ADRIANA DOS SANTOS

ANÁLISE DE DESEMPENHO DA PERSISTÊNCIA DE OBJETOS EM BASES DE DADOS OBJETO-RELACIONAL.

São Paulo
2013

ADRIANA DOS SANTOS

ANÁLISE DE DESEMPENHO DA PERSISTÊNCIA DE OBJETOS EM BASES DE DADOS OBJETO-RELACIONAL.

Monografia apresentada a Escola Politécnica da Universidade de São Paulo para a conclusão do curso de MBA em Tecnologia da Informação.

Área de Concentração: Tecnologia da Informação.

Orientadora: Profª. Dra. Solange Nice Alves de Souza

São Paulo

2013

FICHA CATALOGRÁFICA

Santos, Adriana dos.
Análise de desempenho da persistência de objetos em base de dados objeto-relacional / A. dos Santos. -- São Paulo, 2013.
58p

Monografia (MBA em Tecnologia da Informação) – Escola Politécnica da Universidade de São Paulo. Programa de Educação Continuada em Engenharia.

1. Banco de dados (Desempenho; Análise) 2. Banco de dados objeto-relacional 3. Bases de dados 4. Padrão SQL I. Universidade de São Paulo. Escola Politécnica. Programa de Educação Continuada em Engenharia II. t.

ADRIANA DOS SANTOS

ANÁLISE DE DESEMPENHO DA PERSISTÊNCIA DE OBJETOS EM BASES DE DADOS OBJETO-RELACIONAL.

Monografia apresentada a Escola Politécnica da Universidade de São Paulo para a conclusão do curso de MBA em Tecnologia da Informação.

São Paulo

2013

DEDICATÓRIA

Dedico este trabalho ao corpo docente da Escola Politécnica pelo incentivo e orientações nas questões técnicas, e a todos que direta ou indiretamente contribuíram com criticas ou sugestões para o aperfeiçoamento de meus conhecimentos. E consequentemente para minha progressão acadêmica auxiliando na conclusão deste trabalho o qual jugo de extrema importância.

AGRADECIMENTOS

Primeiramente agradeço a Deus, amigo e companheiro onipresente por mais esta etapa de minha vida, que gentilmente me acompanha.

A Profª. Dra. Solange Nice Alves de Souza, pela paciência e dedicação. Sendo muito mais que uma orientadora, mas, também amiga. Atuando sempre de forma eficaz e comprometida.

Ao amigo Carlos Rombaldo que por muitas vezes auxiliou em questões técnicas, sendo pró-ativo e companheiro, mesmo sem esperar nada em troca, caso precisasse independente da hora e de onde se encontrasse se dispunha a ajudar, "muito obrigada".

A meus pais e familiares pelo apoio emocional, e principalmente ao meu irmão Edmundo José dos Santos Junior, que de demasiadas formas me acompanhou pelas madrugadas para a finalização deste trabalho.

EPÍGRAFE

"Sua meta é ser o melhor do mundo naquilo que você faz. Não existem alternativas".

(FALCONI).

RESUMO

Este trabalho visa à elaboração de testes de desempenho controlados a partir de simulações realizadas com dados reais. Tendo por objetivo avaliar o desempenho de Banco de Dados Objeto-Relacional (BDOR), que se mostram mais apropriados para aplicações que exigem estruturas complexas e persistência de objetos, tais como: Sistemas de Informações Geográficas (SIG), aplicações de engenharia, sistemas de multimídia, etc.. Para avaliar o desempenho optou-se pela comparação com BDR. Para tal utilizou-se o Sistemas Gerenciadores de Banco de Dados (SGBD) Oracle 11g em plataforma Windows, por esse SGBD oferecer tanto uma estrutura relacional como Objeto-Relacional (OR), adicionando os novos recursos para o tratamento de objetos em Banco de Dados (BD), de acordo com o definido na especificação SQL. No estudo, uma aplicação foi implementada nas duas estruturas usadas para o comparativo de desempenho, ou seja, BDR e BDOR, de modo a manter a similaridade e a consistência dos dados. Nos testes observou-se o comportamento para as operações de inclusão, além de algumas consultas. Os resultados apesar de não serem "estressantes" são representativos e constituem informações importantes para aqueles que precisam da representação de dados complexos.

Palavras-chave: Banco de Dados Objeto-Relacional; Desempenho de Banco de Dados; Desempenho de BDOR, Padrão SQL.

ABSTRACT

This work aims at the development of performance tests controlled from simulations with real data. Having for goal to evaluate the performance of Object Relational Database (ORDB), which are more suitable for applications requiring complex structures and persistent objects such as: Geographic Information Systems (GIS), engineering applications, multimedia systems, etc. The performance evaluation was made by the comparison between the RDB and ORDB. The DBMS Oracle 11g was used to achieve this goal. The DBMS allowed implement the application in both structures, i.e. RDB and ORDB, according to SQL standard. During the tests, we observed the behavior of operations for inclusion. Besides, some queries were also considered. Despite of not being "stressful", the results are representative and provide important information for those who need the representation of complex data.

Keywords: Object-Relational Database, Database performance, ORDB performance, SQL Standard.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIMBOLOS E SIGLAS

ANSI. - *American National Standards Institute.*

BD. – Banco de Dados.

BDR. – Banco de Dados Relacional.

BDOR. – Banco de Dados Objeto Relacional.

BDOO. - Banco de Dados Orientado a Objetos.

DML. – *Data Manipulation Language.*

EMC. – *EMC Corporation.*

FIPS. – *Federal Information Processing Standard.*

IBM. – *International Business Machines.*

IDC. - *International Data Corporation*

IEC. – *International Electrotechnical Commission.*

ISO. - *International Organization for Standards.*

OO. – Orientação a Objetos

OR. – Objeto-Relacional

PK. – *Primary Key*

SGBD. – Sistema Gerenciador de Banco de Dados.

SGBDOR. - Sistema Gerenciador de Banco de Dados Objeto-Relacional.

SGBDR. - Sistema Gerenciador de Banco de Dados Relacional.

SIBI. - Sistema de Bibliotecas e informação da UFRJ.

SQL. - *Structured Query Language*

SQL-86. – Norma SQL-86 "ANSI X3.135-1986 e ISO 9075:1987".

SQL-89. – Norma SQL-89 "ANSI X3.135.1-1989".

SQL-92. – Norma SQL2 "ISO/IEC 9075:1992"..

SQL:1999. - Norma SQL3 "ISO/IEC 9075:1999".

SQL:2003. - Norma "ISO/IEC 9075:2003". Suporte a XML.

SQL:2006. - Norma "ISO/IEC 9075:2006".

SQL:2008. - Norma "ISO/IEC 9075:2008".

SQL:2011. - Norma "ISO/IEC 9075:2011".

UDT. – *User Defined Type.*

UFRJ. – Universidade Federal do Rio de Janeiro.

SUMÁRIO

1. INTRODUÇÃO.

Nas últimas décadas, com os avanços tecnológicos, surgiram novas aplicações que envolvem dados complexos e/ou multimídia, e o paradigma de Orientação a Objeto (OO). Entretanto, adequar as novas aplicações ao BDR passou a exigir um esforço adicional para armazenar um objeto com estrutura complexa e outro para adequá-lo novamente no momento de recuperá-lo. Aliado a isso, com o volume crescente dos dados, a busca pela informação começou a se tornar mais difícil e demorada.

Limitados pelas restrições do modelo relacional, pesquisadores desenvolveram modelos de dados que incorporavam o paradigma de OO, dando origem aos Bancos de Dados Orientados a Objeto (BDOO). Porém, apesar de atenderem as exigências das novas aplicações com uma estrutura mais rica, seu péssimo desempenho e falta de padrão acabou por limitar seu uso. Outra iniciativa foi à criação do BDOR, que pode ser visto como uma extensão do BDR unindo a necessidade de processamentos de dados complexos a uma estrutura com processamento eficaz (KORTH; SILBERSCHATZ; SUDARSHAN, 2006). E, como demonstrado nos estudos de Rombaldo (2012) e Castro (2011) os recursos disponíveis no BDOR evidenciam vantagens, entre as quais é possível citar a representação de objetos em banco de dados, herança, uso de referências para representação de relacionamentos, tipos de dados complexos, coleção, entre outros.

Esses novos recursos permitem que o descasamento de impedância, que tem sido relatado como uma desvantagem no uso de BDRs seja eliminada, sendo então possível mapear os objetos da aplicação em objetos de Banco de Dados (BD). O que incentiva sua aplicação em ambientes de produção, como em Sistemas de Informação Geográficas, que dependem de grandes volumes de dados, para gestão ou representação do espaço, ou até mesmo para auxilio em áreas da medicina, como ortopedia, que para um diagnóstico eficiente, precisam compor objetos complexos multidimensionais e dados variados (imagens, números, descrições, etc.).

Contudo, pouco tem sido divulgado sobre o comportamento deste tipo de BD, quando esses novos recursos são largamente utilizados e em grandes volumes de dados. Acredita-se que, por ser uma tecnologia relativamente nova, pouco tem sido realmente empregado em ambiente de produção, e isto se deve tanto a falta de

conhecimento efetivo dos novos recursos como também a falta de informações válidas sobre o seu desempenho. Assim, entende-se que faltam estudos comparativos sobre seu desempenho e ampla divulgação dos resultados.

1.1 Motivação.

Por ser atualmente a informação o bem mais precioso de uma organização, a escolha pelo SGBD adequado às necessidades da empresa deve ser feito com muita cautela. Estudos da *International Data Corporation* (IDC) junto com a EMC em junho de 2011 revelaram que o volume de informações dentro das empresas cresce 34% ao ano, chegando a 60% em algumas empresas, exigindo mais espaço de armazenamento e otimização no gerenciamento dos dados. Outra questão importante é o tipo e formato em que os dados estão sendo gerados e armazenados, podendo ter ou não correlações claras se seguirem uma estrutura pré-definida. Com isso, fornecedores de Banco de Dados (BD), como Oracle e IBM já disponibilizaram em suas versões a inclusão de novos tipos de dados e um maior suporte a OO, tendo como alvo usar dos conceitos de OO descritos na norma SQL: 2011. O que acaba por justificar a utilização real do Sistema Gerenciador de Banco de Dados Objeto-Relacional (SGBDOR).

No entanto, para seu efetivo emprego em ambientes de produção, por melhor que sejam os possíveis benefícios da sua utilização é preciso que se tenham informações sobre seu desempenho, além da avaliação do comportamento dos novos tipos e elementos disponibilizados. Nesse contexto, entende-se que a comparação com o desempenho de BDR é natural e fornece dados importantes para a avaliação do BDORs. Portanto, a principal motivação para o desenvolvimento deste trabalho, não é mostrar qual é o melhor SGBD, e sim apresentar qual seria o mais adequado para a adoção das empresas de acordo com o tipo de aplicação utilizada e/ou desenvolvida para armazenar e recuperar seus dados da forma mais hábil possível. Sem desconsiderar o conjunto de benefícios do SGBDOR.

1.2 Objetivo.

O objetivo deste trabalho é aplicar testes de desempenho em BDOR a fim de obter informações que incentivem seu uso efetivo em ambientes de produção. Para ter algum parâmetro de comparação, optou-se também por comparar os resultados obtidos com os de BDR. Cabe ressaltar, porém, que entende-se que as duas

estruturas de BD são apropriadas para diferentes tipos de aplicações, ou seja, este trabalho não promove o uso de um em substituição ao outro, mas o uso de BDOR para aplicações em que é necessário o uso de estruturas mais complexas de objetos.

Foi escolhido o SGBD Oracle 11g, para a execução dos testes de desempenho porque este possui os recursos disponíveis tanto para o modelo relacional como para o suporte a objetos (BDOR). Apesar dos testes realizados não serem suficientes para uma avaliação mais profunda em relação aos recursos de objetos disponíveis em BDORs, ainda sim fornecem dados importantes que podem ser somados a outros esforços para efetivamente alavancarem o uso de BDORs em aplicações onde BDRs não têm se mostrado eficientes.

Foi utilizada uma base de dados que pode ser modelada e gerada nas duas estruturas. Os resultados dos testes foram avaliados de forma a identificar os pontos fortes e fracos de cada aplicação, além de detectar em qual cenário há o maior ganho ou perda de desempenho.

1.3 Resultados Esperados.

Dentre toda atividade desenvolvida no conjunto deste trabalho, espera-se que o processo de comparação do desempenho entre os dois BDs assim como o modelo e cargas usadas, permitam medir e qualificar os resultados obtidos. De forma que os resultados provejam informações confiáveis para a escolha apropriada do BD que melhor se adapte as necessidades da empresa.

Entretanto, dado a simplicidade da aplicação utilizada neste trabalho, dos testes e da limitação inerente à aplicação, entende-se que o real conhecimento ou expectativa do desempenho será obtido com o uso crescente dos novos recursos e a divulgação dos resultados, e isso é o que efetivamente impulsionará a utilização de SGBDORs.

1.4 Metodologia.

Este trabalho se iniciou por uma investigação ao tema BDOR. E para que fosse possível a implementação dos conceitos e teorias investigadas, foram levantadas as características do SGBD Oracle que comporta tanto o modelo relacional como objeto-relacional. Em paralelo, estudou-se a especificação da SQL considerando as versões que possuem recursos de orientação a objeto, SQL: 1999 á SQL: 2011.

Os estudos iniciais foram primordiais para o entendimento e discernimento das principais diferenças entre os padrões da SQL e na sintaxe apresentada para o SGBD Oracle, assim como, para a elaboração dos testes reais realizados no SGBD.

Estas investigações e estudos ocorreram juntamente com a fase de levantamento bibliográfico, os quais tiveram como principal fonte, livros, artigos publicados em comunidades cientificas e dissertações, destacando trabalhos como os de Rombaldo, Castro e Souza (2009) em seu artigo "Banco de Dados Objeto-Relacional: Comparação do Suporte Oferecido por SGBDs para a Persistência de Objetos", o qual apresenta um diagrama que ilustra exatamente os novos tipos de dados que atendem o paradigma de orientação a objeto. E a dissertação de mestrado de Rombaldo (2012), que, de todos os trabalhos, mostrou de forma mais real a aplicação de um BDOR e a criação de um Framework para implementação do mesmo. Estes trabalhos foram fundamentais para a elaboração do segundo e do terceiro capítulo desta monografia.

Na evolução da pesquisa, o foco foi o SGBDOR, suas características e formas de implementação dos novos conceitos sob o paradigma da OO utilizando operações de inclusão, e consultas com volume de dados variados.

1.5 Revisão Bibliográfica.

Por se tratar, basicamente, de uma complementação a trabalhos correlatos, que apresentam estudos aprofundados sobre BDOR e possíveis melhorias para sua implementação, nesta seção são comentados os trabalhos que fomentaram a elaboração desta monografia. Estes trabalhos apresentam resultados importantes para o incentivo do uso efetivo de DBOR em ambientes de produção. Assim, eles funcionaram não apenas como fonte de informação e guia para a elaboração desta monografia, como também apontaram pontos importantes de pesquisa para a promoção de BDOR.

Na dissertação de Rombaldo (2012), a proposta de um Framework de Persistência de Objetos em BDOR, apresenta não só as funcionalidades do SGBD Objeto-Relacional, mas também um estudo mais aprofundado das especificações da SQL: 2003 e 2008. O trabalho apresenta como principal contribuição o Framework O-ODBM (*Object-Object Database Mapping*), destacando as vantagens de sua utilização para a elaboração e implantação de esquemas objeto-relacional no SGBD Oracle 11g. Frameworks de persistência para banco de dados relacionais como

Hibernate, Torque e outros têm sido cada vez mais empregados para facilitar o mapeamento de objetos da aplicação em tabelas relacionais, além de atuarem como ponto de criação e acesso ao esquema relacional. Esses frameworks auxiliam na melhora da produtividade de desenvolvedores, pois elimina a necessidade de conhecimentos mais específicos da teoria relacional e da linguagem SQL. Isso permite que desenvolvedores se atenham aos requisitos da aplicação e ao ambiente de programação, diminuindo assim, os conceitos com que precisam lidar. Além disso, em ambiente onde se precisa persistir os dados em diferentes SGBDs, o uso de frameworks de persistência simplifica o trabalho e diminuem o tempo gasto, sendo necessária apenas a alteração de uma diretiva de configuração (ou uma linha de código). Em compensação, sem o uso de um framework, seria necessária a revisão manual de todo o código SQL por parte do desenvolvedor, o que, obviamente, demandaria mais tempo e conhecimento específico. Tendo em vista as vantagens da utilização de frameworks de persistência, Rombaldo trouxe como principal contribuição o framework O-ODBM, que diferentemente dos já citados enxerga os novos tipos e elementos de BDOR, e mapeia objetos da aplicação em objetos de banco de dados, eliminando assim o casamento de impedância. Os resultados da dissertação foram divulgados nos trabalhos (Rombaldo, Souza e De Souza 2012A) e (Rombaldo, Souza e De Souza, 2012B).

Na dissertação de Castro (2011), Modelo Lógico Gráfico para BDORs, aborda a representação de objetos em banco de dados, mostrando os principais aspectos de BDORs. O trabalho apresenta como principal contribuição à proposta de um modelo lógico gráfico para a elaboração de esquemas de BDOR. Esse modelo pode ser elaborado usando a ferramenta Case, também produto resultante da dissertação. A ferramenta Case é uma extensão da ArgoUML, onde módulos foram incorporados para o tratamento de objetos em banco de dados. Através da ferramenta o modelo lógico gráfico é traduzido automaticamente para código SQL nos dialetos SQL: 2003 e Oracle 11g. Os resultados da dissertação são também apresentados em (Castro, Souza e De Souza, 2012).

O Artigo Rombaldo, Castro e Souza (2009) apresenta um comparativo do suporte oferecido pelos SGBDs Oracle e PostgreSQL na persistência de objetos. Este trabalho deu subsídios importantes para a compreensão de BDOR, além de apresentar o suporte de objetos ofertado por alguns SGBDs. Consequentemente este artigo foi o ponto de partida no desenvolvimento desta monografia.

1.6 Estrutura do Trabalho.

O presente trabalho está organizado em 5 capítulos: O primeiro capítulo trata da introdução com conceitos e informações da ideia principal que originaram este trabalho. O segundo capítulo aborda a fundamentação teórica, com assuntos importantes sobre BDOR e Padrão SQL, enfatizando uma breve contextualização sobre o SGBD Oracle, suas principais características, vantagens e desvantagens na utilização. O terceiro capítulo apresenta a aplicação, mostrando todos os processos e conceitos utilizados para a elaboração do projeto. Estudo dos modelos, testes e amostragens de testes comparativos. Seguindo etapas de controle previamente definidas, o quarto capítulo, apresenta as avaliações dos testes de desempenho realizados no capítulo três e seus resultados. No quinto capítulo seguem as conclusões e aspirações para trabalhos futuros. E, por fim, tem-se as referências bibliográficas.

2. FUNDAMENTAÇÃO TEÓRICA

Esta seção aborda os principais conceitos e informações sobre BDORs, assim como um aprofundamento da linguagem de manipulação SQL, em sua versão estendida SQL: 2011 com suporte a modelo de objetos. E por último uma breve descrição a respeito do SGBD Oracle 11g e seu suporte a OO.

2.1 Banco de Dados Objeto-Relacional (BDOR).

Nos últimos anos, vários fornecedores estenderam seus SGBDs para fornecer suporte a objetos de banco de dados. Exemplos incluem DB2, Oracle 11g, SQL Server, PostgreSQL, MySQL, entre outros. A ideia geral em cada caso é admitir recursos tanto de objetos quanto relacional, não descartando mais de 40 anos de pesquisa voltadas para o modelo relacional, inicialmente apresentada pelo Dr. E. F. Codd (1970). Com isso, esses SGBDs permitem o tratamento de objetos em banco de dados, eliminando o problema do descasamento de impedância (*impedance mistmatch*) e, permitindo o tratamento mais apropriado para aplicações complexas que precisam representar estruturas de dados complexas (DATE, 2000).

Assim, combinando as melhores características da OO ao modelo relacional, e com uma implementação apropriada, resultou-se no BDOR, uma extensão do BDR capaz de tratar tipos de dados complexos, considerada mais eficiente e inteligível (BLAHA; RUMBAUGH, 2006).

Em BDOR pode-se definir tabelas aninhadas, tipos definidos pelo usuário (UDTs), definir relacionamentos usando referências (REF), identificadores de objetos (OID), coleções, herança e métodos. Atendendo aos requisitos da nova geração de aplicações de negócio (ELMASRI, NAVATHE, 2005).

2.2 Evolução da Linguagem e do Padrão SQL.

A SQL (*Structured Query Language)* é uma linguagem de quarta geração. Diferente de uma linguagem de programação, a SQL é uma linguagem declarativa, isto é, descreve o problema ao invés da solução, especificando o que deve ser feito e não como (LIMA, 2011). Padronizada em 1986 pelo *American National Standards Institute* (ANSI) e em 1987 pela *International Organization for Standardization* (ISO).

O padrão recebeu uma série de revisões incorporando aspectos da OO em 1999 dando origem a versão conhecida como SQL: 1999. Desde então, com as versões SQL: 2008 e a mais recente SQL: 2011, novos elementos foram

acrescentados, outros foram removidos ou compartilhados, melhorando o suporte a OO (CASTRO, 2011).

A Tabela 1 apresenta as versões da SQL desde sua padronização em 1986.

Etapas de desenvolvimento do padrão SQL		
Ano	**Nome**	**Descrição**
1986	SQL-86 (SQL-87)	Primeiro padrão formal feito pela ANSI.
1989	SQL-89 (FIPS 127-1)	Menor Revisão – Adição de restrições de integridade. Adotada como FIPS 127-1.
1992	SQL-92 (SQL2, FIPS 127-2)	Maior revisão (ISO 9075). Adicionados novos tipos de dados, suporte para conjuntos de caracteres adicionais, funções escalares, definição de esquema e manipulação de dados. Pela primeira vez são utilizados os comandos ALTER (para alterar um esquema) e DROP (para excluir).
1999	SQL: 1999 (SQL3)	Matching de expressões regulares, Queries recursivas, Triggers, suporte de procedimentos e controle de fluxo, tipos não escalares. Funcionalidades de OO.
2003	SQL: 2003	Introduzidos recursos relacionados à XML (SQL/XML) e suporte para linguagem de programação Java. Além disso, foi introduzido o uso de sequências, colunas com autovalores gerados e valores de identidade.
2006	SQL: 2006	ISO / IEC 9075-14:2006 define formas de SQL que pode ser usado em conjunto com XML, como importação e armazenamento de dados XML em um banco de dados SQL, além de permitir o uso de XQuery, linguagem de consulta XML publicado pelo World Wide Web Consortium (W3C), para acessar SQL-XML de dados e documentos. No entanto, a SQL: 2006 não faz uma revisão completa da SQL: 2003, mas, somente a revisão da parte 14, que trata de SQL/XML.
2008	SQL: 2008	ORDER BY fora da definição de cursores, trigger INSTEAD OF, TRUNCATE.
2011	SQL: 2011	ISO / IEC 9075: 2011. Entre suas melhorias há a possibilidade de emitir, inserir, modificar e apagar os comandos usados diretamente em consultas e suporte para tabelas temporárias (Temporal Extensions).

Tabela 1: Etapas de desenvolvimento do padrão SQL.
Fonte: (SCHWEINSBERG, 2012)

Percebe-se que a SQL evoluiu gradativamente durante as ultimas décadas se tornando uma linguagem bastante complexa, com capacidade de atender as mais diversas necessidades (KORTH et. al., 2006).

Atualmente o padrão SQL é composto pelos seguintes componentes descritos na Tabela 2.

Componentes do Padrão SQL		
	Componente	**Descrição**
1	Framework (SQL/Framework)	Descreve cada parte do padrão e contém informações comuns a todas as partes.
2	Foundation (SQL/Foundation)	Determina a sintaxe e semântica para definição e manipulação dos dados na linguagem SQL.
3	Call Level Interface (SQL/CLI)	Desenvolvida pelo grupo SQL Access, essa parte corresponde à especificação do ODBC da Microsoft.
4	Persistent Stored Modules (SQL/PSM)	Define instruções SQL de linguagem procedural que são úteis em funções e procedimentos definidos pelo usuário.
5	Host Language Bindings (SQL/Bindings)	Especifica como a SQL é embutida em linguagens de programação não orientadas a objetos.
6	XA Specialization (SQL/Transaction)	Especialização SQL da especificação X/OPEN XA (Cancelada em 1999).
7	Temporal (SQL/Temporal)	Define suporte para armazenamento e obtenção de dados temporais.
8	Extended Objects (SQL/Objects)	Determina como tipos de dados abstratos definidos em aplicações são tratados por SGBDs.
9	Management of External Data (SQL/MED)	Determina sintaxe e definições adicionais para a SQL/Foundation que permitem à SQL acessar fontes de dados não SQL (arquivos).
10	Object Language Bindings (SQL/OLB)	Especifica a sintaxe e semântica de SQL embutido na linguagem de programação Java. Corresponde a outro padrão ANSI, o SQLJ Parte 0.
11	Information and Definition Schemas (SQL/Schemata)	Informação e definição de esquemas.
12	Replication (SQL/Replication)	Define suporte e funcionalidades para replicação de um banco de dados SQL.

13	SQL Routines and Types Using the Java Programming Language (SQL/JRT)	Define como o código Java pode ser usado dentro de um banco de dados SQL.
14	XML-Related Specifications (SQL/XML)	Define como XML pode ser usado dentro de um banco de dados SQL. Essa parte está alinhada com especificação XQuery 1.1 da W3C.

Tabela 2: Componentes do padrão SQL.
Fonte: <www.sirmacstronger.eti.br>

2.2.1 Principais características da SQL para aplicações OR.

Para melhor entender os principais elementos da linguagem SQL, uma ontologia representada por um diagrama de classes UML é apresentada na Figura 1. A ontologia foi construída com base na SQL: 2003. Contudo, ela é válida também para as especificações posteriores, uma vez que, o suporte oferecido para a OO não sofreu grandes modificações, sendo o maior diferencial entre a SQL: 2003 e 2008 o suporte à XML, e entre estas e a SQL: 2011 o suporte aos aspectos do tratamento temporal ao dado, chamado de "Extensões Temporárias" (definições de período, sistema de tempo por período, aplicações de tempo período, etc.), que permite a validação física de dados simplificando a codificação da aplicação.

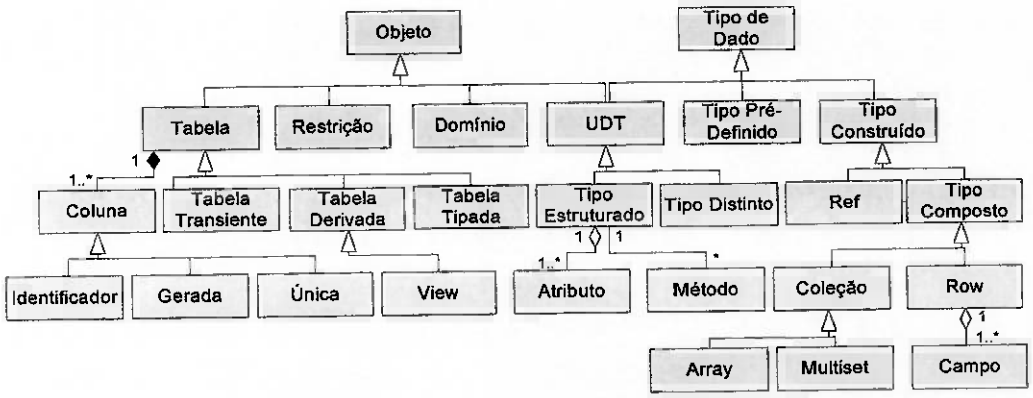

Figura 1: Diagrama de Classes UML apresentado os elementos da SQL: 2003 e seus inter-relacionamentos para suporte a BDOR

O diagrama da Figura 1 está dividido em duas sub-hierarquias, uma que especifica os objetos que podem ser definidos num BDOR e outra os tipos de dados que podem ser empregados.

Os tipos de dados são classificados em três classes principais: Tipo Pré-Definido, Tipo Construído e Tipo Definido pelo Usuário (UDT - User Defined Type).

Conforme Figura 1, os tipos construídos consistem em: tipos atômicos, como o REF (referência), tipos complexos, como ROW, ou ainda coleções, como ARRAY e MULTISET (CASTRO, 2011).

O tipo REF corresponde a um valor que faz referência a (ou aponta para) um valor do tipo referenciado. Os únicos valores que podem ser referenciados são as linhas de tabelas tipadas. (CASTRO, 2011).

Uma ROW é uma estrutura formada por atributos e seus respectivos tipos de dados que permite armazenar valores estruturados em uma única coluna (LIMA, 2011).

Um ARRAY é uma coleção ordenada de valores, cujos elementos são referenciados pela posição ordinal, e sua estrutura de dados é composta por um número definido de elementos do mesmo tipo. Útil para atributos multivalorados (LIMA, 2011). Já o tipo MULTISET é uma coleção de elementos não ordenados, não possui cardinalidade, pois, o espaço de armazenamento é alocado dinamicamente. Além disso, pode ser usado para criar composições ou objetos complexos (ROMBALDO, 2012).

Ainda na Figura 1, um Tipo Definido pelo Usuário (*User Defined Type* - UDT), é um tipo complexo definido pelo usuário, equivalente ao objeto (OO) no contexto do BD. Um UDT encapsula atributos e métodos da mesma forma que um objeto OO, implementa herança e polimorfismo. Tabelas Tipadas são criadas a partir de um UDT (ROMBALDO, 2012). UDTs podem ser das categorias Tipo Distinto ou Tipo Estruturado. Um tipo distinto corresponde a um UDT que é baseado em algum tipo pré-definido. Os valores de um tipo distinto são representados pelos valores do tipo em que se baseia. Já, um Tipo Estruturado é composto por um ou mais atributos, pode gerar hierarquias supertipo-subtipo, e apresentar métodos para manipular o Tipo Estruturado a que se refere. Um Tipo Estruturado pode ser utilizado para definir uma tabela tipada, ou para especificar o domínio de campos de uma tabela (CASTRO, 2011).

2.3 ORACLE.

O Oracle é um SGBD que surgiu no final da década de 70. Foi uma descoberta de Larry Ellison que ao encontrar uma descrição de um protótipo funcional, baseado

no trabalho de Edgar Frank Codd (que trabalhava na IBM), sobre banco de dados relacionais, descobriu que nenhuma empresa tinha se empenhado em comercializar essa tecnologia. Ellison e os cofundadores da *Oracle Corporation*, Bob Miner, Ed Oates, lançaram o primeiro SGBD comercial do mercado, vendido em sua segunda versão para à Base da Força Aérea americana em 1979. A primeira versão nunca foi lançada (KELLY, 2007).

Hoje a Oracle é um dos principais fornecedores de software para gerenciamento de informações e a segunda maior empresa de software independente do mundo. Além da base de dados que atualmente encontra-se na versão 11 g, a Oracle desenvolve uma suíte de desenvolvimento chamada *Oracle Developer Suite*, que interage com sua base de dados e criou a linguagem de programação *PL/SQL,* utilizada no processamento de transações (ORACLE, 2013).

2.3.1 Características do SGBD Oracle.

Um BD Oracle tem estruturas lógicas e estruturas físicas. Na estrutura lógica incluem esquemas, blocos de dados, extensões, segmentos e tablespace. Já em sua estrutura física temos: arquivos de dados, arquivos de log de rede (registram todas as mudanças realizadas no BD) e arquivos de controle.

A versão do Oracle utilizada nesta pesquisa foi o Oracle database 11g Express Edition, um banco de dados de pequenas dimensões com base no Oracle 11g Database Release base de código 2 (ORACLE, 2013).

O SGBD Oracle 11g proporciona suporte aos três tipos de implementação: Relacional, Objeto-Relacional e Orientado a Objetos. Suporta dois tipos de tabelas: Tabela Relacional e Tabela de Objetos (*Object Table*). E, para OR oferece diferentes tipos de objetos, sendo os principais:

- Tipos de Objetos (TADs): um tipo abstrato de dados definido pelo usuário que encapsula propriedades (atributos) e comportamento (métodos). Não aloca espaço de armazenamento e nem pode armazenar dados.
- Nested Tables (Tabelas aninhadas - MULTISET): conjunto não ordenado sem número fixo de elementos. Interessante quando se precisa executar consultas eficientes em coleções ou efetuar várias operações de insert, update ou delete.
- VArrays (*Varying Arrays*): coleção ordenada com numero fixo de elementos e mesmo tipo de dados.

- References (REF): ponteiro lógico para um "*row object*". Usado para fazer referências. REF e coleções de REFs são utilizadas na modelagem de associações entre objetos. Por exemplo, o relacionamento entre uma ordem de compra e um cliente.
- *Object View* (Visão de Objetos): tabelas de objetos (*object table*) virtuais, a qual cada linha é um objeto.

3. A APLICAÇÃO.

Neste capítulo, foi definido o domínio da aplicação através da geração de um modelo de dados. Usando a mesma aplicação, primeiro foi elaborado o modelo com base em BDR e posteriormente elaborou-se o modelo para o BDOR. Uma vez que o diagrama de classes UML tenha sido usado como alternativa ao modelo entidade-relacionamento (ER) para a elaboração do esquema de BD, além de ser obviamente apropriado para o objetivo deste trabalho, optou-se por usar um diagrama de classes para a modelagem conceitual da aplicação usada para os testes de avaliação realizados. Nesta etapa também foi feita a configuração dos SGBDs e do equipamento a ser utilizado para a realização do projeto.

A aplicação usada para as avaliações propostas neste trabalho é relativamente simples, mas permite que vários tipos e objetos de BDOR possam ser avaliados. Com a mesma mostra de dados foi possível fazer testes diferentes, sendo que para cada teste foi gerado um modelo de dados especifico, de forma a permitir o melhor entendimento das associações e dos mapeamentos realizados para BDOR.

Assim, dividiram-se os testes pelos modelos, mostrando o mapeamento para BDOR, os códigos SQL utilizados e os respectivos resultados.

3.1 Descrição da aplicação.

O presente trabalho utiliza uma aplicação de rede, na qual objetos fazem parte de um contexto de negócio para exploração de vulnerabilidades, conforme Figura 2. Nesta, um diagrama de classe é usado para descrever os objetos da aplicação. Apenas objetos persistentes são apresentados no diagrama da Figura 2.

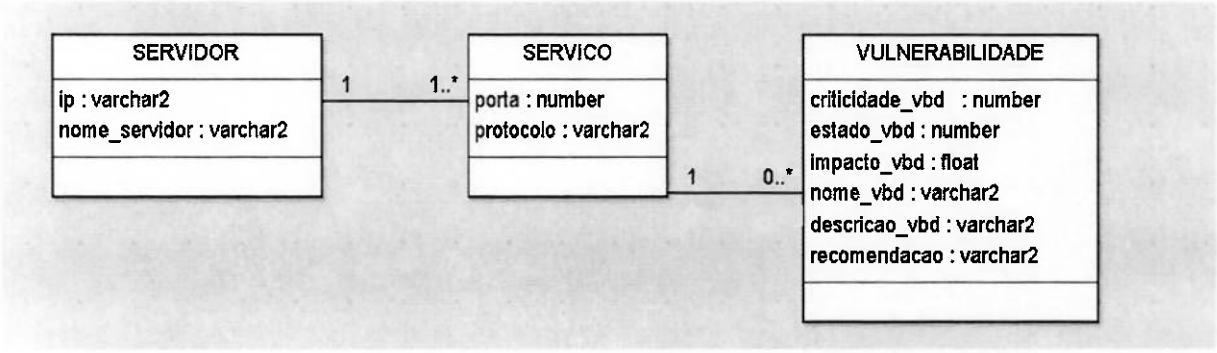

Figura 2: Aplicação proposta como modelo de testes para este trabalho.
Fonte: Elaborado pela autora.

Na aplicação usada, um Servidor (uma máquina) pode ter vários serviços ativos e cada serviço pode ter várias vulnerabilidades. Esta dependência gera

importância para o controle de quais vulnerabilidades um serviço pode estar sujeito e a qual servidor este serviço pertence. O modelo ilustrado na Figura 2 é base tanto para os testes realizados em BDR como para BDOR.

Tendo em vista que BDOR oferece vários recursos diferentes dentro do paradigma de objetos, como o uso de referências e/ou composição para definir os relacionamentos entre objetos. O presente trabalho restringe-se especificamente ao modelo da aplicação apresentado na Figura 2, entretanto, procurando difundir e avaliar os recursos disponibilizados pelo BDOR optou-se pela criação de outros modelos para melhor ilustrar as diferentes formas que uma mesma aplicação pode ser implementada. Os demais modelos serão apresentados no decorrer desta pesquisa.

3.2 Configurações.

A Tabela 3 mostra as configurações do hardware, a especificação do sistema operacional, além do SGBD utilizados para a realização dos testes.

Componentes do Sistema e Hardware	
Máquina	Sony VAIO
Processador	Intel Core I5-2450M 250 GHz
HD	500 GB
Memória RAM	4 GB + 1 GB (dedicado)
Sistema Operacional	Windows 7 Home Premium
SGBD	Oracle 11g Express R2

Tabela 3: Componentes do Sistema e Hardware.
Fonte: Elaborado pela autora.

Como mostrado na Tabela 3, todos os testes foram realizados na máquina SONY VAIO. Na qual foram acionados quatro núcleos do processador, a fim de aumentar o desempenho e diminuir os riscos de sobrecarga no processamento. O espaço físico em máquina assim como o poder de processamento e memória RAM foi mais que suficiente para os testes propostos. A plataforma Windows se comportou bem para a instalação e configuração do SGBD escolhido.

Na instalação do SGBD Oracle 11g (em sua versão gratuita), inicialmente foram utilizados os parâmetros *default*, entretanto, em função dos testes com execução de um DML (*insert)* com inserção de milhares de registros, alguns parâmetros tiveram seus valores modificados. *Tablespace_maxsize (2MB)* – aumento da tablespace para 2MB, a fim de, evitar problemas de alocação de espaço

(estouro da tablespace), um erro que acarreta um desperdício de tempo imenso pelo fato de toda a transação ser abortada. *Resumable_timeout (0),* nos testes iniciais este parâmetro teve seu valor alterado para 10, ocasionando a suspensão da sessão por 10 segundos quando o estouro da tablespace, tempo que permitia a verificação do erro, no entanto, com o aumento da tablespace optou-se por desabilitar este recurso deixando o valor deste parâmetro igual à zero.

O ambiente de trabalho utilizado para acessar o BD inicialmente foi a ferramenta gráfica Oracle SQL Developer. Contudo, para os testes, esta ferramenta se comportou de forma limitada, inseriu no máximo 50.000 registros por carga e retornou um máximo de 500 registros por consulta. Logo, optou-se pela utilização do processador de linha de comando Oracle *SQL*Plus,* que não demonstrou qualquer limitação durante os testes.

A escolha do SGBD Oracle foi feita devido ao suporte de geração para base de dados, tanto relacional quanto objeto-relacional. Cabe aqui ressaltar, que este trabalho avalia apenas parte do que está especificado para o tratamento de objetos em banco de dados de acordo com especificação SQL. Apesar de restrita, entende-se que a avaliação realizada traz, ainda sim, resultados importantes para compreender um pouco mais sobre o comportamento real de alguns elementos de BDOR.

3.3 Especificação e Organização dos Testes.

Com o objetivo de testar a aplicação como um todo, foram determinados os aspectos a serem testados e os detalhes da abordagem. Como o controle e o tempo de resposta das operações são feitos pelo próprio SGBD, foi adotada a nomenclatura HH: MM: SS: MMM, sendo: HH – horas, MM minutos, SS, segundos e MMM – milissegundos. Para melhor visualizar o tempo de execução retornado em cada teste executado. As medidas são baseadas no tempo de resposta definido pelo intervalo entre a requisição (*start*) e a resposta do sistema (retorno da operação pelo SGBD). Os dados carregados na tabela são reais, contudo, para efeito de ilustração e conhecimento das características dos dados utilizados, em cada comando serão apresentados alguns registros utilizados apenas para os testes iniciais, isto é, o estado inicial do banco: criação da tabela, inserção, exclusão e extração de apenas um registro. Ao iniciar os testes de desempenho, com volume superior a 4KB, ou seja, mais de 1 registro por tabela, a amostragem real de todos os dados ocuparia

espaço desnecessário, além de violar o sigilo de dados de terceiros, sendo então estes dados omitidos no trabalho. A Figura 3 mostra as etapas que foram executadas detalhando melhor a ordem dos testes realizados.

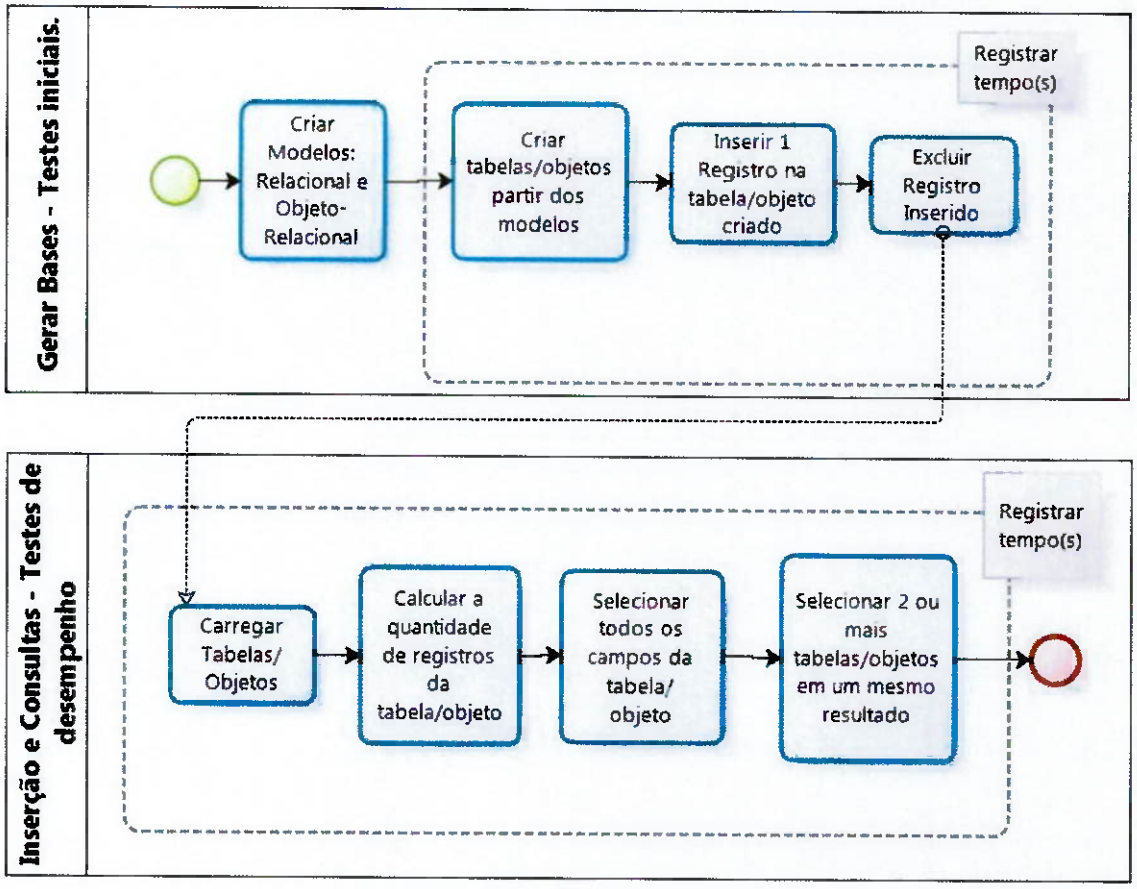

Figura 3: Procedimentos dos Testes: criação, inserção e consultas para avaliação de desempenho.
Fonte: Elaborado pela autora.

A Figura 3 ilustra as etapas seguidas para que os testes fossem feitos de forma organizada e estruturada, sendo registrados os tempos de todos os testes. Assim, no quadro "Gerar Bases – Testes iniciais" – foram criados os modelos de dados que posteriormente foram implementados no SGBD. Nesta etapa, foi inserido um único registro que após é excluído (esta exclusão se dá apenas para a limpeza do banco para o carregamento da massa de testes real), finalizando os testes iniciais de manipulação. Ainda na Figura 3, no quadro "Inserção Consultas – Testes de desempenho" foram feitos testes de carga, carregando o BD através de um arquivo SQL (script) contendo dados reais não viciados avaliando o comportamento da aplicação. O próximo passo foram os testes de consulta. Primeiro avaliou-se a função "count", para conhecer o número de linhas existentes na tabela/objeto. Segundo, avaliou-se o tempo de execução para recuperar todos os campos da

tabela/objeto. Ambas as consultas foram feitas sobre apenas uma tabela/objeto selecionado. Completando os testes de desempenho, foram feitas consultas através da junção das tabelas, avaliando:

a) Especificação de condição (ou seja, considerando que apenas parte da tabela/objeto satisfaz a consulta);
b) Mais de uma tabela/objeto;
c) Referência e DEREF.

É importante salientar que em cada passo o buffer foi zerado e o sistema totalmente encerrado, a fim de evitar dados mascarados.

3.4 *Descrição dos Testes.*

Nesta seção mostra-se a fase de implementação da aplicação em BDR e BDOR para o SGBD Oracle.

3.4.1 BDR.

A partir da estrutura da aplicação ilustrada na Figura 2, foi definido o modelo relacional da aplicação utilizada nos testes. O modelo para BDR é ilustrado na Figura 4.

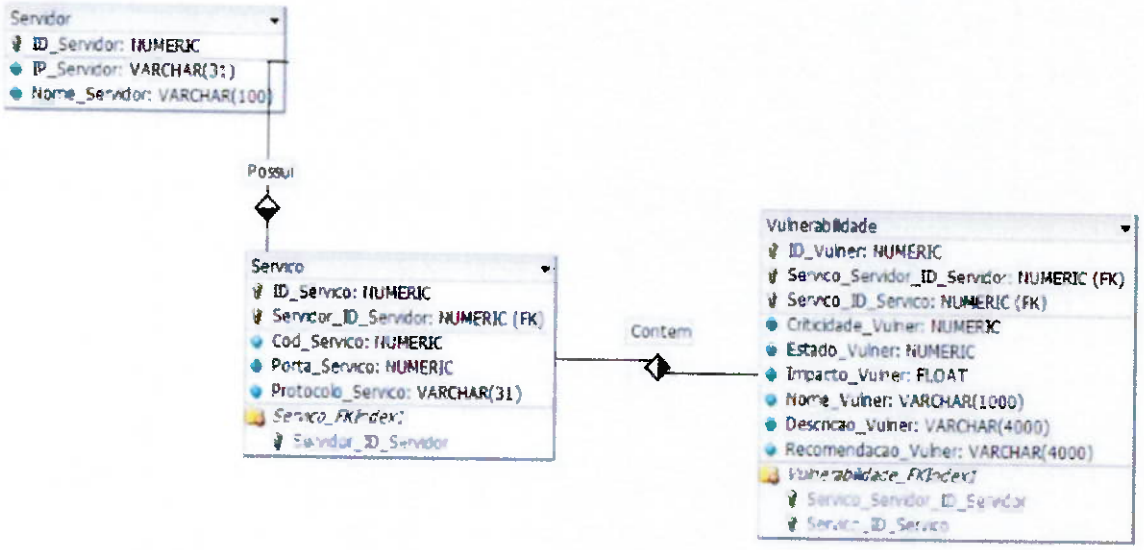

Figura 4: BDR Modelo de Dados Relacional.
Fonte: Elaborado pela autora.

Conforme pode ser obervado na Figura 4 os relacionamentos entre as tabelas, Servidor e Serviço e entre este e Vulnerabilidade são 1:N. Não foram avaliados relacionamentos N:N.

A seguir é mostrada na Tabela 4 a sintaxe utilizada para a criação das três tabelas representadas pelo modelo relacional da aplicação.

	Tabela BDR_Servidor
Oracle	CREATE TABLE BDR_Servidor (ID_Servidor NUMBER(15,0) NOT NULL, IP_Servidor VARCHAR2(30) NOT NULL, Nome_Servidor VARCHAR2(100), constraint PKBRD_Servidor PRIMARY KEY (ID_Servidor));
	Tabela BDR_Serviço
Oracle	CREATE TABLE BDR_Servico (ID_Servico NUMBER(15,0) NOT NULL, ID_Servidor NUMBER(15,0) NOT NULL, Porta_Servico NUMBER(38,0), Protocolo_Servico VARCHAR2(30 BYTE), constraint PKBDR_Servico PRIMARY KEY (ID_Servico), constraint FKBDR_Servidor foreign KEY (ID_Servidor) references BDR_Servidor (ID_Servidor));
	Tabela BDR_Vulnerabilidade
Oracle	CREATE TABLE BDR_Vulnerabilidade (ID_Vulner NUMBER(15,0) NOT NULL, ID_Servico NUMBER(15,0) NOT NULL, ID_Servidor NUMBER(15,0) NOT NULL, Criticidade_Vulner NUMBER(38,0), Estado_Vulner NUMBER(38,0), Impacto_Vulner FLOAT(63), Nome_Vulner VARCHAR2(1000 BYTE), Descricao_Vulner VARCHAR2(4000 BYTE), Recomendacao_Vulner VARCHAR2(4000 BYTE), constraint PKBDR_Vulnerabilidade PRIMARY KEY (ID_Vulner), constraint FKBDR_Servico foreign KEY (ID_Servico) references BDR_Servico (ID_Servico), constraint FKBDR_Servidor_Vulner foreign KEY (ID_Servidor) references BDR_Servidor (ID_Servidor));

Tabela 4: Implementação da aplicação em BDR.
Fonte: Elaborado pela autora.

Para os testes, as três tabelas foram criadas no SGBD Oracle (Tabela 4) e carregadas com tamanhos distintos: Servidor com 2,40MB, Serviço com 2,69MB e

Vulnerabilidade com 57,4MB. Para esses tamanhos a quantidade de linhas por tabela foi de 30100 (2,23MB), 45000 (2,69MB) e 124.712(57,4MB) respectivamente. Como os conceitos de BDR já são bem definidos e, de fácil manipulação e compreensão. Os códigos descritos para a criação das tabelas dispensam maiores detalhes.

3.4.2 BDOR.

Assim, como dito anteriormente, foram feitos mais de um modelo a partir da estrutura da aplicação definida na Figura 2. Para que fosse possível testar diferentes estruturas e tipos disponibilizados em BDOR.

1. BDOR1: Tabelas aninhadas MULTISET- Composição/Agregação.

Utilizando o conceito de coleção (ou atributos compostos com valor de tupla) é permitida a criação de coleções aninhadas, ou seja, uma estrutura multivalorada (BAPTISTA, 2009).

A Figura 5 apresenta o modelo de classes para criação de tabelas aninhadas sobre aplicação proposta.

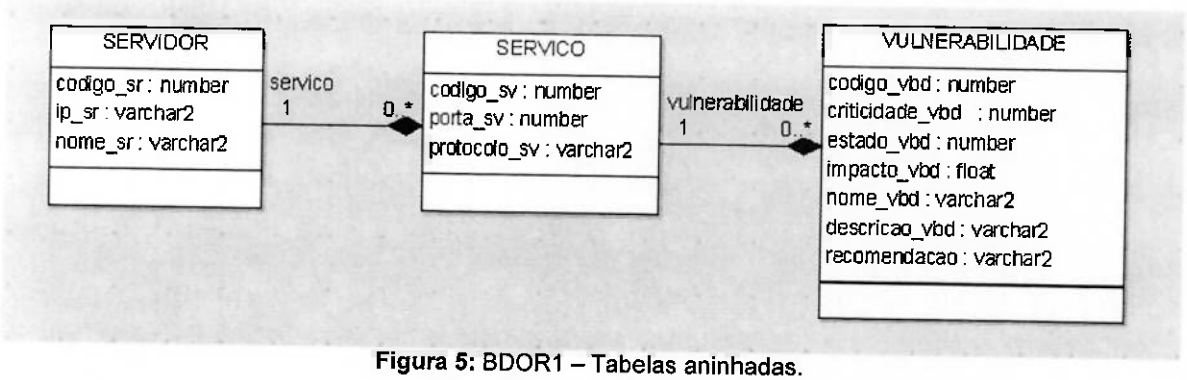

Figura 5: BDOR1 – Tabelas aninhadas.
Fonte: Elaborado pela autora.

Para a implementação do conceito de tabelas aninhadas, utilizou-se a função Nested Table do Oracle. A tabela aninhada será aqui representada por BDOR1, apresentando o conceito de coleções multidimensional, ou Multiset de objetos.

A seguir são apresentadas as queries para criação das tabelas:

- **Tipo de objeto Vulnerabilidade1_TY.**

```
CREATE TYPE VULNERABILIDADE1_TY AS OBJECT
(CODIGO_VBD        NUMBER(15),
CRITICIDADE_VBD  NUMBER(38),
ESTADO_VBD        NUMBER(38),
IMPACTO_VBD       FLOAT(63),
NOME_VBD          VARCHAR2(4000),
DESCRICAO_VBD     VARCHAR2(4000),
RECOMENDACAO_VBD VARCHAR2(4000));

CREATE TYPE NT_VULNERABILIDADE1_TY AS TABLE OF VULNERABILIDADE1_TY;
```

Como não é possível ocorrer uma inserção de dados em um tipo de objeto (tipo abstrato de dados – TAD), pois, assim como numa classe abstrata, este descreve os dados, mas, não os armazena. Para armazenar dados é necessária a criação de uma tabela a partir de um tipo de objeto. Logo, criou-se a tabela NT_Vulneravilidade1_TY a partir do tipo de objeto Vulnerabilidade1_TY, ou seja, NT_Vulneravilidade1_TY irá armazenar dados com a estrutura do tipo Vulnerabilidade1_TY. E o mesmo ocorre para a tabela NT_Servico_TY que possui estrutura do tipo Serviço_TY, cujo código SQL para a criação é apresentado a seguir:

- **Tipo de objeto Servico_TY.**

```
CREATE TYPE SERVICO_TY AS OBJECT
(CODIGO_SV    NUMBER(15),
 PORTA_SV     NUMBER(38),
 PROTOCOLO_SV VARCHAR2(30),
 VULNERS_SV   NT_VULNERABILIDADE1_TY);

CREATE TYPE NT_SERVICO_TY AS TABLE OF SERVICO_TY;
```

- **Tabela Servidor_OR_TAB.**

```
CREATE TABLE SERVIDOR_OR_TAB
(CODIGO_SR   NUMBER(15),
 IP_SR       VARCHAR2(30),
 NOME_SR     VARCHAR2(100),
 SERVICOS_SR NT_SERVICO_TY)
 NESTED TABLE SERVICOS_SR STORE AS NT_SERVICOS_TAB
(NESTED TABLE VULNERS_SV STORE AS NT_VULNERABILIDADE1_TAB);
```

A tabela Vulnerabilidade foi mapeada como um tipo de objeto e compõem a tabela Serviço. Por sua vez, Serviço é um tipo de objeto que compõem a tabela Servidor representando em suas formas de implementação atributos compostos ou coleções. Portanto, Servidor (Servidor_OR_TAB) é uma tabela de objetos onde cada valor é na realidade uma coleção de valores aninhados dentro das linhas que as contêm. Ou, em termos de SQL, um objeto complexo representado por coleções multidimensionais e em Oracle uma Nested table de Nested table type.

Aplicando estes conceitos, segue a ilustração da implementação de BDOR1 no SGBD Oracle, para a aplicação proposta.

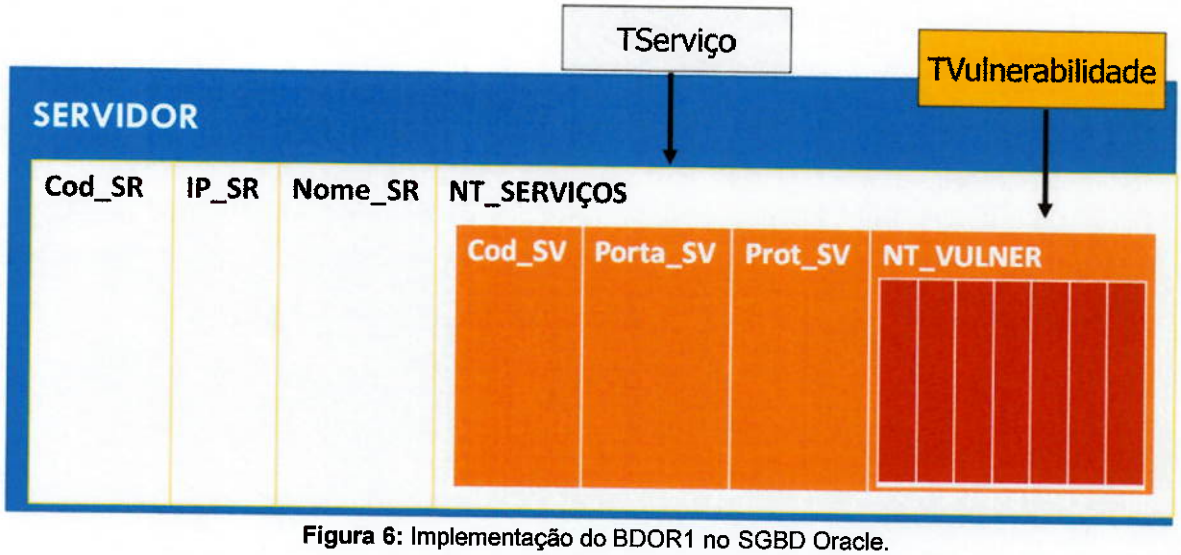

Figura 6: Implementação do BDOR1 no SGBD Oracle.

Fonte: Elaborado pela autora.

– **Inserindo dados:**

```
INSERT INTO SERVIDOR_OR_TAB
VALUES(1, '191.162.1.2', 'SERVIDOR_TESTE', NT_SERVICO_TY(
SERVICO_TY(1,5000,'FTP',NT_VULNERABILIDADE1_TY(
VULNERABILIDADE1_TY(1,2,2,1.0,'TESTE ERRO FTP','CALCULANDO ERRO','RE-TESTE'),
VULNERABILIDADE1_TY(2,1,2,3.0,'TESTE ERRO2 FTP','CALCULANDO ERRO2','RE-TESTE2'),
VULNERABILIDADE1_TY(3,4,2,5.0,'TESTE ERRO3 FTP','CALCULANDO ERRO3','RE-TESTE3'))),
SERVICO_TY(2,5000,'TCP/IP',NT_VULNERABILIDADE1_TY(
VULNERABILIDADE1_TY(1,2,2,1.0,'TESTE ERRO TCP','CALCULANDO ERRO','RE-TESTE
TCCP'),
VULNERABILIDADE1_TY(2,4,2,5.0,'TESTE ERRO3 TCP','CALCULANDO ERRO3','RE-TESTE
TCP/IP'))));
```

Em BDOR o método construtor de um tipo definido permite apenas a inserção de uma linha por vez. Mas, em uma Nested Table (coleção), é possível a inserção

em todas as tabelas de uma única vez, sendo até mais de um registro por tabela de acordo com a estrutura e associação. Neste caso, respeitando o relacionamento de 1:N, foram inseridos:

- 1 registro na Tabela Servidor;
- 2 registros na Tabela Serviço;
- 5 registros na Tabela Vulnerabilidade.

– **Operações em coleções:**

a. Consulta com resultado aninhado

```
SELECT A.*
 FROM SERVIDOR_OR_TAB A;
```

Resultado:

```
1     191.162.1.2 servidor_teste
      ADRIANA.NT_SERVICO_TY(ADRIANA.SERVICO_TY(1,5000,'ftp',
      ADRIANA.NT_VULNERABILIDADE1_TY(ADRIANA.VULNERABILIDADE1_TY(1,2,2,1,'t
este erro ftp','calculando erro','re-teste'),
      ADRIANA.VULNERABILIDADE1_TY(2,1,2,3,'teste erro2 ftp','calculando
erro2','re-teste2'),
      ADRIANA.VULNERABILIDADE1_TY(3,4,2,5,'teste erro3 ftp','calculando
erro3','re-teste3'))),
      ADRIANA.SERVICO_TY(2,5000,'tcp/ip',
      ADRIANA.NT_VULNERABILIDADE1_TY(ADRIANA.VULNERABILIDADE1_TY(1,2,2,1,'t
este erro tcp','calculando erro','re-teste tccp'),
      ADRIANA.VULNERABILIDADE1_TY(2,4,2,5,'teste erro3 tcp','calculando
erro3','re-teste tcp/ip'))))
```

Como pode ser visto no quadro anterior, a resposta da pesquisa são os campos das três tabelas de forma aninhada. O resultado é mais natural, pois, a consulta é feita a tabela de objetos Servidor e, como esta é composta pelo objeto Serviço e este por Vulnerabilidades, todas essas informações são apresentadas em uma única consulta.

b. Consulta com resultado desaninhado.

Este tipo de consulta mostra que é possível consultar não apenas o todo, mas também, a parte de um objeto composto. A query a seguir resulta apenas as linhas de serviços que possuem vulnerabilidades.

```
SELECT E.*
FROM SERVIDOR_OR_TAB B,
TABLE(B.SERVICOS_SR)E;
```

Resultado:

```
1     5000  ftp
      ADRIANA.NT_VULNERABILIDADE1_TY(ADRIANA.VULNERABILIDADE1_TY(1,2,2,1,
'teste erro ftp','calculando erro','re-teste'),
      ADRIANA.VULNERABILIDADE1_TY(2,1,2,3,'teste erro2 ftp','calculando
erro2','re-teste2'),
      ADRIANA.VULNERABILIDADE1_TY(3,4,2,5,'teste erro3 ftp','calculando
erro3','re-teste3'))
2     5000  tcp/ip
      ADRIANA.NT_VULNERABILIDADE1_TY(ADRIANA.VULNERABILIDADE1_TY(1,2,2,1,
'teste erro tcp','calculando erro','re-teste tccp'),
      ADRIANA.VULNERABILIDADE1_TY(2,4,2,5,'teste erro3 tcp','calculando
erro3','re-teste tcp/ip'))
```

2. *BDOR2. Criando tabelas de objetos com REF e Nested table.*

Neste item a mesma aplicação é modelada e implementada de forma diferente, como pode ser visto na Figura 7 que ilustra o modelo baseado na Figura 2.

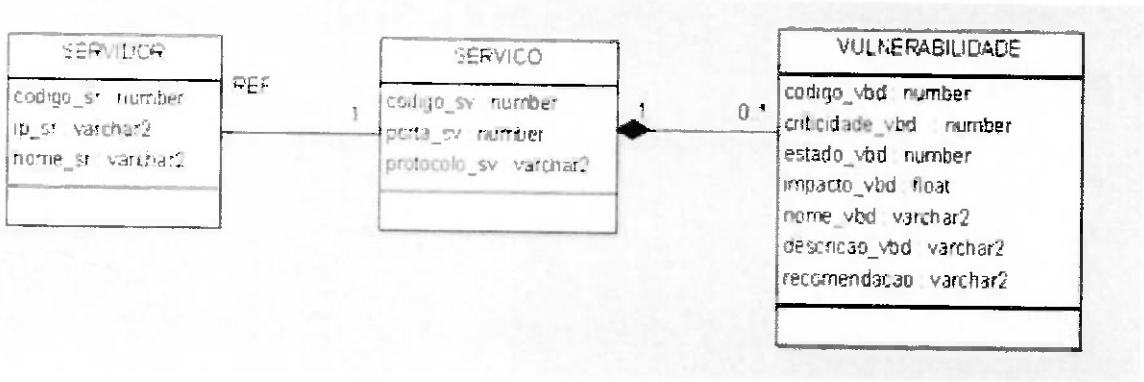

Figura 7: BDOR2 – Composição, associação por REF.
Fonte: Elaborado pela autora.

Na Figura 7, o relacionamento entre Servidor e Serviço é implementado no BD através de REF (referência) e a composição entre Servidor e Vulnerabilidade é feito através do conceito de Nested Table, gerando o conceito de tabela aninhada. Como várias tabelas podem ser de um mesmo tipo, para este modelo utilizou-se o conceito de reutilização na criação da tabela Vulnerabilidade. Logo, a tabela é a mesma apresentada no item 1 – e pode ser vista em BDOR1 desta seção.

A seguir os códigos SQL para a criação das demais tabelas.

- **Tipo de objeto Servidor_TY**

```
CREATE TYPE SERVIDOR_TY AS OBJECT
(CODIGO_SR   NUMBER(15)
 IP_SR       VARCHAR2(30),
 NOME_SR     VARCHAR2(100));

CREATE TABLE SERVIDOR_OR_TAB_TY OF SERVIDOR_TY
```

- **Tabela de objetos com REF e Nested Table.**

```
CREATE TABLE SERVICO_OR_TAB
(CODIGO_S     NUMBER(15),
 SERVIDOR_REF REF SERVIDOR_TY,
 PORTA_S      NUMBER(38),
 PROTOCOLO_S  VARCHAR2(30),
 VULNERS_S    NT_VULNERABILIDADE1_TY)

NESTED TABLE VULNERS_S STORE AS NT_VULNERABILIDADE2_TAB;
```

Explicação: diferente do conceito dado em BDOR1, em BDOR2 Servidor é mapeado como um tipo de objeto (precisamente uma tabela tipada), que se relaciona com Serviço através de REF. Já Serviço está mapeado como uma tabela de objetos a qual é composta por Vulnerabilidades, representada como Nested Table. A Figura 8 ilustra a forma como as tabelas foram implementadas dentro do SGBD Oracle.

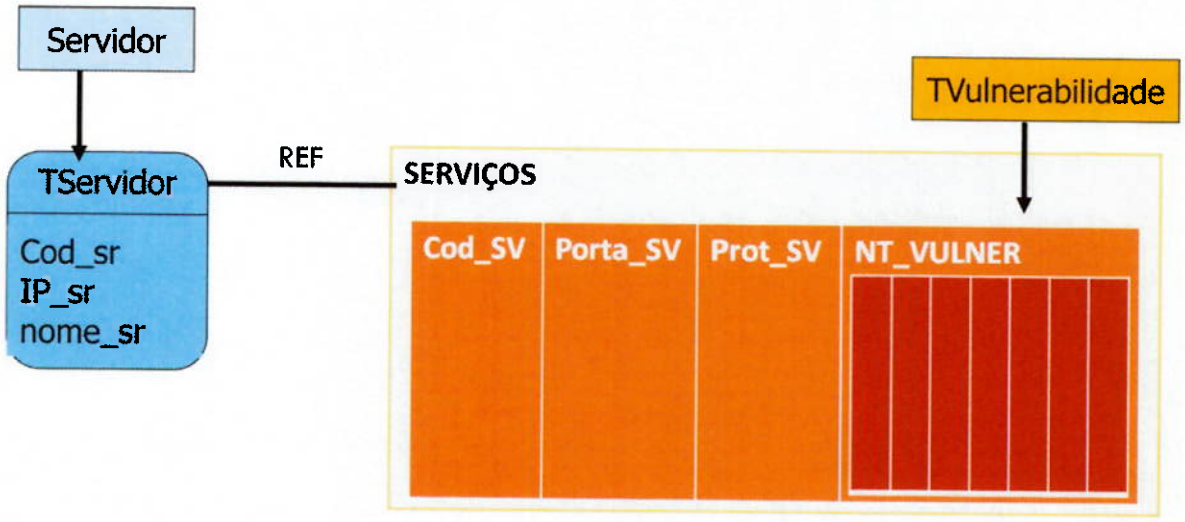

Figura 8: Implementação do BDOR2 no SGBD Oracle.

Fonte: Elaborado pela autora.

- **Inserindo dados no objeto Servidor:**

```
INSERT INTO SERVIDOR_OR_TAB_TY
VALUES(1, '161.192.3.5', 'SERVIDOR TESTE2');
```

- **Inserindo dados na tabela de objetos Serviço com atributo Vulnerabilidade.**

```
INSERT INTO SERVICO_OR_TAB
VALUES(1,
(SELECT REF(A) FROM SERVIDOR_OR_TAB_TY A
WHERE A.CODIGO_SR=1),
50000, 'DHCP',
NT_VULNERABILIDADE1_TY(VULNERABILIDADE1_TY(1,2,2,1.0,'TESTE ERRO FTP' ,
'CALCULANDO ERRO','RE-TESTE'),
VULNERABILIDADE1_TY(2,1,2,3.0,'TESTE ERRO2 FTP','CALCULANDO ERRO2','RE-
TESTE2')));
```

Para a representação do relacionamento por REF, primeiramente são inseridos dados no objeto Servidor e posteriormente em Serviço. Uma vez que só se referência uma tabela que tenha um OID definido, é importante observar que para inserção de dados na tabela Serviço é necessária uma consulta ao objeto Servidor para o retorno do registro que será referenciado. Por outro lado, a inclusão em Vulnerabilidades pode ser feita exatamente como no modelo anterior (BDOR1) e com quantos registros forem necessários.

- **Operações:**
  a. **Consultar todos os campos do objeto Serviço, inclusive a REF.**

```
SELECT * FROM SERVICO_OR_TAB
```

Resultado:

```
1     ADRIANA.SERVIDOR_TY(1,'161.192.3.5','servidor teste2')
      50000 dhcp
      ADRIANA.NT_VULNERABILIDADE1_TY(ADRIANA.VULNERABILIDADE1_TY(1,2,2,1,'t
este erro ftp','calculando erro','re-teste'),
      ADRIANA.VULNERABILIDADE1_TY(2,1,2,3,'teste erro2 ftp','calculando
erro2','re-teste2'))
```

Como a consulta é feita na tabela de objetos Serviço e esta é composta por Vulnerabilidades e REF para Servidor. O resultado anterior apresenta de forma direta o objeto referenciado, isto é, o valor de Servidor relacionado ao Serviço

consultado, assim como a informação de Serviço e Vulnerabilidade, estes últimos de forma aninhada por ser um Nested Table.

b. Consultar apenas o conteúdo da REF a partir da tabela de referência

```
SELECT DEREF(A.SERVIDOR_REF)
FROM SERVICO_OR_TAB A
WHERE A.CODIGO_T=1;
```

Resultado:

```
ADRIANA.SERVIDOR_TY(1,'161.192.3.5','servidor teste2')
```

No resultado anterior, o operador DEREF "desfaz" o REF retornando apenas o objeto referenciado pela coluna do tipo REF, ou seja, apenas o valor contido na tabela Servidor para o Serviço referenciado.

3. BDOR3. Tabelas de Objeto com tipo de atributo cujo domínio é um tipo de objeto e associação por REF.

Neste item a aplicação é representada através do conceito de agregação. A Figura 9 ilustra o conceito de agregação em BDOR para a aplicação apresentada na Figura 2.

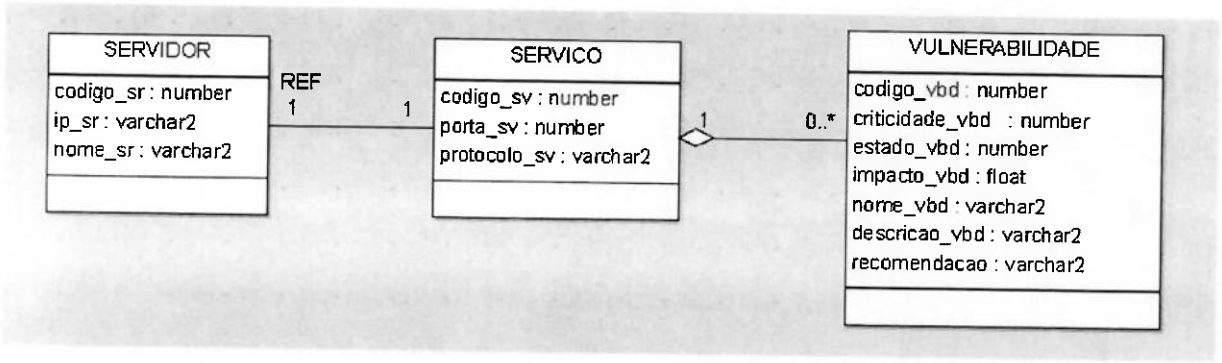

Figura 9: BDOR3. Agregação, associação por REF.
Fonte: Elaborado pela autora.

No modelo da Figura 9, a associação entre Servidor e Serviço é implementada através de REF. E Serviço tem o conjunto de objetos Vulnerabilidade. A diferença em relação ao caso do BDOR2, é que aqui não se usa o tipo *Nested Table*, pois, a intenção é representar uma agregação e não uma composição.

A seguir são listados os códigos SQL que geraram as tabelas:

- **Tipo de objeto Vulnerabilidade**

```
CREATE TYPE VULNERABILIDADE_TYP AS OBJECT
(VCODIGO      NUMBER(15),
 VCRITICIDADE NUMBER(38),
 VESTADO      NUMBER(38),
 VIMPACTO     FLOAT(63),
 VNOME        VARCHAR2(4000),
 VDESCRICAO   VARCHAR2(4000),
 VRECOMENDACAO VARCHAR2(4000));
```

- **Tipo de objeto Serviço com atributo do tipo Vulnerabilidade.**

A tabela Serviço irá agregar o objeto Vulnerabilidade definido como UDT no quadro anterior (observe que não é uma referência, e sim um objeto).

```
CREATE TYPE SERVICO_TYP AS OBJECT
(SCODIGO     NUMBER(15),
 SPORTA      NUMBER(38),
 SPROTOCOLO  VARCHAR2(30),
 SVULNERABILIDADE VULNERABILIDADE_TYP);

CREATE TABLE TB_SERVICO_TYP OF SERVICO_TYP
(SCODIGO PRIMARY KEY);
```

- **Tabela Servidor com REF para Serviço.**

```
CREATE TABLE TB_SERVIDOR
(R_CODIGO     NUMBER(15)
 R_IP         VARCHAR2(30),
 R_NOME       VARCHAR2(100),
 R_SERVICO_REF REF SERVICO_TYP);
```

A tabela Servidor faz referência ao tipo Serviço_TYP, que define a estrutura da tabela TB_Serviço_TYP, que por sua vez agrega o atributo multivalorado Vulnerabilidade do tipo Vulnerabilidade_TYP. A seguir é apresentada a implementação das tabelas no SGBD Oracle:

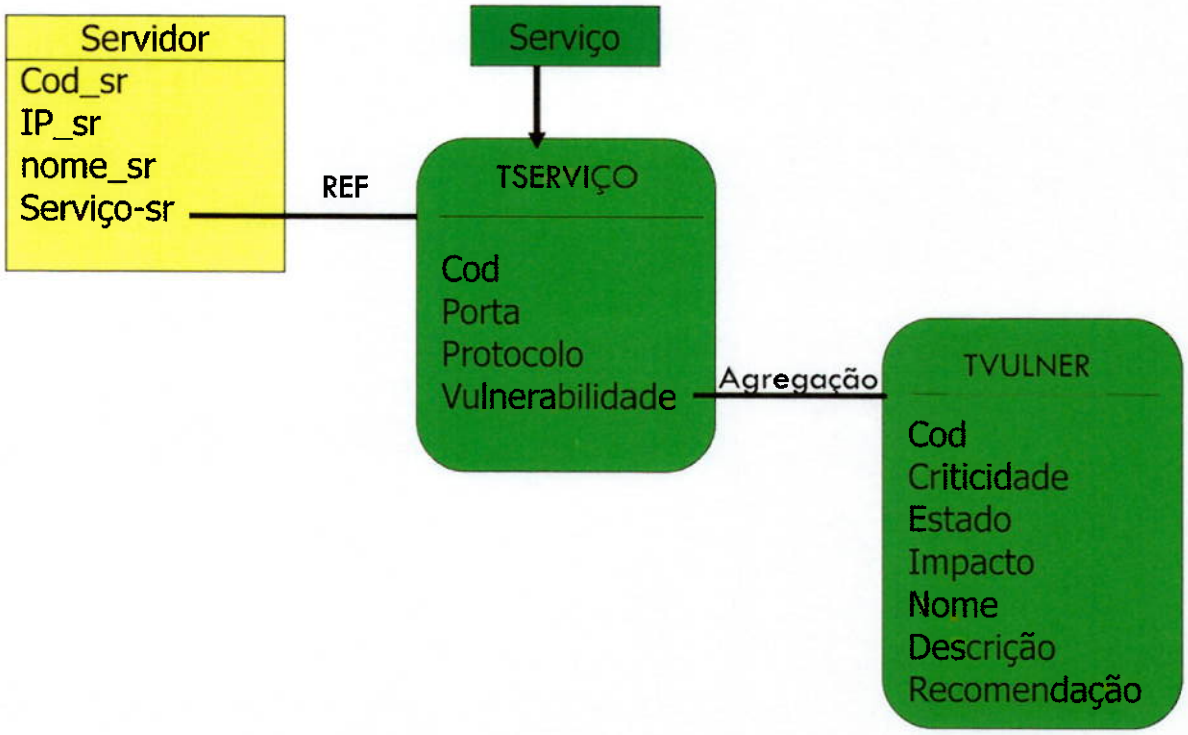

Figura 10: Implementação do BDOR3 no SGBD Oracle.
Fonte: Elaborado pela autora.

– **Inserindo dados no objeto Serviço.**

```
INSERT INTO TB_SERVICO_TYP VALUES(SERVICO_TYP(1, 15, 'FTP',
VULNERABILIDADE_TYP(4,2,2,1.0,'TESTE ERRO FTP','CALCULANDO'));
```

Quando se aloca um atributo de um tipo definido (UDT) a uma tabela, é permitida apenas uma inserção para aquele tipo. Logo, neste modelo a associação entre Serviço e Vulnerabilidade passa a ser de 1:1. Em outras palavras, um atributo pode conter apenas um objeto. Caso seja necessário representar na base que um Serviço agrega mais de uma Vulnerabilidade, não se deve optar por este tipo de implementação.

No código apresentado a seguir pode-se observar que ao instanciar um Servidor (que corresponde a inclusão de um elemento na tabela TB_SERVIDOR), para incluir à referência ao Serviço, uma consulta é submetida ao objeto TB_SEVICO_TPY. Isso é feito, pois referências são feitas aos OID de objetos, que é o resultado da consulta. Aqui vale ressaltar que a associação foi estabelecida de forma mais natural, sem a necessidade de repetir valores e definir restrições sobre atributos, como seria o caso em BDR.

- **Inserindo dados no objeto Servidor.**

```
INSERT INTO TB_SERVIDOR
VALUES(1, '169.155.2.3', 'SERVIDOR_ESQUEMA_2',
(SELECT REF(D) FROM TB_SERVICO_TYP D
WHERE D.SCODIGO=1));
```

- **Operações:**

  a. **Consultar todos os campos da tabela Servidor, inclusive a REF.**

  SELECT * FROM TB_SERVIDOR;

Resultado:

```
1     169.155.2.3 servidor_esquema_2
      ADRIANA.SERVICO_TYP(1,15,'ftp',ADRIANA.VULNERABILIDADE_TYP(4,2,2,1,'t
este erro ftp','calculando erro','re-teste3'))
```

A consulta resulta em novos dados aninhados, ou seja, todos os campos das três tabelas.

  b. **Consultar apenas a REF a partir da tabela de referência.**

  SELECT DEREF(A.R_SERVICO_REF)
  FROM TB_SERVIDOR A;

Resultado:

```
ADRIANA.SERVICO_TYP(1,15,'ftp',ADRIANA.VULNERABILIDADE_TYP(4,2,2,1,'teste
erro ftp','calculando erro','re-teste3'))
```

No resultado do quadro anterior, assim como em BDOR2 o operador DEREF "desfaz" o REF retornando apenas o objeto referênciado pela coluna do tipo REF, mas, neste caso o retorno é o objeto Serviço com a Vulnerabilidade agregada.

4. BDOR4. Tabelas Tipadas com associação através de REFs.

A peculiaridade em trabalhar com REFs é que apenas tabelas baseadas em tipos podem ser referenciadas, por isso, assim como nos itens anteriores em que se utilizou a REF, neste item todas as tabelas são criadas com o conceito de UDTs. Outro detalhe é que a expressão REF corresponde a um ponteiro, objetiva acesso mais rápido e não regra de integridade, logo, é apresentada a possibilidade de utilizar PK (*Primary Key)* em tabelas tipadas como forma de assegurar que não haverá repetições de tuplas. Vale ressaltar que não é a coluna REF a qual receberá a PK, mas sim, um campo específico diferente do referenciado.

No modelo da Figura 11 baseado na Figura 2, todos os objetos se relacionam através de REFs.

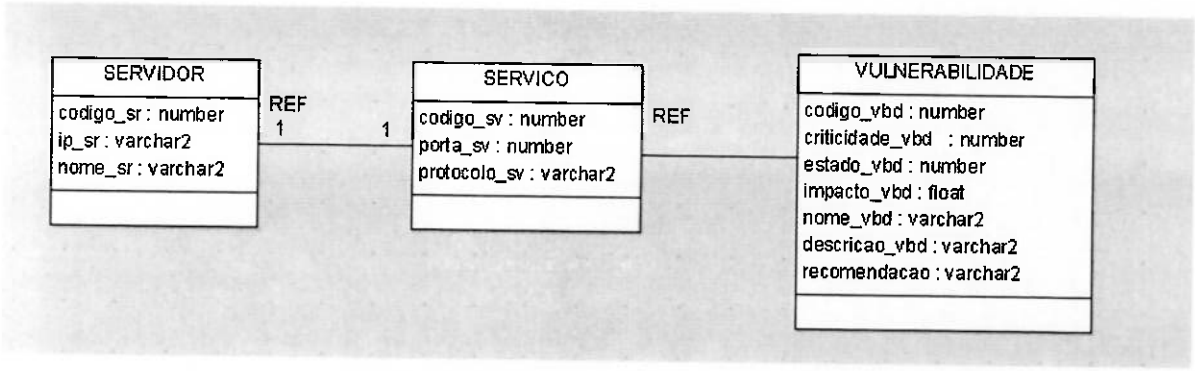

Figura 11: BDOR4 – Associação por REF.
Fonte: Elaborado pela autora

A seguir é apresentado o código SQL para implementação das tabelas no SGBD Oracle.

- **Tipo de objeto Vulnerabilidade:**

```
CREATE TYPE VULNER_TYP AS OBJECT
(VCOD      NUMBER(15),
 VCRITIC   NUMBER(38),
 VEST      NUMBER(38),
 VIMPAC    FLOAT(63),
 VNM       VARCHAR2(4000),
 VDESC     VARCHAR2(4000),
 VRECOMEND VARCHAR2(4000));
```

- **Tipo de objeto Serviço:**

```
CREATE TYPE SERVIC_TYP AS OBJECT
(SCOD     NUMBER(15),
 SPORT    NUMBER(38),
 SPROTC   VARCHAR2(30),
REF_VULNERS REF VULNER_TYP)
```

- **Tipo de objeto Servidor:**

```
CREATE TYPE SEVIDOR_TYP AS OBJECT
(RCOD     NUMBER(15),
 RIP      VARCHAR2(30),
 RNM      VARCHAR2(100),
 REF_SERVICO REF SERVIC_TYP);
```

- **Tabela Vulnerabilidade com estrutura do Tipo VULNER_TYP:**

```
CREATE TABLE TVULNER OF VULNER_TYP
(VCOD PRIMARY KEY);
```

- **Tabela Serviço com estrutura do Tipo SERVIC_TYP:**

```
CREATE TABLE TSERVICO OF SERVIC_TYP
(SCOD PRIMARY KEY,
REF_VULNERS SCOPE IS TVULNER);
```

- **Tabela Servidor com estrutura do Tipo SERVIDOR_TYP:**

```
CREATE TABLE TSERVIDOR OF SEVIDOR_TYP
(RCOD PRIMARY KEY,
RIP NOT NULL,
RNM NOT NULL,
REF_SERVICO SCOPE IS TSERVICO);
```

Todas as tabelas foram criadas a partir de um tipo definido pelo usuário (UDT). As associações são representadas através de REFs com definição de escopo da referência (*SCOPE IS*) indicando a qual tabela ou tipo de objeto pertence à referência. A implementação da aplicação é ilustrada a seguir:

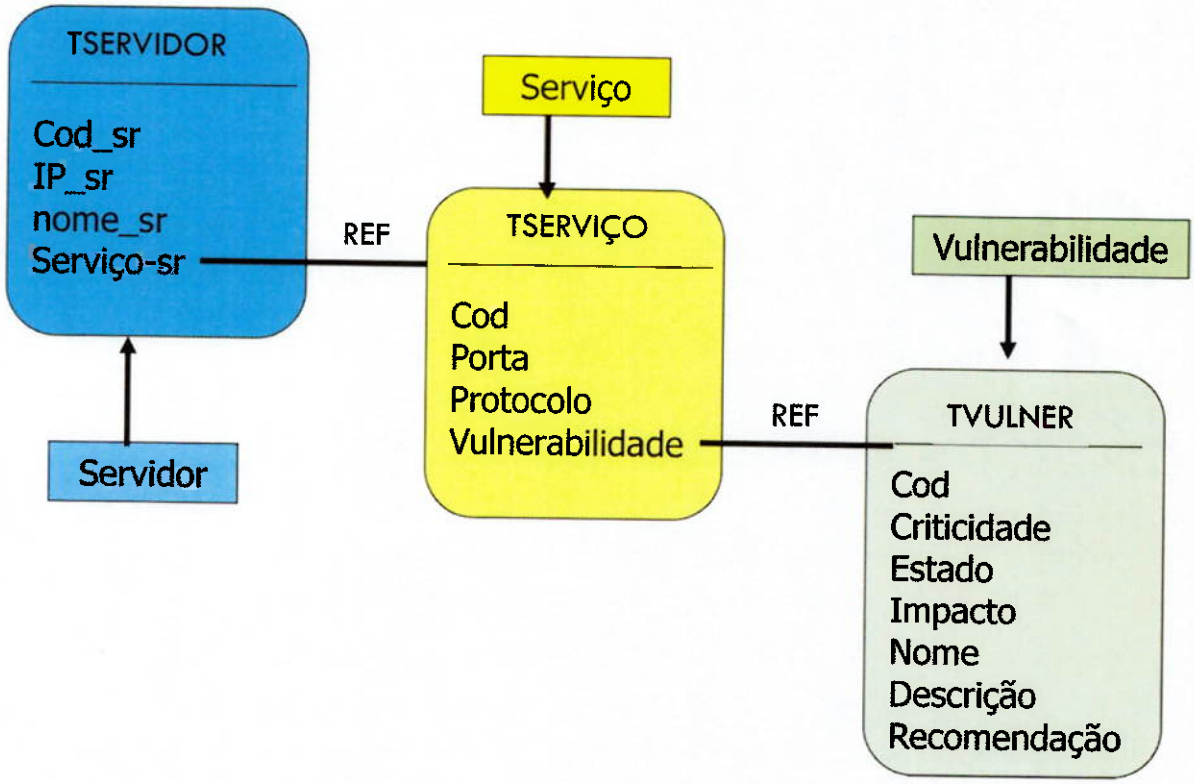

Figura 12: Implementação do BDOR4 no SGBD Oracle.
Fonte: Elaborado pela autora.

- **Inserindo dados na tabela Vulnerabilidade:**

```
INSERT INTO TVULNER VALUES(1,2,5,5.0,' TESTE ESQUEMA 3','TESTE ESQUEMA 3','TESTE');
```

- **Inserindo dados na tabela Serviço:**

```
INSERT INTO TSERVICO VALUES(1,5000,'DNS',
(SELECT REF(B) FROM TVULNER B
WHERE B.VCOD=1));
```

- **Inserindo dados na tabela Servidor:**

```
INSERT INTO TSERVIDOR
VALUES(1, '169.155.2.3', 'SERVIDOR_ESQUEMA_4',
(
SELECT REF(D) FROM TSERVICO D
WHERE D.SCOD=1
));
```

Como poder ser observado nos quadros anteriores, a ordem de inserção das tabelas é feita da última para a primeira tabela, ou seja, primeiro as tabelas filhas e por ultimo a mãe. Isso se deve ao fato de que para referenciar um tipo de objeto ou uma tabela, primeiro este deve existir.

- **Operações.**

  a. **Consulta de todos os campos das tabelas e as REFs.**

```
SELECT * FROM TSERVIDOR;
```

Resultado:

```
1     192.154.1.1 servidor
      ADRIANA.SERVIC_TYP(1,5000,'dns')
      ADRIANA.VULNER_TYP(1,2,5,5,' teste esquema 3','teste esquema
3','teste')
```

Efetuando a consulta através da tabela mãe, obtêm-se como resultado todos os campos das três tabelas. Para selecionar apenas os registros da tabela Serviço, Servidor, basta efetuar um select com DEREF como já vistos nos modelos anteriores. Mas, para selecionar apenas dados de Vulnerabilidade, é necessário apenas um select comum, ou seja, *Select * from TVULNER*.

3.5 *Resultado dos testes.*

Nesta seção são analisados os resultados das execuções dos experimentos. Para a apresentação e execução dos testes, cada conceito estudado foi separado, utilizando a nomenclatura BDORx, onde 'x' é um número de 1 à 4, representando os modelos criados na seção 3.4, como uma forma coerente para orientar e entender o processo de avaliação. Nesta etapa são medidas e documentadas as execuções do experimento e o comportamento do SGBD Oracle mediante a criação do BD e as DMLs de inserção e consultas.

Criação das tabelas: Um ponto interessante ao gerar um tipo de dado é que, ao serem executados no SGBDOR, os mesmos são compilados e não criados como as tabelas comuns e, por isso, não podem armazenar dados fisicamente. A compilação dos tipos levou apenas cerca de milésimos de segundos para serem concluídas. Já as tabelas com estrutura complexa chegaram ao máximo de 2 minutos para sua criação. A seguir é apresentada na Tabela 5 a referência de tempo entre a criação das tabelas/objetos nos BDORs: 1,2,3 e 4:

Create BDOR	
Banco.Tabela	**Tempo**
BDOR1.SERVIDOR_OR_TAB	00:00:01.166
BDOR1.VULNERABILIDADE1_TY	00:00:00.37
BDOR1.NT_VULNERABILIDADE1_TY	00:00:00.3
BDOR1.SERVICO_TY	00:00:00.165
BDOR1.NT_SERVICO_TY	00:00:00.08
BDOR2.SERVICO_OR_TAB	00:00:00.35
BDOR2.SERVIDOR_TY	00:00:01.186
BDOR2.SERVIDOR_OR_TAB_TY	00:00:00.53
BDOR2.VULNERABILIDADE_TYP	00:00:01.248
BDOR3.SERVICO_TYP	00:00:00.219
BDOR3.TB_SERVICO_TYP	00:00:00.89
BDOR3.TB_SEVIDOR	00:00:00.062
BDOR4.VULNER_TYP	00:00:00.827
BDOR4.TVULNER	00:00:02.044
BDOR4.SERVIC_TYP	00:00:00.125
BDOR4.TSERVICO	00:00:00.047
BDOR4.SEVIDOR_TYP	00:00:00.125
BDOR4.TSERVIDOR	00:00:00.093

Tabela 5: Criação das bases de teste – BDOR1, BDOR2, BDOR3 e BDOR4.
Fonte: Elaborado pela autora.

Números de Registro por Tabela/Objeto: Na Tabela 6 é mostrado o número de registros inserido em cada tipo de tabela/objeto utilizado para a realização dos testes, seu tamanho físico em Megabytes (MB), assim como o modelo de BD a qual pertence.

Modelo BD	Tabela	Nr° Registros	Tamanho Físico
BDOR1	SERVIDOR_OR_TAB	22372	163MB
BDOR1	NT_VULNERABILIDADE1_TY	246092	Não se aplica
BDOR1	NT_SERVICO_TY	111860	Não se aplica
BDOR2	SERVICO_OR_TAB	150500	175MB
BDOR2	SERVIDOR_OR_TAB_TY	30100	2,40MB
BDOR2	NT_VULNERABILIDADE1_TY	331100	Não se aplica
BDOR3	TB_SERVICO_TYP	50043	75,7MB
BDOR3	TB_SERVIDOR	150000	19,4MB
BDRO3	VULNERABILIDADE_TYP	50043	Não se aplica
BDOR4	TVULNER (Tabela Vulnerabilidade)	144903	67,2MB
BDOR4	TSERVICO (Tabela Serviço)	150000	7,16MB
BDOR4	TSERVIDOR (Tabela Servidor)	26986	4,53MB

Tabela 6: Número de registros nas tabelas/objetos.
Fonte: Elaborado pela autora.

O conceito de coleção ou Nested Table no Oracle permite manipular um número arbitrário de elementos, entretanto, a identificação de seu espaço físico de armazenamento é definida apenas a tabela mãe. Em BDOR1 e BDOR2 utilizou-se o conceito de Nested Table como Multiset de objetos e em BDOR3, Serviço é a tabela mãe que agrega Vulnerabilidade como atributo multivalorado. Já BDOR4 são tabelas tipadas que se relacionam por meio de REF. Logo, BDOR1, 2 e 3 possuem dependências implícitas que permitem o conhecimento do "Tamanho Físico" apenas das tabelas mães, por isso, o item "não se aplica" as demais.

Para entender melhor, na Tabela 7 são mostrados os resultados da inclusão de apenas 1 registro por tabela e a carga máxima obtida como referência de inclusão de dados para os testes de desempenho e persistência.

Inserts em BDOR				
Banco.Tabela	Nr° de Registros	Tempo	Nr° de Registros	Tempo
BDOR1. SERVIDOR_OR_TAB	22372	00:20:15.01	1	00:00:00.298
BDOR2. SERVIDOR_OR_TAB_TY	30100	00:01:40.00	1	00:00:00.13
BDOR2. SERVICO_OR_TAB	150500	00:07:20.00	1	00:00:00.06
BDOR3. TB_SERVICO_TYP	50043	00:02:44.00	1	00:00:00.78
BDOR3. TB_SERVIDOR	150000	00:04:05.00	1	00:00:00.31
BDOR4.TVULNER	144903	00:03:25.00	1	00:00:00.031
BDOR4.TSERVICO	150000	00:03:19.00	1	00:00:00.032
BDOR4.TSERVIDOR	26986	00:01:20.05	1	00:00:00.025

Tabela 7: Inserção de registros nos BDORs.
Fonte: Elaborado pela autora.

Cabe lembrar que em BDOR1 tem-se o objeto complexo Servidor composto pelo objeto Serviço e, este composto pelo objeto Vulnerabilidade através do uso de Nested Table o qual também pode ser entendido como uma coleção multidimensional ou tabela aninhada. No caso de BDOR2, utiliza-se uma REF para estabelecer uma ligação entre Servidor e Serviço, e entre Vulnerabilidade e Serviço há uma composição definida por Nested Table. Em BDOR3, Vulnerabilidade é um tipo de objeto que define um atributo em Serviço e este é uma coluna REF em Servidor. Já BDOR4, as três tabelas possuem ligação através de REF, com escopo de referência e chaves primárias.

Como o processo é transparente ao usuário, ao totalizar a inserção de BDOR1 o SGBD apresenta 22372 registros inseridos, mas, este resultado reflete apenas a quantidade de Servidores. Pois, a carga de dados foi feita diretamente nas três tabelas de objetos de forma praticamente simultânea. Na relação de 1:N, Serviço e Vulnerabilidade receberam mais de um registro por Servidor incluso gerando 111860 registros na tabela Serviço e 246092 registros na tabela Vulnerabilidade. Totalizando 380324 linhas inseridas ou 163MB de dados.

Em BDOR2 seguindo o mesmo conceito de BDOR1 para Serviço e Vulnerabilidade, o SGBD apresenta um total de 150500 registros inseridos, refletindo apenas a quantidade de Serviço, ao olhar para Vulnerabilidade foram inseridos mais 331100 registros, ou seja, 175 MB de dados ou 481600 linhas inseridas. Como

Servidor é independente, a amostragem de dados inseridos de 30100 foi apresentada de forma evidente.

BDOR3, apesar de não possui objetos compostos, possui agregação entre Serviço e Vulnerabilidade, ocasionando o mesmo mascaramento ao visualizar 19,4 MB como tamanho total da tabela Serviço, sendo assim, além dos 50043 registros inseridos, há também 50043 registros em Vulnerabilidades. E 150000 Servidores. Cabe lembrar que nesta forma de implementação, não foi possível manter o relacionamento de 1:N entre as tabelas Servidor e Vulnerabilidade, por isso há uma queda na quantidade de registros inseridos. Já BDOR4, não possui mascaramento algum, todos os registros inseridos são retornados pelo SGBD sem qualquer perda de dados. A seguir a Tabela 8 apresenta os valores reais inseridos em cada tabela/objeto.

Inserts em BDOR – Valores Reais				
Banco.Tabela	**Servidor**	**Serviço**	**Vulnerabilidade**	**Total**
BDOR1.	22372	111860	246092	380324
BDOR2.	30100	150500	331100	481600/ 511700
BDOR3.	150000	50043	50043	100086/ 250086
BDOR4	26986	150000	144903	321889

Tabela 8: Quantidade de registros reais existentes por tabela nos BDORs.
Fonte: Elaborado pela autora.

Retornando o total de linhas da tabela. A Tabela 9 mostra as consultas *"Select count"* para retorno do total de linhas encontrado em cada da tabela e os respectivos tempos decorridos.

Count(*) em BDOR			
Entidade	**Linhas retornadas**	**Tempo**	**Nr° Registros da Tabela**
BDOR1.SERVIDOR_OR_TAB	1	00:00:00.0	22372
BDOR2. SERVIDOR_OR_TAB_TY	1	00:00:00.07	30100
BDOR2.SERVICO_OR_TAB	1	00:00:00.32	331100
BDOR3.TB_SERVICO_TYP	1	00:00:00.00	50043
BDOR3.TB_SERVIDOR	1	00:00:00.00	150000
BDOR4.TVULNER	1	00:00:00.17	144903
BDOR4.TSERVICO	1	00:00:00.10	150000
BDOR4.TSERVIDOR	1	00:00:00.00	26986

Tabela 9: Testes com a função count.

Fonte: Elaborado pela autora.

Observando a Tabela 9, percebe-se que ocorre para os BDORs: 1, 2 e 3 um mascaramento similar aos do momento da inserção. Ou seja, para cada *Select count*, são buscados apenas a quantidade de linhas existentes na tabela mãe. Para que se visualize a quantidade de registros individualmente entre as tabelas filhas, foi necessária uma consulta direta a um campo único da tabela, que como não apresentou tempos acima da linha de segundos, não recebeu maior destaque. Logo, a exceção se mantém a BDOR4 o qual o SGBD retorna a quantidades de registros de forma individual.

Seleções - Consulta aninhada. Apesar de nenhuma entidade ter ultrapassado 1 GB em tamanho, a fim de avaliar o comportamento e tempo de execução do SGBD mediante buscas de objetos complexos, foram feitas consultas para a busca de 500 registros e depois para todos os registros da tabela. A Tabela 10 ilustra o tempo de acesso a cada base de dados.

Seleção de todos os dados da tabela BDOR		
Banco.Tabela	**Linhas retornadas**	**Tempo**
BDOR1.SERVIDOR_OR_TAB	22372	00:09:41.11
BDOR1.SERVIDOR_OR_TAB	500	00:00:00.205
BDOR2.SERVIDOR_OR_TAB_TY	30100	00:00:20.19
BDOR2.SERVIDOR_OR_TAB_TY	500	00:00:0.088
BDOR2.SERVICO_OR_TAB	150500	00:20:57.38
BDOR2.SERVICO_OR_TAB	500	00:00:00.296
BDOR3.TB_SERVICO_TYP	50043	00:02:17.51
BDOR3.TB_SERVICO_TYP	500	00:00:00.042
BDOR3.TB_SERVIDOR	150000	00:02:28.78
BDOR3.TB_SERVIDOR	500	00:00:00.13
BDOR4.TVULNER	144903	00:08:53.93
BDOR4.TSERVICO	150000	00:00:27.39
BDOR4.TSERVIDOR	26986	00:00:55.08
BDOR4.TVULNER	500	00:00:00.057
BDOR4.TSERVICO	500	00:00:00.033
BDOR4.TSERVIDOR	500	00:00:00.05

Tabela 10: Seleção de todos os dados das tabelas de testes.
Fonte: Elaborado pela autora.

Novamente avaliando cada BDOR, em BDOR1 apesar de apresentadas 22372 linhas como retornadas, o resultado como visto anteriormente (seção 3.4) é uma consulta aninhada, ou seja, todos os campos definidos nas três tabelas são

retornados. Mas a contagem do SGBD leva em conta apenas o número de linhas da tabela mãe, o que neste caso não é um problema, afinal, o resultado da consulta não é mascarada. E se houver necessidade de consulta a qualquer uma das tabelas, basta efetuar uma query para busca desaninhada. BDOR2 tem as mesmas considerações de BDOR1 com exceção para a REF, que traz de forma direta os dados da tabela referenciada. Contudo, BDOR3 apesar de ter sido construído sobre o conceito de agregação recebe a mesma análise que BDOR2, principalmente em se tratando do item REF. Por outro lado, BDOR4 apesar de sua estrutura desaninhada, consegue devolver todos os dados das três tabelas de forma peculiar, assim como individualmente. Entretanto, seu tempo de resposta apresentou diferenças em relação as demais bases. Ao compará-lo com BDOR1, por exemplo, percebe-se certo grau de disparidade entre as tabelas, visto que BDOR4 não possui aninhamento, mas, isso será melhor discutido no item de avaliação de resultados.

Seleções - Consulta desaninhada. A avaliação de consultas desaninhadas será feita apenas no modelo BDOR1, por este apresentar estrutura complexa mais completa que os demais modelos até aqui definidos. A tabela 11 ilustra os resultados obtidos para BDOR1 ao ter suas tabelas consultadas de forma individual.

BDOR1 – Consulta desaninhada		
Tabela	**Quantidade de Registros**	**Tempo**
SERVIDOR_OR_TAB	22372	00:09:41.11
SERVIDOR + SERVICO	111860	00:01:39:39
SERVIDOR + VULNERABILIDADE	246092	01:40:46.48
SERVICOS + VULNERABILIDADE	111860	00:09:51.58
SERVIDOR	22372	00:00:14.26
SERVICOS	111860	00:00:21.30
VULNERANILIDADE	246092	00:17:11.63

Tabela 11: BDOR1 – Consultas desaninhadas.
Fonte: Elaborado pela autora.

Observando os resultados apresentados na Tabela 11, a consulta desaninhada em alguns casos resulta em respostas bem mais rápidas e diretas, em outros, o tempo tende a ser bem maior. No caso da consulta na qual se obtém todos os campos da tabela Servidor, juntamente com seus respectivos Serviços, o resultado é animador. Mas, se comparado com Servidor e Vulnerabilidades é necessária uma observação mais minuciosa, visto que, o tempo ultrapassa a consulta aninhada, porém, se considerados os 22372 registros há um retorno 11 vezes maior, uma vez

que, o SGBD efetua a contagem a partir da tabela Vulnerabilidades e não pela tabela Servidor. Agora se for considerado o valor real de 380324 registros, a consulta aninhada retorna quase 2 vezes mais registros que a união entre Servidor e Vulnerabilidades.

Vale ressaltar que as consultas individuais se aproximam da implementação de um BDR, por isso, não merecem maiores detalhes, visto que o tempo de resposta esta dentro dos padrões aceitável. (Consultas com o mínimo de 1 milésimo de segundo, há 20 minutos a partir de 150000 registros).

3.6 Avaliação dos Resultados.

Analisando os resultados dos testes, é possível constatar que a utilização do BDOR deve ser considerada. Pois, apesar de simples, a aplicação permitiu a avaliação de mais de uma estrutura definida em BDOR.

BDOR1, BDOR2, BDOR3 e BDOR4 com diferentes estruturas baseados em uma única aplicação permitiu resultados compensadores para análise e observação.

BDOR1 e BDOR2 com aninhamento de tabelas demonstraram que a representação com tipos de dados complexos aumenta o grau de inter-relacionamento ente os dados, permitindo a execução de consultas mais eficientes e um desempenho maior ao efetuar operações de inserts, pois, todos os campos podem ser inseridos de uma única vez. BDOR3 também não apresentou problemas de desempenho, sua carga efetuada na tabela Serviço com atributo do tipo Vulnerabilidade se comportou de forma eficaz. O único ponto a ser novamente destacado, é que neste modelo a implementação do atributo na Tabela Tipada só permite relacionamento de 1:1. Entretanto, assim como em BDOR2 a inserção de dados para estas tabelas pode ser feita de forma única, uma query que exige atenção e paciência se for desenvolvida através de linha de código, mas que pode ser mais eficiente se implementada em alguma linguagem de programação ou efetuada através de um framework.

Retomando BDOR4 X BDOR1, por BDOR4 se aproximar de um modelo relacional, esperava-se que seu comportamento seria semelhante, assim, como um desempenho mais dinâmico. Contudo, ao trabalhar com as tabelas/objetos com REF, apesar do retorno direto, a estrutura aninhada se mostrou mais rápida no retorno das pesquisas apresentando desempenho superior. Considerando que

BDOR1 possui 15% mais registros que BDOR4 e BDOR4 possui apenas 48,4% do tamanho de BDOR1.

Outro ponto é BDOR1 e BDOR2, a qual a tabela Serviço de BDOR2 apresentou tempo superior a 20 minutos enquanto BDOR1 teve sua operação realizada em apenas 9 minutos.

O desempenho de BDOR1 demonstrou-se superior indiferente da quantidade de registros e da forma de busca, obtendo um pico apenas ao consultar Servidor com Vulnerabilidades. Portanto, avaliando os tipos definidos em BDOR para as diferentes implementações propostas, destacam-se: tabelas tipadas com atributo de domínio de objeto complexo em consulta aninhada (resultado de todos os campos da tabela/objeto). Posteriormente as tabelas com escopo de REF ou apenas REF e as tabelas com agregação.

3.7 *BDOR1 X BDR.*

Como o modelo que mais se destacou no processo de desempenho em BDOR foi o BDOR1. Pretende-se através do conceito de junção obter todos os dados das tabelas do BDR, para comparar o desempenho deste com o modelo de tabelas aninhadas de BDOR1. A Tabela 12 apresenta a comparação entre os dois modelos:

Junção em BDR X Consultas BDOR1			
Tabela	**Quantidade de Registros**	**Tempo BDOR1**	**Tempo BDR**
SERVIDOR+SERVICO+VULNERABILIDADE	22372	00:09:41.11	00:55:02.62
SERVIDOR + SERVICO	111860	00:01:39:39	00:00:29.83
SERVIDOR + VULNERABILIDADE	246092	01:40:46.48	00:45:46.48
SERVICOS + VULNERABILIDADE	111860	00:09:51.58	00:29:51.58
SERVIDOR	22372	00:00:14.26	00:00:00.055
SERVICOS	111860	00:00:21.30	00:00:00.084
VULNERANILIDADE	246092	00:17:11.63	00:00:00.202

Tabela 12: Junção em BDR X Consultas em BDOR.
Fonte: Elaborado pela autora.

Na Tabela 12 os dados foram recuperados com registros em junções idênticas, não desconsiderando qualquer registro entre as tabelas. Ou seja, os dados da chave estrangeira são iguais os da chave primária. Também foram realizados testes unindo as três ou apenas duas das tabelas, e por ultimo, mas não menos importante foram efetuadas consultas individuais. Comparando a aplicação BDOR1 com BDR, percebe-se que ao efetuar uma busca conjugada entre as três tabelas desenhadas,

o processamento do BDOR não é eficiente, levando cerca de 1 hora para a busca do resultado contra aproximadamente 10 minutos em BDOR1. Em alguns casos, quando a consulta é efetuada com baixa quantidade de dados, o BDR passa a ter um considerável desempenho. Para os outros BDR consegue se superar em alguns milésimos de segundos, entretanto, não há disparidade verdadeiramente relevante entre os resultados.

4. Conclusão e Trabalhos Futuros.

A contribuição deste trabalho destaca-se na pesquisa prática que procurou mostrar o uso produtivo do BDOR e não apenas o conceitual, como em trabalhos correlatos. Procurou-se difundir os principais recursos de BDOR de modo a facilitar sua utilização e estudo, já que o mesmo exige um pouco mais de empenho do analista para ser implementado com êxito, por isso, a conclusão dos testes e o desenvolvimento de frameworks para este tipo de atividade torna-se tão necessária, sendo uma vez difundido, seu uso passará a ser utilizado de forma mais produtiva.

O BDOR oferece estruturas compatíveis com linguagens de programação OO, o que já minimiza o trabalho adicional para o armazenamento de objetos complexo necessário ao modelo relacional. Ao utilizar o SGBD Oracle como ferramenta de teste, toda estrutura, desde a instalação até a criação das tabelas/objetos, não apresentaram maiores dificuldades, cabe apenas ressaltar que a sintaxe da SQL e do Oracle podem diferenciar-se, por isso, é importante separar o que é exagero da realidade. Pois, devido à falta de similaridade entre as linguagens, a princípio muitos itens definidos foram jogados fora durante o processo de construção desta monografia, uma vez que, alguns conceitos e códigos definidos pela SQL não funcionam na prática no Oracle, ou são completamente diferentes entre a teoria e a forma de implementação.

Embora o ajuste de desempenho do SGBD Oracle tenha sido realizado através de manuais disponibilizados pelo fabricante, seu desempenho não deixou a desejar. Mesmo assim, esperar-se que um ajuste mais elaborado possa produzir resultados diferentes.

É bem obvio que existe um esforço adicional para criar, armazenar e buscar dados em estruturas complexas, mas, após um tempo dentro de um circulo vicioso entre estudo, erros e acertos exatamente nesta ordem, é notório que não se trata apenas de um processo complexo, mas sim de um processo ainda em desenvolvimento. Que, se levado a sério, poderá garantir meios eficazes para busca em bases de dados que exigem o mínimo de atenção e trabalham com grandes volumes de dados, assim como já destacado inicialmente, aplicações como, SIG, sistemas de multimídia e softwares de inteligência artificial. Mas, após a conclusão deste trabalho, outro exemplo interessante de utilidade para o BDOR são os processos médicos que exigem diagnóstico por imagem agregados ou compostos

por valores específicos e descrição da consulta. Um ortopedista e ortomolecular pode se beneficiar em muito com uma aplicação que utilize BDOR, visto que será possível carregar e buscar dados complexos de forma mais espontânea ocasionando diagnósticos mais precisos e completos.

Procurou-se constatar itens que revelassem que os diferentes usuários de banco de dados, possam ser liberados da inércia conservadora que permanecem firmes no conceito do BDR, sem levantar as vantagens do BDOR. Bem, sabe que as definições são complexas, visto a quantidade de informações, porém, toda esta coleção de informações mostrou-se motivacionais e dentro do atual contexto comercial ou acadêmico, oferecendo oportunidade de elaboração de produtos uteis e com boa probabilidade de adesão a quem necessite de uma base de dados com persistência de objetos e dados complexos.

Este trabalho baseou-se nas pesquisas de CASTRO (2011) e ROMBALDO (2012), entretanto, não houve a utilização de frameworks para a construção das bases de dados, sendo todo o processo direto no terminal do SGBD (SQL*Plus). Contudo, para trabalhos futuros, incluem a realização dos mesmos testes utilizando bases com tamanhos diferenciados (acima de 1G), o uso do SGBD DB2 para análise e comparação com o SGBD Oracle, e framework de construção, assim como a inclusão de testes de alteração e exclusão. Como também novas consultas e teste de estresse.

5. REFERÊNCIAS BIBLIOGRÁFICAS.

BLAHA, M.; RUMBAUGH, J. Modelagem e Projetos Baseados em Objetos com UML2 – 2ª ed.- Rio de Janeiro: Campus , 2006.

BOSCARIOLI, C.; BEZERRA, A.; BENEDICTO, M.; DELMIRO, G. (2006) "*Uma reflexão sobre Banco de Dados Orientado a Objetos.*" UNIOESTE-FASP.

CASTRO, T. R. (2011) "Proposta de um modelo para projetos lógicos gráficos para BDOR com implementação no ARGOUML" EP-USP.

CERÍCOLA, O. V. Banco de Dados relacional e distribuído – Rio de Janeiro: LTC – Livros Técnicos e Científicos Ed., 1991.

DATE, C. J. Introdução a Sistemas de Banco de Dados – 7ª ed. – Rio de Janeiro: Campus, 2000.

EISENBERG, A.; MELTON, J.; KULKARNI, K.; MICHELS J. E ZEMKE, F. (2004) "SQL:2003 Has Been Published." SIGMOD Record.

KORTH, H.F.; SILBERSCHATZ, A.; SUDARSHAN, S. Sistemas de Banco de Dados- 5ª e.d.- Rio de Janeiro: Campus, 2006.

LIMA, A. G. A. (2011) "Padrão SQL e sua Evolução." UNICAMP.

MICHAEL, A.; IAN, A.; MICHAEL, C. Oracle 9i, guia introdutório – Rio de Janeiro: Campus, 2002.

KELLY, D. A. (2007). "Oracle Celebrates 30 Years of Innovation". Disponível em: < www.oracle.com>. Acesso em: 10 mar. 2013.

ORACLE, Technology Products. Disponível em: < http://www.oracle.com> Acesso em: 10 mar. 2013.

RAMOS, J. M.; RIO, J. H. (2010) "DB2 Conceitos e Análise do Sistema" Disponível em: <www.fct.unl.pt/>. Acesso em: 10 mar. 2013.

ROMBALDO, C. A. J.; CASTRO,T. R.; SOUZA, S. N. A. (2009) "Banco de Dados Objeto-Relacional: comparação do suporte oferecido por SGBDs para a persistência de objetos" EP-USP.

ROMBALDO, C. A. J. (2012) "Proposta de um framework de persistência de objetos em bases de dados objeto-relacional" EP-USP.

SCHWEINSBERG, Dipl-Math Kai. (2012) "Abbildung von XML-Dokumenten auf SQL: 2003-konforme Datentypen." Universität Kassel.