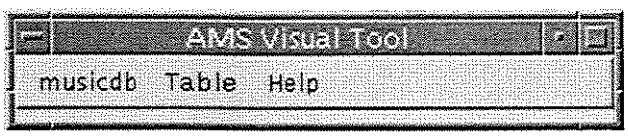

Figura 5.2: Interface Gráfica de SGA - Interface principal.

A seguir são explicados os diferentes menus incluídos.

- Menu Banco de Dados (`musicdb`): apresenta um conjunto de opções que permitem realizar operações de definição do banco de dados indicado pelo usuário, segundo mostra a Figura 5.3.

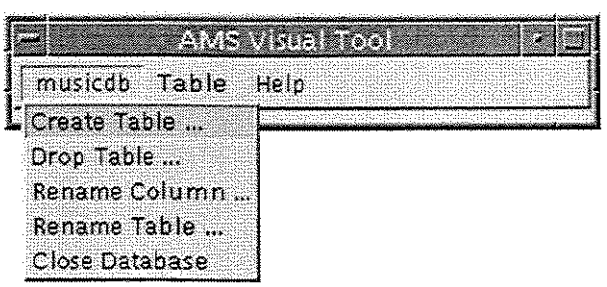

Figura 5.3: Interface Gráfica de SGA - Menu Banco de Dados.

Essas operações incluem criar, remover e alterar relações do banco de dados, assim como fechar a conexão ao banco de dados e terminar a aplicação. Cada uma das operações tem associada uma janela onde o usuário especifica as informações necessárias para realizar a operação.

No exemplo mostrado na Figura 5.3, o nome do banco de dados é `musicdb`.

A Figura 5.4 mostra a interface para criar uma relação no banco de dados. O usuário indica o nome da relação, os atributos e seus domínios, e a chave primária da relação.

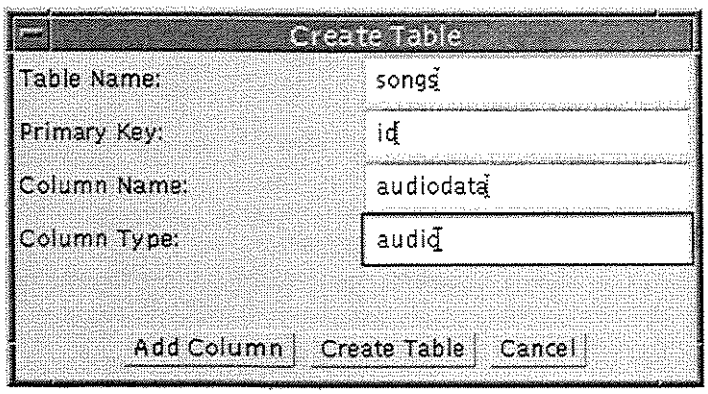

Figura 5.4: Interface Gráfica de SGA - Interface para criar uma tabela.

- Menu Tabela (*Table*): apresenta um conjunto de opções que permitem realizar operações de manipulação de dados, segundo mostra a Figura 5.5.

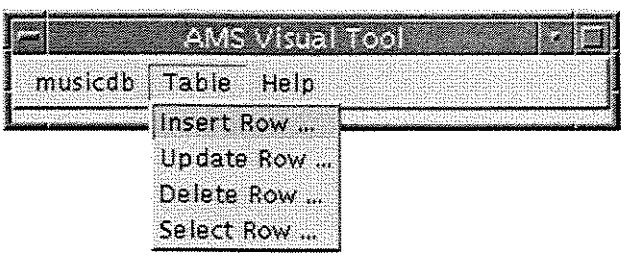

Figura 5.5: Interface Gráfica de SGA - Menu Tabela.

Essas operações incluem inserir, remover, modificar e recuperar tuplas. Cada uma das operações tem associada uma janela onde o usuário especifica as informações necessárias.

A Figura 5.6 mostra a interface que possibilita consultar o banco de dados. Condições sobre atributos convencionais e atributos áudio podem ser integradas. No exemplo destacado, o usuário coloca uma condição convencional (`author = 'Gilberto Gil'`) e uma condição sobre áudio (`getDuration(audiodata) > 200`).

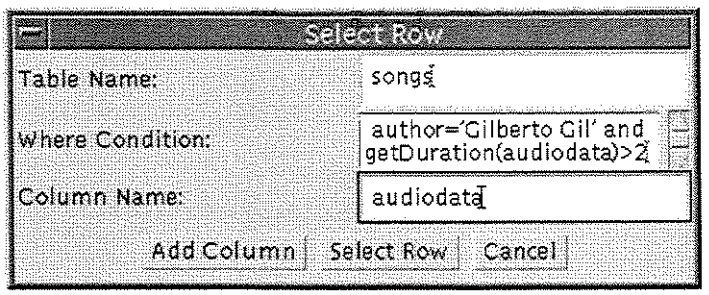

Figura 5.6: Interface Gráfica de SGA - Interface para consultar.

Na Figura 5.7 apresenta-se a interface para examinar os resultados da consulta. Eles são mostrados uma vez que o usuário seleciona a tupla recuperada.

Quando os dados recuperados por uma consulta incluem áudio, habilita-se um conjunto de componentes que proporciona facilidades para apresentá-los e acessá-los temporalmente.

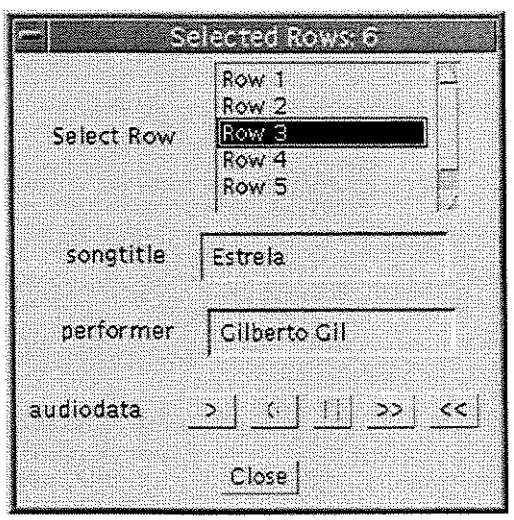

Figura 5.7: Interface Gráfica de SGA - Interface para examinar dados recuperados.

- Menu Ajuda: permite obter informação sobre a aplicação, assim como orientações para utilizar a interface adequadamente.

Esta ferramenta pode ser utilizada com qualquer banco de dados e o Driver adequado. Ela está orientada a usuários que conhecem SGA, pois eles precisam saber os domínios

suportados e a sintaxe para definir condições de manipulação no sistema.

5.3 Sistema de Recuperação de Músicas

A segunda aplicação desenvolvida consiste em uma interface gráfica para a recuperação de músicas e informações associadas a elas. O usuário da aplicação pode consultar um banco de dados que contém músicas, escolhendo os atributos que deseja recuperar e formando condições sobre atributos convencionais e áudio.

Banco de Dados

Definiu-se um banco de dados PostgreSQL de estrutura simples, que armazena músicas (atributos associados a uma música e o dado áudio). O banco de dados está formado por uma única relação cuja estrutura aparece na Tabela 5.1. A interface gráfica de SGA foi utilizada para definir e povoar o banco de dados.

Atributo	Domínio	Descrição
songtitle	text	Título da música
author	text	Nome do compositor
performer	text	Nome do cantor
albumtitle	text	Título do álbum ao qual pertence a música
audiodata	audio	Dado áudio

Tabela 5.1: Relação `songs` - Banco de Dados `musicdb`

Interfaces de usuário

A aplicação de recuperação de músicas apresenta uma interface gráfica tal como ilustrado na Figura 5.8. Na parte superior da tela é apresentado um conjunto de nomes de atributos que o usuário pode selecionar para serem recuperados. Os atributos selecionados são vizualizados após o usuário fazer a consulta.

Na parte central da tela é apresentado um conjunto de componentes gráficos que permitem a especificação de condições para restringir os resultados da recuperação. O número de atributos envolvidos nas condições limitou-se a três. Duas condições podem ser baseadas em atributos tradicionais e uma pode envolver o atributo áudio. O usuário pode escolher entre os diferentes atributos convencionais para estabelecer as duas condições. No caso

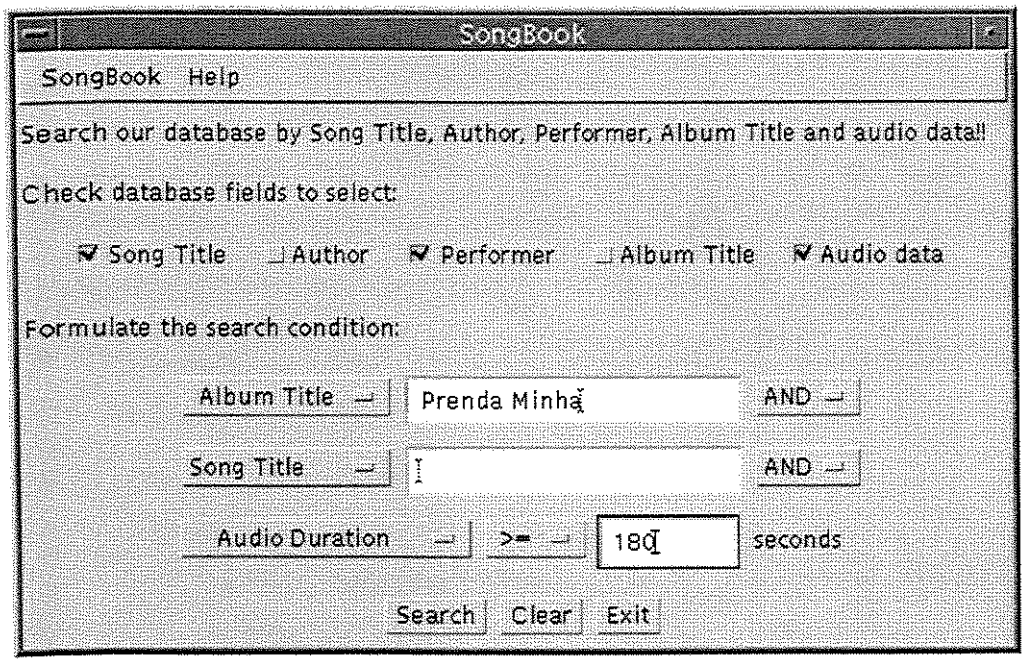

Figura 5.8: Sistema de Recuperação de Músicas - Interface principal.

do atributo áudio, a condição pode ser especificada escolhendo um dos critérios de busca oferecidos sobre áudio (tipo de conteúdo e duração).

Na parte inferior da tela são mostrados diferentes botões que permitem fazer a consulta, limpar as opções do usuário e encerrar a aplicação. Essas operações também estão disponíveis na barra de menu da interface.

Uma vez que o usuário seleciona os atributos a visualizar, especifica as restrições da consulta e a executa, apresenta-se uma nova janela que contém os dados recuperados. A Figura 5.9 mostra os resultados da consulta do exemplo que aparece na Figura 5.8.

Quando dados áudio são recuperados, a janela que mostra os resultados contém um conjunto de botões que permitem apresentar, deter a apresentação, acessar temporalmente e mudar o volume da mídia.

Adicionalmente, oferece-se informação geral sobre a mídia (formato, duração, etc.) e permite-se exportá-la para um arquivo. Caso o usuário resolva exportar um áudio, apresenta-se uma caixa de diálogo onde se solicita informação do arquivo a criar. As Figuras 5.10 e 5.11 mostram a interface que oferece informação geral sobre a mídia selecionada e a interface para exportar um áudio para um arquivo, respectivamente.

O banco de dados de músicas é gerenciado utilizando as funcionalidades de SGA. Utilizou-se o Driver desenvolvido para o SGBDR PostgreSQL.

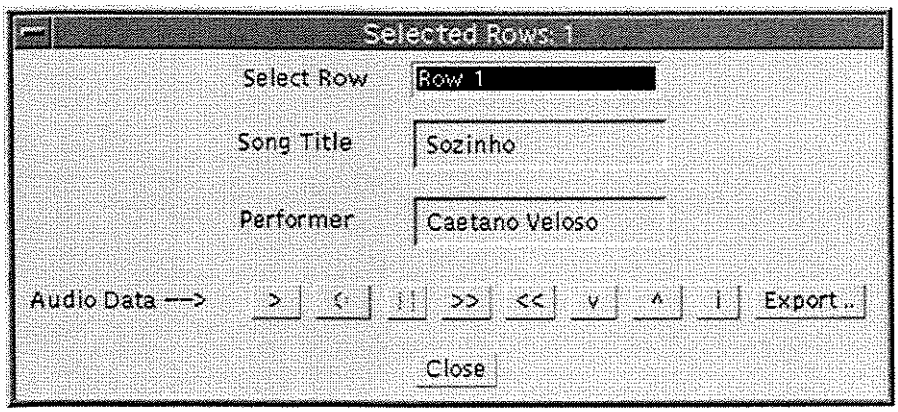

Figura 5.9: Sistema de Recuperação de Músicas - Interface para examinar dados recuperados.

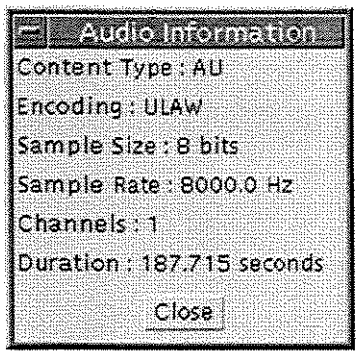

Figura 5.10: Sistema de Recuperação de Músicas - Interface para mostrar informação de um áudio.

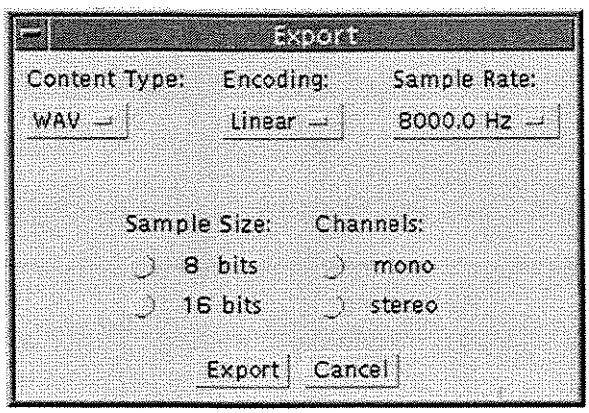

Figura 5.11: Sistema de Recuperação de Músicas - Interface para exportar um áudio.

5.4 Sumário

Este capítulo apresentou duas aplicações protótipo: Interface Gráfica de SGA e Sistema de Recuperação de Músicas, implementadas na linguagem Java. Elas utilizaram as funcionalidades de SGA para gerenciar bancos de dados que integram áudio, e apenas precisaram desenvolver os detalhes de domínio específico. O objetivo foi demonstrar que SGA é um sistema de propósito geral, que permite integrar dados convencionais e áudio, e que é independente de um SGBDR específico.

Capítulo 6

Conclusões e Trabalhos Futuros

O rápido incremento do número de aplicações e ferramentas multimídia vem estimulando a integração de dados multimídia aos sistemas de bancos de dados. As informações do tipo áudio têm recebido menor atenção devido a suas relações temporais e às dificuldades para analizar o conteúdo dessa mídia.

Os sistemas de bancos de dados relacionais são sistemas padronizados, utilizados amplamente para gerenciar dados convencionais. As estruturas desses sistemas não permitem estruturar adequadamente áudio. O esquema de blocos binários permite armazenar esses dados, mas não oferece mecanismos para sua interpretação, delegando tal tarefa às aplicações.

Este trabalho definiu AGA, uma arquitetura genérica que visa o desenvolvimento de sistemas de gerenciamento de áudio em junção com dados tradicionais. AGA é definida como uma camada intermediária entre um SGBDR com suporte para blocos binários e uma aplicação. Dessa maneira, aproveitam-se as vantagens oferecidas pelo SGBDR no gerenciamento de dados convencionais e seu mecanismo de armazenamento de blocos binários. Além disso, oferecem-se, aos programadores de aplicações, funcionalidades para definir e manipular bancos de dados relacionais que integram dados convencionais e áudio.

AGA engloba componentes que esboçam funcionalidades para definir e manipular relações que integram áudio através de um domínio para representá-lo. Operações de atualização e recuperação de dados podem integrar condições convencionais e baseadas em áudio.

A arquitetura proposta é de propósito geral, pois as funcionalidades de seus componentes independem do domínio das aplicações. Além disso, AGA é independente de um SGBDR específico, pois define em sua estrutura um componente que permite a transparência com respeito ao sistema de banco de dados utilizado pela aplicação.

A contribuição prática deste trabalho foi uma implementação simplificada da arquitetura proposta, o sistema SGA, e duas aplicações protótipo, Interface Gráfica de SGA e

Sistema de Recuperação de Músicas, que utilizam o sistema implementado.

SGA foi implementado em Java para oferecer um sistema independente de plataforma e possibilitar o gerenciamento de bancos de dados na Web.

Muito ainda pode ser feito para que AGA se torne uma plataforma atraente para os programadores de aplicações que incorporam áudio. Vários pontos de pesquisa foram identificados para serem resolvidos, tais como a integração de mecanismos e estruturas de indexação que possibilitem o acesso rápido a áudio, e a integração de facilidades para o controle de transações concorrentes.

Quanto a SGA, seu modelo de áudio pode ser estendido para representar diferentes tipos de áudio, por exemplo, fala e música. Adicionalmente, métodos de análise do conteúdo de áudio devem ser incorporados, o que permitirá a busca baseada em aspectos semânticos desses dados.

Apêndice A

Sintaxe do URL de um banco de dados em SGA

Um URL (*Uniform Resource Locator*) fornece informação sobre a localização de um recurso na Internet. Ele especifica o protocolo utilizado para acessar o recurso e sua localização.

O URL de um banco de dados em SGA permite identificar o banco de dados de maneira tal que um Driver apropriado possa reconhecê-lo e estabelecer uma conexão com ele.

A sintaxe padrão do URL de um banco de dados em SGA é:

```
ams:<driver_id>:<database_id>
```

O URL é dividido em três partes:

1. `ams`: protocolo de acesso aos dados (acrônimo de *audio management system*).
2. `<driver_id>`: identificador do Driver utilizado.
3. `<database_id>`: identificador do banco de dados, incluindo localização na Internet (se for remoto) e seu nome.

   A localização deve seguir a sintaxe:

   ```
   //hostname:port/database_name
   ```

O desenvolvedor de um Driver de SGA é quem determina seu identificador, que deve ser do conhecimento do usuário do Driver.

Apêndice B

Domínios convencionais de SGA

A Tabela B.1 mostra os domínios convencionais suportados na primeira versão de SGA (tipos SQL genéricos).

Categoria	Nome do domínio	Descrição
caracteres		
	char	um caracter
	text	cadeia de caracteres de comprimento variável
numéricos		
	integer	número inteiro
	float	número ponto flutuante
lógicos		
	boolean	`true` ou `false`
temporais		
	date	data
	time	tempo

Tabela B.1: Domínios convencionais de SGA

Apêndice C

Sintaxe de condições de manipulação em SGA

<CONDITIONS>: <CONDITION> [<LOGIC_OP> <CONDITION> [<LOGIC_OP> ...]]

<CONDITION>: <CONVENTIONAL_CONDITION> | <AUDIO_CONDITION>

<LOGIC_OP>: and | or

<CONVENTIONAL_CONDITION>: <CONVENTIONAL_ATTRIBUTE> <OP> <VALUE>

<AUDIO_CONDITION>: <AUDIO_METHOD> (<AUDIO_ATTRIBUTE> [, <PARAMETERS>]) <OP> <VALUE>

<CONVENTIONAL_ATTRIBUTE>: nome de atributo convencional

<AUDIO_ATTRIBUTE>: nome de atributo áudio

<OP>: >> | << | == | >= | <= | !=

<AUDIO_METHOD>: nome de um método da classe Audio (getContentType, getSampleSize, getSamplingRate, getChannels, getDuration, getEncoding)

<PARAMETERS>: <VALUE> [, <VALUE> [, ..]]

<VALUE>: valor integer, float, char, text, date, time, boolean

Bibliografia

Aoki, P. M. (1999). How to Avoid Building DataBlades that Know the Value of Everything and the Cost of Nothing, *Proceedings of the 11th International Conference on Scientific and Statistical Database Management*, Cleveland, OH.
URL: *http://GiST.CS.Berkeley.EDU:8000/gist/*

Atkinson, M., Bancilhon, F., DeWitt, D., Dittrich, K., Maier, D. e Zdonik, S. (1992). The Object-Oriented Database System Manifesto, *in* F. Bancilhon, C. Delobel e P. Kanellakis (eds), *Building an Object-Oriented Database System. The Story of O_2*, Morgan Kaufmann Publishers, San Mateo, CA, Capítulo 1.

Bu, L. e Chiueh, T.-D. (2000). Perceptual Speech and Phonetic Feature Mapping for Robust Vowel Recognition, *IEEE Transactions on Speech and Audio Processing* **8**(2): 105–114.

Campbell, S. T. e Chung, S. M. (1996). Database Aproach for the Management of Multimedia Information, *in* K. C. Nwosu, B. Thuraisingham e P. B. Berra (eds), *Multimedia Database Systems. Design and Implementation Strategies*, Kluwer Academic Publishers, Norwell, MA, Capítulo 2.

Codd, E. F. (1990). *The Relational Model for Database Management*, Addison-Wesley Publishing Company, Reading, MA.

Date, C. J. (1986). *Relational Database: Selected Writings*, Addison-Wesley Publishing Company, New York, NY.

Deitel, H. M. e Deitel, P. J. (1998). *Java: How to Program*, 2 edn, Prentice-Hall, Upper Saddle River, NJ.

Dionisio, J. D. e Cárdenas, A. F. (1998). A Unified Data Model for Representing Multimedia, Timeline, and Simulation Data, *IEEE Transactions on Knowledge and Data Engineering* **10**(5): 746–767.

Dittrich, A. K. e Dittrich, K. R. (1995). Where Object-Oriented DBMSs Should Do Better: A Critique Based on Early Experiences, *in* W. Kim (ed.), *Modern Dabase Systems*, Addison-Wesley Publishing Company, ACM Press, New York, NY, Capítulo 12.

Du, T. C.-T. e Wolfe, P. M. (1997). Overview of Emerging Database Architectures, *Computers & Industrial Engineering* **32**(4): 811–821.

Elmasry, R. e Navathe, S. B. (1994). *Fundamentals of Database Systems*, 2 edn, Benjamin/Cummings, Redwood City, CA.

Foote, J. (1999). An Overview of Audio Information Retrieval, *Multimedia Systems* **7**(1): 2–10.

Ghias, A., Logan, J., Chamberlin, D. e Smith, B. C. (1995). Query by Humming, *Proceedings ACM Multimedia.*

Gibbs, S. J. e Tsichritzis, D. C. (1995). *Multimedia Programming -Objects, Environments and Frameworks*, Addison-Wesley Publishing Company, Reading, MA.

Grosky, W. I. (1994). Multimedia Information Systems, *IEEE Multimedia* **1**(1): 12–24.

Grosky, W. I. (1998). Managing Multimedia Information in a Database Enviroment, *in* W. Litwin, T. Marzy e G. Vossen (eds), *Lecture Notes in Computer Science*, Vol. 1475.

IBM (2000). DB2 Universal Database Extenders.
URL: *http://www-4.ibm.com/software/data/db2/extenders/*

Informix (2000). Informix Dynamic Server.
URL: *http://www.informix.com/servers/*

Informix & Muscle Fish (2000). Audio Information Retrieval (AIR) DataBlade Module.
URL: *http://www.musclefish.com*

Kim, W. (1990). Object-Oriented Databases: Definition and Research Directions, *IEEE Transactions on Knowledge and Data Engineering* **2**(3): 327–341.

Momjian, B. (2000). PostgreSQL: Introduction and Concepts. A ser publicado por Addison-Wesley Publishing Company.
URL: *http://www.postgresql.org/docs/aw_pgsql_book/*

Naughton, P. (1996). *The Java Handbook*, McGraw-Hill, New York, NY.

Nierstrasz, O. (1989). A Survey of Object-Oriented Concepts, *in* W. Kim e F. H. Lochovsky (eds), *Object-Oriented Concepts, Databases, and Applications*, Addison-Wesley Publishing Company, ACM Press, New York, NY, Capítulo 1.

Pazandak, P. e Srivastava, J. (1997). Evaluating Object DBMSs for Multimedia, *IEEE Multimedia* **4**(3): 34–49.

PREDATOR Development Group (2000). PREDATOR - Object Relational DBMS. Universidade de Cornell, Ithaca, NY.
URL: *http://www.cs.cornell.edu/database/predator/*

Seshadri, P. (1998). PREDATOR: A Resource for Database Research, *SIGMOD RECORD* **27**(1): 16–20.

Seshadri, P., Livny, M. e Ramakrishnan, R. (1997). The Case for Enhanced Abstract Data Types, *Proceedings of 23th VLDB Conference.*

Silberschatz, A., Korth, H. F. e Sudarshan, S. (1997). *Database System Concepts*, 3 edn, McGraw-Hill, New York, NY.

Stonebraker, M. e Brown, P. (1999). *Object-Relational DBMSs*, 2 edn, Morgan Kaufmann Publishers, San Francisco, CA.

Strawn, J. (1994). Digital Audio Representation and Processing, *in* S. Cunningham (ed.), *Multimedia Systems*, Addison-Wesley Publishing Company, ACM Press e SIGGRAPH Series, New York, NY, Capítulo 4.

Sukkor, R. A., Gandhi, M. B. e Setlur, A. R. (2000). Speaker Verification Using Mixture Decomposition Discrimination, *IEEE Transactions on Speech and Audio Processing* **8**(3): 292–299.

The PostgreSQL Development Group (1998). PostgreSQL Programmer's Guide. University of California, Berkeley, CA.
URL: *http://www.postgresql.org/docs/programmer/index.html*

Wold, E., Blum, T., Keislar, D. e Wheaton, J. (1996). Content-Based Classification, Search, and Retrieval of Audio, *IEEE Multimedia* **3**(3): 27–36.

Yoshitaka, A. e Ichikawa, T. (1999). A Survey on Content-Based Retrieval for Multimedia Databases, *IEEE Transactions on Knowledge and Data Engineering* **11**(1): 81–94.