UNIVERSIDADE DE CAXIAS DO SUL
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO
TRABALHO DE CONCLUSÃO DE CURSO II

MIGRATE DATABASES – MIGRADOR GENÉRICO DE BANCOS DE DADOS RELACIONAIS

Maycon Ferraça

Caxias do Sul
2008

MAYCON FERRAÇA

MIGRATE DATABASES – MIGRADOR GENÉRICO DE BANCOS DE DADOS RELACIONAIS

Trabalho de conclusão apresentado para a obtenção de grau de Bacharel em Sistemas de Informação, junto à Universidade de Caxias do Sul.

Supervisor:

Profº. Ms. Daniel Luis Notari

Orientadora:

Profª. Drª. Helena Graziottin Ribeiro

Caxias do Sul

DEZ/2008

AGRADECIMENTOS

Agradeço à orientadora Profª. Drª. Helena Graziottin Ribeiro pelo suporte no desenvolvimento deste trabalho.

Agradeço aos professores do curso de Sistemas de Informação, em especial ao Profº. Ms. Daniel Luis Notari pela boa coordenação do presente trabalho de conclusão de curso e a professora Msª. Iraci Cristina da Silveira pelo auxílio na modelagem do projeto proposto.

Agradeço também os meus pais, Sandra e Devanir, e a minha irmã Camyla, pela presença e apoio constante.

“A soul in tension that's learning to fly,
condition grounded but determined to try”

David Gilmour

RESUMO

Este documento apresenta a proposta de um migrador genérico de bancos de dados relacionais, para suprir a necessidade do mercado de não possuir uma ferramenta de código aberto que migre todos os metadados e os dados, e que seja independente das bases de origem e destino. Serão descritas quais são as necessidades de uma ferramenta de migração, quais são as ferramentas disponíveis atualmente e uma comparação entre estas. Serão apresentadas as justificativas do desenvolvimento de um novo aplicativo e quais as características fundamentais para este sistema. Além disto, será apresentada a proposta de desenvolvimento, definição do projeto, restrições de escopo do aplicativo, distribuição, licença, configuração da ferramenta, metodologia de uso e os principais problemas entre migrações.

Palavras-chave: Adaptação, migração, conversor, SGBD, banco de dados, origem, destino.

ABSTRACT

This document shows the proposal of a generic migratory of relational databases, to fill the necessity of the market that does not have an open-source tool to migrate all the metadata and the data, and is independent of the database source and database target. The necessities of the migratory tool, the currently available tools and the comparison among these shall be described. The reason of the development of a new application and what the elementary features of this system are shall be raised. In addition to defining the proposal, the project definition, restrictions of the application scope, distribution, license, settings of the tool, methodology of use and the principal problems among migrations shall also be defined.

Keywords: Adaptation, migration, converter, DBMS, database, source, target.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE EXEMPLOS

LISTAS DE SIGLAS

ACID	Atomicidade, Consistência, Isolamento e Durabilidade
ANSI	*American National Standards Institute*
ASA	*Adaptive Server Anywhere*
ASE	*Adaptive Server Enterprise*
BDE	*Borland Database Engine*
BLOB	*Binary Large Objects*
CLOB	*Character Large Object*
CVS	*Concurrent Version System*
DBX	*DataBase eXpress*
DDL	*Data Definition Language*
DML	*Data Manipulation Language*
DBMS	*Data Base Management System*
ETL	*Extract, Transform and Load*
ERP	*Enterprise Resource Planning*
HTML	*HyperText Markup Language*
IBM	*International Business Machines*
IDE	*Integrated Development Environment*
INI	*INItialization*
JDBC	*Java Database Connectivity*
MPL	*Mozilla Public License*
ODBC	*Open Data Base Connectivity*
OLE DB	*Object Linking and Embedding DataBase*
SQL	*Select Query Language*

SGBD	Sistema Gerenciador de Banco de Dados
UML	*Unified Modeling Language*
VCL	*Visual Component Library*
XML	*eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Este documento descreve o desenvolvimento do Trabalho de Conclusão de Curso no segundo semestre de 2008, para obtenção do grau de bacharel em Sistemas de Informação pela Universidade de Caxias do Sul.

A idealização do trabalho se fez através do estágio curricular para a empresa Núcleo Sistemas de Informática. Esta possuía o seu ERP em banco de dados *Sybase Adaptive Server Anywhere* e necessitou da adaptação deste para o banco de dados gratuito *Firebird*. Para esta adaptação foi necessário a utilização de um migrador de banco de dados, que pudesse migrar as informações de metadados e os dados em si. Como solução foi desenvolvida uma ferramenta pela própria Núcleo Sistemas [Ferraça, 2007]. Entretanto esta ferramenta somente migra bancos ASA para *Firebird* e a partir deste ponto surgiu a idéia de desenvolver um conversor que fosse independente dos produtos utilizados para o Trabalho de Conclusão de Curso.

O objetivo principal do projeto é o desenvolvimento de uma ferramenta que seja código aberto e que tenha a capacidade de ser genérica o suficiente para independer dos bancos de dados utilizados. Esta ferramenta deve ser capaz de ler e migrar metadados e os dados de um SGBD de origem e gerar comandos DML e DDL para o SGBD de destino, respeitando as possíveis diferenças de sintaxes.

Softwares utilizam a arquitetura de camadas, sendo elas de uma à "*n*" camadas. Estas podem ser a camada de apresentação para o usuário, regras de negócios, persistência e entre outras [Rodrigues, 2002,]. O objetivo da ferramenta é justamente de migrar a camada de persistência, possibilitando os *softwares* serem mais flexíveis.

Este trabalho irá definir a ferramenta, sendo o segundo capítulo deste documento responsável por caracterizar o modelo de multicamadas e a sua origem, mostrando a necessidade da utilização de uma ferramenta de migração de uma base de dados para outra. Após são descritas as ferramentas existentes no mercado com suas principais características, apontando uma tabela comparativa entre elas.

O terceiro capítulo, primeiramente, irá apresentar o escopo do projeto, isto é, quais tipos de bancos de dados o aplicativo irá suportar. Também serão apresentadas as operações

que deverão ser feitas por ele. Será inclusive descrito a idéia para o funcionamento do sistema, mostrando como o aplicativo pretende ler bancos de dados diferentes e como serão gerados comandos DML e DDL para diferentes produtos.

Para a apresentação da forma de desenvolvimento do aplicativo serão utilizados diagramas UML, já que o sistema será desenvolvido com uma linguagem orientada a objetos. Também será definido um protótipo do aplicativo, referenciando as principais telas que estarão disponíveis para o usuário final. Além disso, será definida a ferramenta para o desenvolvimento do aplicativo, incluindo o *driver* que será utilizado para dispor conexões com as bases de dados, e as restrições que são aplicadas à utilização do sistema, isto é, quais são as exceções do escopo do projeto. Será também apresentado como se pretende distribuir o software depois do seu desenvolvimento e qual será a licença para uso deste, já que se tratará de um sistema de código aberto.

O quarto capítulo apresentará o desenvolvimento da ferramenta, mostrando quais foram às alterações na proposta inicial de desenvolvimento, tratadas pelo terceiro capítulo. O quinto capítulo será responsável por descrever a maneira de configurar a ferramenta desenvolvida, para novas migrações ou para ajustes finos em migrações já existentes. O sexto e último capítulo irá demonstrar o uso da ferramenta, testes de migração e mostrar os problemas encontrados e as correções para a configuração dos quatro bancos de dados homologados, o *Firebird 2.1*, *Microsoft* SQL *Server* 2005, *PostgreSQL* 8.3 e o *Sybase Adaptive Server Anywhere 8*.

2 ESTADO DA ARTE

Segundo o dicionário *on-line Priberam* da Texto Editores Universal[1] a palavra "*estado*" é definido como "*modo de ser ou de estar de uma pessoa ou coisa*" e a palavra "*arte*" como o "*conjunto de preceitos ou regras para bem dizer ou fazer qualquer coisa*", definindo o título do corrente capítulo como "*estado da arte*" por indicar a situação a qual se encontra o aspecto teórico e prático que envolve o corrente projeto como uma fotografia no tempo, apresentando o que é necessário para o estudo em questão deixar de ser apenas teoria para se tornar uma utilidade.

Este capítulo irá apresentar o modelo de multicamadas, para introduzir a necessidade da utilização de uma ferramenta de migração de banco de dados. Serão apresentadas algumas das ferramentas existentes no mercado e uma comparação entre elas.

2.1 Arquitetura multicamadas dos *softwares*

Segundo Fowler (2006) os *softwares* são divididos em camadas desde os seus primórdios, passando desde a arquitetura de uma camada até um modelo que possa conter três ou mais camadas.

O modelo de uma camada era visto em sistemas centralizados, no qual juntava banco de dados, regras de negócios e interfaces de usuário em um único computador de grande porte de processamento, os conhecidos mainframes. Este computador por sua vez era acessado por terminais clientes que não possuíam nenhum recurso de processamento ou armazenamento, os conhecidos terminais burros.

[1] Disponível em http://www.priberam.pt/.

O modelo de duas camadas originou-se com o surgimento dos microcomputadores, onde os terminais conectados ao servidor também possuíam poder de processamento, podendo conter regras de negócios (figura 1). Existem modelos de duas camadas onde as regras de negócios ainda ficam no servidor, geralmente de responsabilidade do banco de dados.

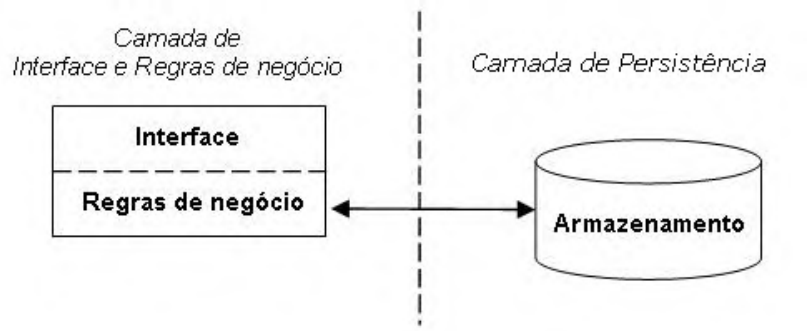


Figura 1: Paradigma de duas camadas.

Fonte: Ferraça (2007).

A separação das camadas de interface, regras de negócio e armazenamento, o modelo de "*n*" camadas ou multicamadas é a evolução dos modelos anteriores. O modelo é conhecido de multicamadas por que ainda podem existir camadas adicionais para entre estas três, com alguma funcionalidade em específico. Como na figura 2 que mostra uma camada adicional para o tratamento de comunicação entre a camada de regras de negócios com a camada de persistência.

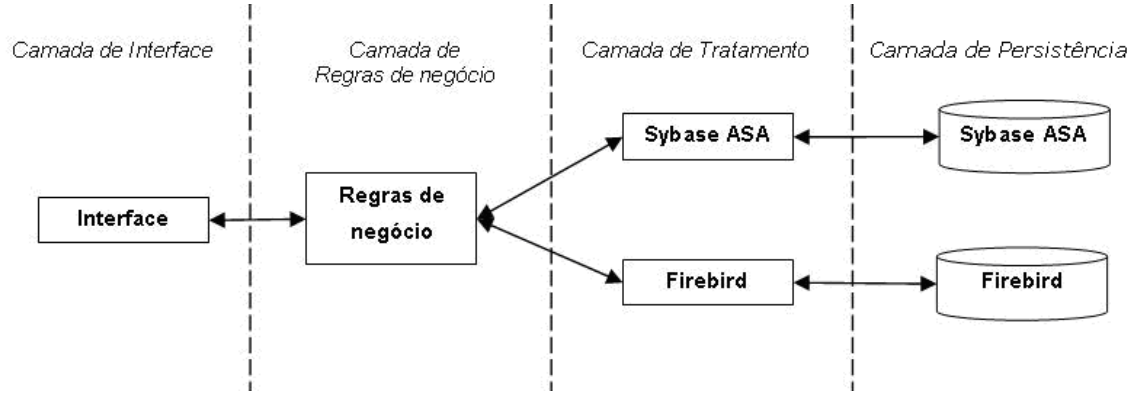


Figura 2: Modelo multicamadas.

Fonte: Ferraça (2007).

Rodrigues (2002) diz que uma aplicação desenvolvida no modelo de multicamadas apresenta diversas vantagens em relação a outros modelos, entre estas vantagens estão a independência do sistema gerenciador de banco de dados.

O papel do banco é de repositório de dados em sistema de multicamadas, não utilizando recursos como gatilhos ou procedimentos armazenados. Isto porque se for utilizado algum destes recursos a camada de regras de negócios acaba se acoplando com a camada de armazenamento. Além do mais, estes recursos são implementados diferentemente por cada fabricante, o que acaba dificultando a migração desta camada de persistência.

Mas mesmo assim, na troca da camada de armazenamento, a criação do metadados e a conversão dos próprios dados em bancos diferentes pode ser um problema, pois nem todos seguem a risca a conformidade de comandos SQL, DML e DDL conforme o que especifica o padrão SQL ANSI [ANSI-SQL 92; ANSI-SQL 99; Ferraça, 2007].

2.2 Necessidade de uma ferramenta de migração

Algumas empresas, como a Núcleo Sistemas, já possuem o seu produto desenvolvido e necessitaram do suporte para um SGBD diferente do atual utilizado. A empresa recém citada já possuía o seu ERP em banco de dados *Sybase Adaptive Server Anywhere* e necessitou a adaptação deste para o banco de dados gratuito, o *Firebird*. Para esta adaptação foi necessário a utilização de um migrador de banco de dados, que pudesse migrar as informações de metadados e os dados em si. Como solução foi adotada o desenvolvimento de uma ferramenta pela própria *software house* [Ferraça, 2007].

O desenvolvimento poderia ter sido evitado, pois são disponíveis no mercado diversas ferramentas para este fim, o que talvez pudesse auxiliar nesta situação. Contudo as ferramentas disponíveis possuem as suas limitações, como por exemplo, o número finito de sistemas gerenciadores de banco de dados nos quais atuam. E como será visto a seguir nenhum dos conversores analisados suporta a condição de migrar um banco de dados em

Sybase Adaptive Server Anywhere para *Firebird*, justamente a necessidade da Núcleo Sistemas.

2.3 Ferramentas disponíveis

Nos próximos subtópicos são descritas algumas ferramentas que suportam a conversão de um banco de dados para outro, com o intuito de saber o que está disponível no mercado e o que elas atendem e também não atendem. As informações foram obtidas do próprio site dos fabricantes e as ferramentas apresentadas foram a ESF *Database Convert, SQLWays, SwisSQL, FlySpeed, Migrate-Data, Fullconvert* e a *openDBcopy*.

Segundo Giustina (2002), existe uma espécie de ferramentas, conhecidas como ETL responsáveis pela extração, transformação e carga de dados de sistemas do tipo *Data Warehouse* com origem de diversas fontes de dados, inclusive bancos de dados relacionais. Entretanto, esta categoria de ferramentas não foi analisada, porque independente de terem ou não também a capacidade de converter um banco de dados relacional para outro, foge do escopo do trabalho, isto é, uma ferramenta somente para este fim.

2.3.1 Ferramenta ESF *Database Convert*

A primeira ferramenta analisada foi à ferramenta ESF *Database Convert* (versão 5.7.37) da empresa *Easy From Technology,* gratuita para testes pelo período de 30 dias e disponível em http://www.easyfrom.net/.

Segundo o fabricante, a ferramenta disponibiliza acesso nativo para diversos formatos descritos na tabela 1, sendo possível a escolha das tabelas de origem e seus campos para serem convertidos para um determinado destino. A ferramenta também suporta caracteres

Unicode, conversão de visões, chaves primárias, índices, chaves estrangeiras, a alteração de nomes de tabelas, tipos, tamanhos, *default* de campos. E a conversão de dados (e somente dados) de qualquer origem ODBC para qualquer outro destino ODBC.

Tabela 1:[2] [3] Suporte de conversão para a ferramenta ESF *Database Convert*.
Fonte: http://www.easyfrom.net/.

Origem	**Destino**
MySQL	*MySQL*
SQL *Server* 2000/2005	SQL *Server* 2000/2005
PostgreSQL	*PostgreSQL*
Oracle	*Oracle*
Interbase	*Interbase*
Access 2000/2007	*Access* 2000/2007
Excel 2000/2007	*Excel* 2000/2007
Visual Foxpro	*Visual FoxPro*
dBase	*dBase*
Lotus	*Lotus*
Paradox	*Paradox*
Arquivos Texto/CVS	Arquivos Texto/CVS
Arquivos HTML	Arquivos HTML
ODBC	

2.3.2 Ferramenta *SQLWays*

[2] É possível migrar todos os bancos de dados de origem com todos os bancos de dados de destino, para esta e pra as tabelas de 2 a 7.

[3] As informações de versões dos bancos de dados suportados, tanto de origem como de destino, foram obtidas dos sites dos fabricantes e nem todos disponibilizavam informações completas. Por isto algumas tabelas (2 a 7) possuem versão dos produtos e outras não.

A segunda ferramenta analisada foi à ferramenta *SQLWays* (versão 3.9) da empresa *Ispirer*, gratuita para testes pelo período de 30 dias e disponível em http://www.ispirer.com/.

Conforme informações do *site* da desenvolvedora, a ferramenta oferece suporte à conversão de campos BLOB/CLOB, conversão de tabelas (incluindo valores *defaults*, nulos, *check constraints* e *unique constraints*, chaves primárias, chaves estrangeiras), além de comentários de objetos, a conversão de índices, visões. Também oferece suporte a conversão de gatilhos e procedimentos armazenados, desde que o gerenciador de banco de dados seja o mesmo. A ferramenta oferece suporte de conversão aos bancos de dados descritos na tabela 2.

Tabela 2: Suporte de conversão para a ferramenta *SQLWays*.

Fonte: http://www.ispirer.com/.

Origem	**Destino**
IBM DB2 para *Linux*, *Unix* e *Windows* 9.x, 8.2, 8.1, 7.2, 7.1, 7.0, 6.1 e anteriores	IBM DB2 para *Linux*, *Unix* e *Windows* 9.x, 8.2, 8.1, 7.2, 7.1, 7.0, 6.1 e anteriores
IBM DB2 para z/OS e OS/390, MVS 9.x, 8.1, 7.1, 6.1, 5.2, 5.1 e 4.1	IBM DB2 para z/OS e OS/390 9.x, 8.1, 7.1, 6.1, 5.2, 5.1
IBM DB2 para *iSeries* e AS/400 V5R3, V5R2, V5R1, V4R5, V4R4 e anteriores	IBM DB2 para *iSeries* e AS/400 V5R3, V5R2, V5R1, V4R5, V4R4
Oracle 11g, 10g, 9i, 8i, 8.0.x e 7.x	*Oracle* 11g, 10g, 9i, 8i e 8.0.x
Microsoft SQL *Server* 2005, 2000, 7.0 e 6.5	*Microsoft* SQL *Server* 2005, 2000, 7.0 e 6.5
Sybase Adaptive Server Enterprise (ASE) 15.x, 12.5.1, 12.5, 12.0, 11.x e anteriores	*MySQL* 5.x, 4.x e 3.23
Sybase Adaptive Server Anywhere (ASA), *Sybase SQL Anywhere* 9.0, 8.0.x, 7.0.x, 6.0.x, 5.5 e 5.0	*PostgreSQL* 8.x e 7.x
Sybase IQ 12.x	*Sybase Adaptive Server Enterprise* 15.x, 12.5.1, 12.5, 12.0, 11.x
Informix Dynamic Server (IDS) 10, 9.4, 9.3, 9.2, 9.1, 7.3, 7.2, 7.1 e anteriores	*Sybase Adaptive Server Anywhere* 9.0, 8.0.x, 7.0.x, 6.0.x, 5.5 e 5.0
Informix Steard Engine (SE) 7.x, 5.x, 4.x e anteriores	*Sybase* IQ 12.x
MySQL 5.x, 4.x, 3.23 e anteriores	*Informix Dynamic Server* (IDS) 10, 9.4, 9.3, 9.2, 9.1, 7.3, 7.2, 7.1 e anteriores
PostgreSQL 8.x, 7.x e 6.x	*Pervasive.*SQL v8
Progress 10.x, 9.x e 8.x	

SAP DB 7.4, 7.3 e anteriores	
Pervasive.SQL v8, 2000 e 7	
Microsoft Access 2003, 2000, 97, 95 e 2.0	
Interbase 7.1, 6.x, 5.x e 4.x	
Firebird	
Lotus Notes 6, 5 e anteriores	
dBase, *FoxPro*, *Excel*, *Paradox*, *Gupta SQLBase*, *Clipper* e qualquer outro a partir de uma fonte ODBC	

2.3.3 Ferramenta *SwisSQL*

A terceira ferramenta analisada foi a *SwisSQL - Data Migration Tool* (Versão 4.9) da própria *SwisSQL*, disponível para testes por 30 dias no endereço eletrônico da fabricante: http://www.swissql.com/products/datamigration/data-migration.html.

A ferramenta tem capacidade de migração de tabelas, índices, visões, *constraints* através das fontes indicadas na tabela 3. Suporta migração de procedimentos armazenados, funções e gatilhos (desde que sistema gerenciador de banco de dados de origem e destino seja o mesmo). Suporta também a opção de troca de tipos de dados de colunas entre as fontes, além de outras características.

Tabela 3: Suporte de conversão para a ferramenta *SwisSQL*.
Fonte: http://www.swissql.com/products/datamigration/data-migration.html.

Origem	**Destino**
Oracle 8i, 9i, e 10g	*PostgreSQL* 7.3.x, 7.4.x e 8.x
IBM DB2 UDB para *Linux*, *Unix*, e *Windows* 7.1 e 8.x	IBM DB2 UDB para *Linux*, *Unix*, e *Windows* 7.x, 8.x
Microsoft SQL *Server* 7, 2000, e 2005	*Microsoft* SQL *Server* 2000 e 2005
Sybase ASE 11.x, 12.x, e 15.0	*Oracle* 9i e 10g
Sybase ASE 11.x, 12.x, e 15.0	*Sybase* ASE 11.x, 12.x, e 15.0
Sybase ASE 11.x, 12.x, e 15.0	*MySQL* 3.23.x, 4.x, e 5.0

Sybase ASE 11.x, 12.x, e 15.0	
Sybase ASE 11.x, 12.x, e 15.0	
MySQL 3.23.x, 4.x, e 5.0	
PostgreSQL 7.3.x, 7.4.x e 8.x	
MySQL MaxDB (SAP DB) 7.5.00	
Microsoft Access 2000	
Microsoft Excel e arquivos texto	

2.3.4 Ferramenta *FlySpeed*

A quarta ferramenta analisada foi a *FlySpeed* (versão 1.6) da fabricante *Active Database Software* disponível para testes por 30 dias no endereço http://www.activedbsoft.com/overview-migrate.html.

Esta ferramenta suporta qualquer fonte de dados de origem através da conexão de OLE DB ou ODBC, entretanto os destinos somente são os SGBD *MySQL* ou *Microsoft SQL Server* (tabela 4). Algumas outras características desta ferramenta são conversão de campos do tipo BLOB/CLOB, suporte a *Unicode*, suporte a conversão de metadados (tabelas, *constraints* e índices).

Tabela 4: Suporte de conversão para a ferramenta *FlySpeed*.
Fonte: http://www.activedbsoft.com/overview-migrate.html.

Origem	**Destino**
Qualquer fonte de dados utilizando OLE DB ou ODBC	*MySQL*
	Microsoft SQL *Server*

2.3.5 Ferramenta *Migrate-Data*

A quinta ferramenta avaliada foi a *Migrate-Data* (versão 1.1) da empresa *Akcess*. Esta ferramenta também está disponível para testes por 30 dias no endereço http://www.akcess.in/MigrateData.html.

A ferramenta suporta a conversão de tabelas com os de tipos dados BLOB/CLOB, *constraints* do tipo *unique* e *check*, *default* de campos. Suporta também a conversão de visões e a conversão de procedimentos armazenados e gatilhos de tabelas (mas não entre todos os SGBD, e o sistema gerenciador de origem deve ser o mesmo do destino). E oferece suporte para conversão os bancos de dados descritos na tabela 5.

Tabela 5: Suporte de conversão para a ferramenta *Migrate-Data*.
Fonte: http://www.akcess.in/MigrateData.html.

Origem	**Destino**
Oracle	*Oracle*
PostgreSQL	*PostgrSQL*
Microsoft SQL *Server*	*Microsoft* SQL *Server*
DB2	DB2
MySQL	*MySQL*
Ingres	*Ingres*
Informix	*Informix*
Mimer SQL	*Mimer* SQL
MaxDB	*MaxDB*
Microsoft Access	
Excel e arquivos texto	

2.3.6 Ferramenta *Fullconvert*

A sexta ferramenta analisada foi a *Fullconvert* (versão 3.3) da empresa *Spectral Core* que pode ser baixada para testes de 30 dias no endereço http://www.spectralcore.com/fullconvert/index.php.

A ferramenta oferece suporte para conversão de dados, índices e *constraints*. Possibilidade de trocar os tipos de dados dos campos entre a tabela de origem e a tabela de destino. Os bancos de dados suportados são descritos na tabela 6.

Tabela 6: Suporte de conversão para a ferramenta *Fullconvert*.
Fonte: http://www.spectralcore.com/fullconvert/index.php.

Origem	**Destino**
Microsoft Access	*Microsoft* SQL *Server*
dBase	*Oracle*
FoxPro	*MySQL*
Microsoft Excel	*PostgreSQL*
Interbase	*Access*
Firebird	*Interbase*
Lotus 1-2-3	*Firebird*
MySQL	
Oracle	
PostgreSQL	
Qualquer banco com conexão ODBC	
Paradox	
Microsoft SQL *Server*	
Arquivos texto	
XML	

2.3.7 Ferramenta *openDBcopy*

A última ferramenta avaliada foi a *openDBCopy* (versão 0.51), uma ferramenta open-source desenvolvida por um grupo voluntário de programadores. Tanto o seu código-fonte como os binários desta ferramenta podem ser baixados no endereço http://opendbcopy.sourceforge.net/.

Com ferramenta é possível converter qualquer banco de dados para qualquer outro, desde que para estes existam *drives* JDBC. Mas esta conversão somente é feita para dados e não para metadados. Entretanto, segundo o *site* do produto é possível desenvolver *plug-ins* para acoplar com o *software*, o que possibilita o desenvolvido um para a conversão de metadados.

2.3.8 Comparando as ferramentas

A tabela 7 apresenta um comparativo entre as principais características das ferramentas recém analisadas. Estas características são baseadas conforme descrição dos sites dos desenvolvedores, qualquer informação não contida lá, mas mesmo que haja suporte, foi mencionado como sem. Algumas outras características consideradas como somente "comerciais" para exaltar os produtos, tais como "facilidade na utilização do produto" ou "maior gama de recursos", foram ignoradas. Estas restrições forram aplicadas desta maneira porque a intensão desta comparação não é eleger uma melhor ferramenta, mas sim descobrir as características mais importantes para serem suportadas como base no desenvolvimento da ferramenta proposta neste trabalho.

As características apresentadas na tabela 7 são:

- **Suporte a conversão de tabelas, chaves primárias e *check constraints***: Operações responsáveis por converterem a estrutura das tabelas, incluindo, tipo dos dados, valores *default* das colunas, definição de coluna nula ou não, chaves primárias e *check constraints*;
- **Suporte a conversão de visões**: Responsável pela conversão de visões utilizadas no banco de dados de origem para o banco de dados de destino;
- **Suporte a conversão de dados**: Conversão dos dados de uma base para outra;
- **Suporte a conversão de colunas auto-incremento**: Suporte a conversão de colunas do tipo auto-incremento entre as bases, pois cada SGBD implementa diferencialmente este recurso. Por exemplo, o *Firebird* implementa por

gatilhos e *sequences*, sendo necessária a criação destes após a conversão da tabela. Já o ASA, na definição da cláusula *default* de uma coluna [Ferraça, 2007];

- **Suporte a conversão de chaves únicas**: Responsável pela criação de *constraints* do tipo *unique*, que além implementarem consistência dos dados, são necessárias para a conversão de chaves estrangeiras, pois é possível, segundo normas da ANSI-SQL, a criação de chaves estrangeiras através destas [ANSI-SQL 92; ANSI-SQL 99];
- **Suporte a conversão de chaves estrangeiras**: Responsável pela criação de *constraints* do tipo *foreign key*;
- **Suporte a conversão de índices**: Suporte a conversão de índices únicos e não únicos;
- **Suporte a execução de *scripts* pré-conversão e pós-conversão**: Estas operações permitem a execução de scripts para a pré-conversão e pós-conversão dos bancos de dados de origem e destino, de acordo com a necessidade para cada conversão. Importantes, por exemplo, para a necessidade de criação de visões para compatibilidade utilizada no projeto de estágio, que consistiram na criação de visões para cada tabela do banco, com a diferença que todos os identificadores ficaram delimitados por aspas duplas [Ferraça, 2007];
- **Suporte a conversão de campos BLOB/CLOB**: Suporte para a conversão de campos binários e arquivos de textos longos;
- **Suporte a alteração de tipos de dados**: Possibilidade da ferramenta em converter um tipo de dados para outro, por exemplo, a conversão de um tipo de dados de CHAR no banco de dados de origem para VARCHAR no banco de dados de destino [Ferraça, 2007];
- **Suporte a alteração de *default* de colunas**: Possibilidade da ferramenta em converter um valor padrão de uma coluna para outro, por exemplo, a conversão do *default* de “CURRENT TIME” no banco de dados de origem para “CURRENT_TIME” no banco de dados de destino [Ferraça, 2007];

- **Suporte a caracteres *unicode***: Oferecer suporte ao conjunto de dados *unicode*;
- **Suporte a conversão de nomes de objetos**: Possibilidade da conversão de nomes de objetos de maiúsculo para minúsculo, maiúsculo para minúsculo, ou nenhuma conversão;
- **Suporte a conversão de comentários de objetos**: Suporte para a conversão para comandos que definem comentários para objetos;
- **Suporte a conversão de procedimentos armazenados, funções e gatilhos**: Todas as ferramentas analisadas que oferecem este recurso, apenas convertem quando os bancos de dados de origem e destino são os mesmos;
- **Suporte a controle de tamanho máximo de tipos de objetos entre banco de dados distintos**: Alguns bancos de dados possuem limites de tamanho de nomes para objetos [Ferraça, 2007];
- **Suporte a conversão total de múltiplos bancos de dados relacionais (conversão de dados e metadados)**: Ferramentas que seja independentes de uma relação finita de bases de origem e destino;
- **Ferramenta de código aberto ou licença gratuita**: Ferramentas que sejam de código aberto ou de licença gratuita.

Tabela 7: Comparativo de características entre as ferramentas.

Fontes: http://www.easyfrom.net/, http://www.ispirer.com/, http://www.swissql.com/products/datamigration/data-migration.html, http://www.activedbsoft.com/overview-migrate.html, http://www.akcess.in/MigrateData.html, http://www.spectralcore.com/fullconvert/index.php, http://opendbcopy.sourceforge.net/.

Característica	**Ferramentas**						
	ESF Database Convert	***SQL Ways***	***SwisSQL***	***FlySpeed***	***Migrate-Data***	***Fullconvert***	***openDBcopy***
Suporte a conversão de tabelas	X	X	X	X	X	X	
Suporte a conversão de chaves primárias	X	X	X	X	X	X	

Suporte a conversão de *check constraints*		X	X	X	X		
Suporte a conversão de visões	X	X	X	X	X		
Suporte a conversão de dados	X	X	X	X	X	X	X
Suporte a conversão de colunas auto-incremento	X	X	X			X	
Suporte a conversão de chaves únicas		X	X	X	X	X	
Suporte a conversão de chaves estrangeiras	X	X	X	X	X	X	
Suporte a conversão de índices	X	X	X	X	X	X	
Suporte a execução de *scripts* pré-conversão e pós-conversão				X			
Suporte a conversão de campos BLOB/CLOB		X		X	X		
Suporte a alteração de tipos de dados	X	X	X			X	
Suporte a alteração de *default* de colunas	X	X			X	X	
Suporte a caracteres *unicode*	X			X			
Suporte a conversão de nomes de objetos	X	X				X	
Suporte a conversão de comentários de objetos		X					
Suporte a conversão de procedimentos armazenados, funções e gatilhos		X					
Suporte a controle de tamanho máximo de tipos de objetos entre banco de dados distintos		X	X				
Suporte a conversão total de múltiplos bancos de dados relacionais (conversão de dados e metadados)							
Ferramenta de código aberto ou licença gratuita							X

2.4 Porque desenvolver uma ferramenta

Dentre todas as ferramentas analisadas nenhuma delas suporta múltiplos bancos de dados relacionais, ou seja, as fontes de origem e destino são dependentes de uma quantidade de bancos pré-estabelecida pelos fornecedores, exceto a ferramenta *open-source* analisada. Porém esta somente a faz conversão de dados, não suportando a conversão de metadados nativamente.

Pode-se assim confirmar a necessidade do desenvolvimento de um aplicativo que tenha licença de código aberto e que seja independente dos bancos de origem e destino utilizados. Também é objetivo desta ferramenta ser de fácil utilização e adaptar-se para as mais inusitadas situações.

2.5 Características fundamentais para esta ferramenta

O objetivo do projeto é oferecer suporte à conversão de tabelas, visões, índices, chaves primárias, chaves estrangeiras, colunas de auto-incremento, *check constraints*, *unique constraints*, e obviamente suporte a conversão dos dados [Elmasri *et al*., 2005].

Para conversão dos dados a ferramenta de migração deve trabalhar com a heterogeneidade de dados entre diferentes bancos, visto que o propósito de uma utensílio deste tipo é a conversão das informações SGBD entre diversos produtos [Gárcia *et al*., 2001]. Além de trabalhar sobre as propriedades ACID [Date, 2004] durante as conversões.

Como nem todos os bancos de dados implementam suporte aos tipos de dados definidos pelo padrão ANSI-SQL é necessário que esta ferramenta também possibilite o mapeamento entre estes tipos de dados da fonte para o destino, isto é, o banco de dados gerado na conversão pode conter tipos diferentes do original. Também fica válida a mesma regra para o *default* de colunas na especificação de uma tabela [ANSI-SQL 92; ANSI-SQL 99; Ferraça, 2007].

Os sistemas gerenciadores de bancos de dados existentes possuem limitações diferentes entre si, e isto inclui o tamanho máximo de campos (e.g. NUMERIC, VARCHAR, entre outros), o que torna necessário ser tratado durante a conversão.

Também existe a maneira como são tratados os identificadores delimitados, tais como nome de tabelas e colunas. Se estes devem ou não conter aspas nas suas declarações e se seus nomes devem ser em maiúsculo ou minúsculo.

A criação de nomes de índices e *constraints* devem respeitar o escopo do banco de dados, pois em alguns bancos como o Firebird, isto é global, ou seja, os nomes de objetos

devem ser únicos em todo o banco de dados. Já no ASA o escopo é pelos objetos ao qual os índices e *constraints* pertencem, isto é, pode haver por exemplo, dois índices com o mesmo nome mas para tabelas diferentes [Ferraça, 2007].

É de interesse que a ferramenta também suporte a execução e *scripts* pré-conversão e pós-conversão da estrutura dos metadados e dos próprios dados, caso seja necessário a execução de alguma operação específica, oriundas da necessidade de cada caso de conversão.

O suporte para conversão de procedimentos armazenados, funções e gatilhos não são tratados na ferramenta a ser desenvolvida, pois cada banco implementa na sua própria linguagem, o que torna inviável para este trabalho. Mesmo conversão para SGBD de mesmo tipo não é interessante, pois o objetivo é justamente migrar bases com gerenciadores diferentes [Ferraça, 2007].

Existem mais dois itens que devem ser considerados durante o processo de conversão. O primeiro deles é a maneira com que os campos BLOB e CLOB [Elmasri *et al*., 2005] serão convertidos de um SGBD para outro, pois cada produto acaba implementando diferentes variações deste tipo de dados [Ferraça, 2007]. E o segundo é o conjunto de caracteres, que segundo Borrie (2006) são uma coleção de símbolos que compõem um alfabeto ou silabário (e.g. ISO8859-1, Windows-1256, Unicode, etc.), também deve ser referenciado durante a conversão.

Com os quesitos levantados acredita-se que a ferramenta poderá ser útil e realmente flexível para o auxílio em tarefas de conversão de base de dados heterogênias, encaixando-se em situações como ocorridas com a Núcleo Sistemas sobre a conversão do seu ERP [Ferraça, 2007], principalmente por não limitar os SGBD que poderão ser utilizados, tanto na origem dos dados quanto no destino. Além de incluir os recursos importantes de ferramentas já existentes no mercado atual.

3 PROPOSTA DE DESENVOLVIMENTO

Este capítulo irá apresentar a definição do projeto, mostrando a idéia para o funcionamento do sistema, isto é, como o aplicativo pretende ler bancos de dados diferentes e como serão gerados comandos DML e DDL para diferentes produtos.

Serão utilizados alguns diagramas UML para a apresentação da forma de desenvolvimento do aplicativo. Serão referenciadas as principais telas que estarão disponíveis para o usuário final através de um protótipo do projeto. Também, será apresentada a ferramenta escolhida para o desenvolvimento do aplicativo, incluindo o driver que será utilizado para dispor conexões com as bases de dados.

E por fim, o capítulo irá apresentar quais são as exceções do escopo do projeto, isto é, as restrições aplicadas à utilização do sistema. Será também apresentado como se pretende distribuir o software depois do seu desenvolvimento e qual será a licença para uso deste, já que se tratará de um sistema de código aberto.

3.1 Definição do aplicativo

No mundo dos bancos de dados existem bancos de dados relacionais, objeto-relacional, orientados a objetos, entre outros. Entretanto o escopo desejado para o desenvolvimento do aplicativo será apenas bancos de dados relacionais, pois é atualmente o mais utilizado no mercado [Date 2004].

Para o desenvolvimento do aplicativo, pretende-se utilizar o idioma inglês, tanto para definição de nome de classes, métodos e entre outros, quanto para a interface com o usuário, pois no mundo da computação este é o idioma mais utilizado, permitindo que outros desenvolvedores possam dar continuidade ao projeto.

O funcionamento interno do aplicativo, isto é, como o sistema irá funcionar internamente será explicado nos próximos sub-tópicos.

3.1.1 Funcionamento do motor do sistema

Durante o desenvolvimento do Estágio Curricular I e II do Curso de Sistemas de Informação foi necessário o desenvolvimento de um aplicativo para a conversão de bancos de dados de *Firebird* para *Sybase Adaptive Server Anywhere*. Para o desenvolvimento deste foi necessário criar consultas SQL para lerem as tabelas de sistema do banco de dados em ASA obtendo as informações necessárias do conjunto de metadados da base de origem, e através destas informações desenvolver os comandos DDL para o banco de dados de destino em *Firebird*, conforme a sua sintaxe [Ferraça, 2007].

Teve-se a idéia neste ponto para a criação de um aplicativo que pudesse fazer a conversão independente da base de origem, sendo apenas necessário alterar as consultas das tabelas de sistema de acordo com cada SGBD, mas que retornasse as informações elementares para o aplicativo.

Assim pretende-se informar para o sistema um conjunto de consultas com informações pré-estabelecidas para ser retornadas, necessárias para a conversão de um banco de dados para outro. Estas informações serão lidas de *templates*[4] que conterão as consultas pré-estabelecidas para cada tipo de banco de dados configurado. A definição dos *templates* conterá o que deve ser retornado e quais os parâmetros que o sistema fornecerá para obter das informações, por exemplo, nome e colunas de tabela para ser convertida.

Para a geração da sintaxe correta dos comandos DDL na base de destino, o conversor desenvolvido no estágio curricular não teve problemas, pois era somente gerado para a sintaxe do *Firebird*. Entretanto como a proposta de desenvolvimento do sistema é não haver restrição

---

[4] *Templantes* são elementos de transformação que possuem argumentos para produzir uma constante como resultado de um processamento por intermédio de uma ferramenta [Vandevoorde *et al*., 2005].

de utilizar um banco de dados específico, pois os comandos a serem gerados devem ser diferentes para cada tipo de SGBD.

Como solução será adotada a criação de mais arquivos *templates* contendo a sintaxe de cada sistema gerenciador, incluindo parâmetros nestes *templates* para serem substituídos pelas informações lidas pelas consultas efetuadas nas tabelas de sistemas dos bancos de dados de origem. Para exemplificar a idéia, é visto no exemplo 1 a criação de uma chave estrangeira na SGBD *Firebird* e no exemplo 2 como ficaria um *template* para este comando.

```
ALTER TABLE tabela_origem
  ADD CONSTRAINT fk_tab_origem_tab_destino FOREIGN KEY (coluna01, coluna02)
    REFERENCES tabela_destino (coluna01, coluna 02) ON DELETE CASCADE
  USING INDEX x_tab_origem_tab_destino;
```

Exemplo 1: Criação de chave estrangeira no *Firebird*.

```
ALTER TABLE <table_origin_name>
  ADD CONSTRAINT <foreign_key_name> FOREIGN KEY (<columns_list_origin>)
    REFERENCES <table_destination_name> (<columns_list_destination>) <action>
  USING INDEX <index_name>;
```

Exemplo 2: *Template* para criação de uma chave estrangeira no *Firebird*.

Os parâmetros, observados dentro dos sinais de menor (“<”) e maior (“>”), serão preenchidos pelo motor do aplicativo. As informações necessárias serão retornadas por uma determinada consulta, responsável por obter as informações de chaves estrangeiras. Esta consulta poderá ser fornecida por qualquer SGBD que esteja configurado para uso pelo migrador.

As conversões dos dados serão feitas por operações DML, para inserirem os dados do banco de origem para o banco de destino. Serão também, nesta situação, padronizadas as consultas para obtenção de informações, de como, por exemplo, a quantidade de colunas e tipo. E para a geração dos comandos de inserção a utilização também de um *template* como visualizado no exemplo 3.

```
INSERT INTO <table_name> (<columns_list>) VALUES (<values_list>);
```

Exemplo 3: *Template* para inserção de dados.

3.1.2 Configurações do aplicativo

Outros tipos de parâmetros necessários serão a nível de banco de dados de origem e destino para flexibilizar a conversão entre eles. Estes parâmetros serão únicos para cada banco de dados configurado. Por exemplo, para um SGBD em ASA haverá configurações diferentes de um SGBD em *Firebird*. E estes parâmetros serão também diferentes na configuração de um produto como sendo origem dos dados ou destino.

Através da experiência obtida no desenvolvimento do trabalho de estágio [Ferraça, 2007] foi possível identificar antes do início do desenvolvimento do projeto situações que podem necessitar de parametrização. As configurações de origem são os seguintes parâmetros:

- **Tabelas para ignorar**: Será possível configurar tabelas não necessárias para a migração, tais como tabelas de sistemas;
- **Palavras para ignorar**: Alguns bancos gravam a definição inteira de uma visão ou das *constraints* do tipo *check*, o *namespace*, como por exemplo, o ASA. E isto pode não ser compatível em outros bancos de dados, sendo necessários ignorar palavras dentro desta, como por exemplo, “DBA.”;
- **Migrar opcionalmente visões e *constraints* do *tipo check***: Como recém mencionado, alguns bancos gravam a definição inteira de visões e *constraints* do tipo *check* no banco de dados, e como estas podem utilizar funções específicas de cada plataforma, talvez não seja interessante desenvolver a migração destas opções, criando um parâmetro para tornar opcional esta migração.

E para as configurações de destino são os seguintes parâmetros:

- **Conversão de nomes de objetos**: Identificar se as conversões dos nomes de objetos serão maiúsculas, minúsculas ou sem nenhuma modificação. Por

exemplo, a tabela chamada “Maycon” poderá ser convertida para “MAYCON”, “maycon” ou permanecer a original;

- **Tamanho máximo para nomes de objetos**: Os bancos de dados possuem limites de tamanho máximo de nomes de objetos diferentes um do outro, para tal identificar qual é o tamanho máximo do banco de destino. Pretende-se utilizar esta função para nomes de *constraints* e índices possivelmente criados durante a conversão;
- **Ignorar identificadores delimitados na conversão de visões**: Pretende-se criar esta opção para caso de visões criadas com identificadores delimitados, ignorando ou não as aspas duplas;
- **Substituição de tipos de dados**: Foi identificada no desenvolvimento do estágio curricular a necessidade da conversão de tipos de dados CHAR para VARCHAR, pois naquela situação semanticamente o tipo CHAR e VARCHAR do ASA eram iguais somente ao tipo de dados VARCHAR do *Firebird*, havendo esta necessidade;
- **Substituição do valor *default* colunas**: Foi também identificado no estágio curricular que a definição de alguns valores *default* para a criação de colunas eram sintaticamente diferentes umas das outras, mas como mesma semântica, como foi o caso do “CURRENT DATE” do ASA que retorna a data do dia, mas no *Firebird* o uso é como “CURRENT_DATE”;
- **Tamanho máximo para a precisão de colunas numéricas**: Colunas como NUMERIC e DECIMAL podem possuir precisão diferentes entre os tipos de SGBD, sendo interessante a sua parametrização;
- **Troca do caractere delimitador de comandos**: Alguns sistemas gerenciadores, como o *Firebird*, não permitem receber mais de um comando SQL, DML ou DDL por vez. E na execução de comandos DDL para a criação de procedimentos armazenados é necessário intercalar o caractere de delimitação, pois o caractere ponto-e-vírgula (“;”) já é utilizado dentro do corpo da função, sendo necessário outro identificador para real final do comando;

- **Padrão de prefixos para nomes de objetos**: Pretende-se fazer a padronização para a geração de nomes para chaves primárias, chaves estrangeiras, chaves únicas, *constraints* do tipo *check*, *sequences*, gatilhos do tipo *before insert* (necessários para a criação de colunas auto-incrementos em alguns bancos), índices únicos e não únicos, e para tal pretende-se parametrizar estes padrões.

Pretende-se fazer conversões de campos do tipo BLOB e CLOB apenas copiando o conteúdo do campo de origem para o campo de destino. Não haverá problema se os fabricantes seguirem o que especifica o padrão ANSI-SQL.

Para a conversão de conjunto de caracteres também não se pretende fazer nenhuma conversão, porque geralmente a conversão é feita com o mesmo conjunto de caracteres do banco de origem para o banco de destino [Ferraça, 2007].

3.2 Modelagem do aplicativo

Para a modelagem do aplicativo serão apresentados alguns diagramas da linguagem UML. Esta linguagem é utilizada para especificação de sistemas orientados a objetos na qual define a visualização, construção e documentação de um *software*. Esta ferramenta irá ajudar a simplificar o processo de especificação do aplicativo proposto, por dar diferentes visões [Medeiros, 2006].

Serão utilizados alguns diagramas para definir o comportamento externo do sistema (casos de uso e atividades), dando uma visão da interação do usuário (conhecido como um ator). Será também visto um diagrama para definir como a estrutura (diagrama de classes) do aplicativo será construída. E por fim serão vistos diagramas para definir o comportamento interno do sistema (seqüências), definindo como os processos ocorreram entre as estruturas do aplicativo, dando uma visão dinâmica do mesmo [Larman, 2000].

Outros artefatos da UML não foram utilizados porque a necessidade do sistema foi suprida pelos diagramas que serão apresentados. Por exemplo, o diagrama de pacotes não foi utilizado, pois o sistema não irá conter grandes classes, não justificando agrupar estas classes

em pacotes. O diagrama de comunicação (interação) para comportamento interno não foi utilizado porque pelo tamanho do aplicativo, o diagrama de seqüências é suficiente para descrever interação entre as classes. O diagrama de distribuição, responsável por apresentar a distribuição de hardware do sistema, não foi necessário porque este utensílio será desktop, não havendo esta necessidade. O diagrama de estados não foi utilizado porque o sistema somente irá possuir dois estados durante um processo de conversão, e estes estados serão de convertendo e não convertendo, apontados através do atributo booleano "*Converting*" da classe principal "*TMain*" (diagrama de classes, página 45) [Medeiros, 2006].

3.2.1 Requisitos do sistema

Para definição dos requisitos do sistema serão utilizados diagramas de casos de uso, visando descrever os objetivos que um ator externo, neste caso o usuário, tem com o sistema [Medeiros, 2006]. Primeiramente são apresentados na figura 3 os objetivos do usuário com as configurações do aplicativo, e na figura 4, os objetivos do usuário com a utilização do programa.

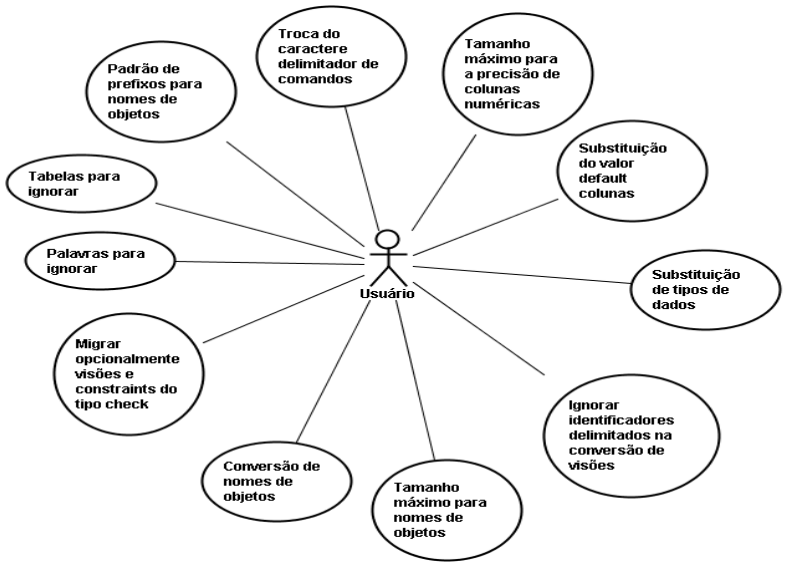


Figura 3: Requisitos para a configuração do sistema.

Fonte: Própria.

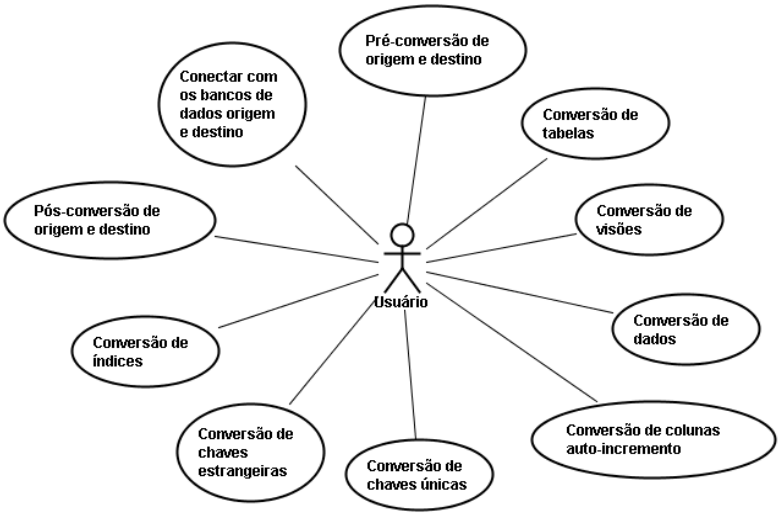


Figura 4: Requisitos para a utilização do sistema.

Fonte: Própria.

3.2.2 Fluxo de atividades do sistema

Um diagrama de atividades é utilizado para descrever o fluxo das atividades, ou seja, a lógica de execução, ou ainda a série de passos [Medeiros, 2006]. O diagrama é apresentado na figura 5.

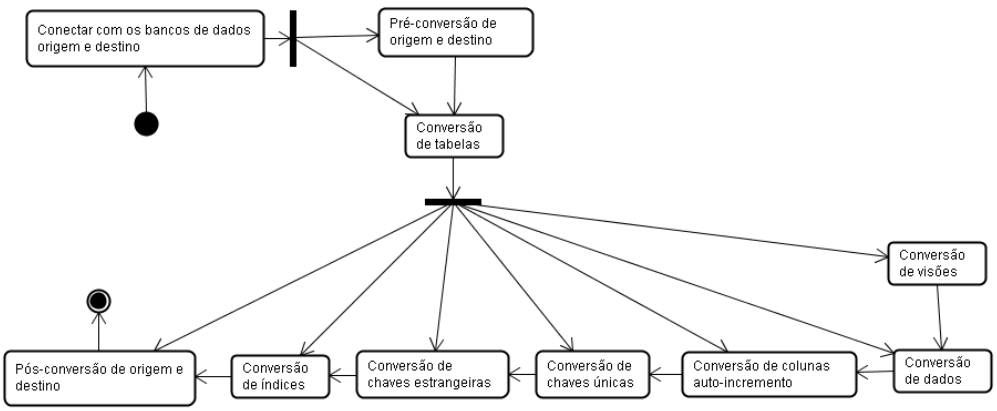


Figura 5: Fluxo de atividades do sistema.

Fonte: Própria.

Como visto no diagrama de atividades, primeiramente o usuário deve conectar-se com os bancos de dados de origem e destino, ficando opcional a operação pré-conversão. O próximo passo é efetuar a conversão das tabelas para prosseguir com qualquer outra operação, que por sua vez, estas outras operações, são opcionais durante todo o restante do processo.

3.2.3 Modelo conceitual

O modelo conceitual será definido pelo diagrama de classes, utilizado para descrever as classes que formam a estrutura do aplicativo e quais são as suas relações [Medeiros, 2006], sendo o diagrama referente ao projeto visualizado na figura 6.

Cada classe terá as suas respectivas utilidades e estas são descritas a seguir:

- ***TConsistency***: Classe responsável por implementar métodos de controle de consistência, geração e controle de nomes dos objetos do banco de dados;
- ***TConversionDDL***: Irá prover métodos para a conversão DDL;
- ***TConversionDML***: Responsável pela conversões DML;
- ***TDMSystem***: Classe responsável pelas conexões com as bases de dados, centralizando tudo que entra e sai das bases. Os atributos definidos nesta classe são componentes da linguagem *Delphi* e serão utilizados para comunicação com os SGBD através da biblioteca BDE;
- ***TFMain***: Classe do sistema que fornecerá a principal interface de comunicação com o usuário. Também será responsável por fazer respectivas chamadas a outros métodos de outras classes. Os atributos do tipo "*TSpinEdit*" e o "*TCheckBox*" são componentes visuais do Delphi, utilizados para especificar as propriedades vistas página 53;

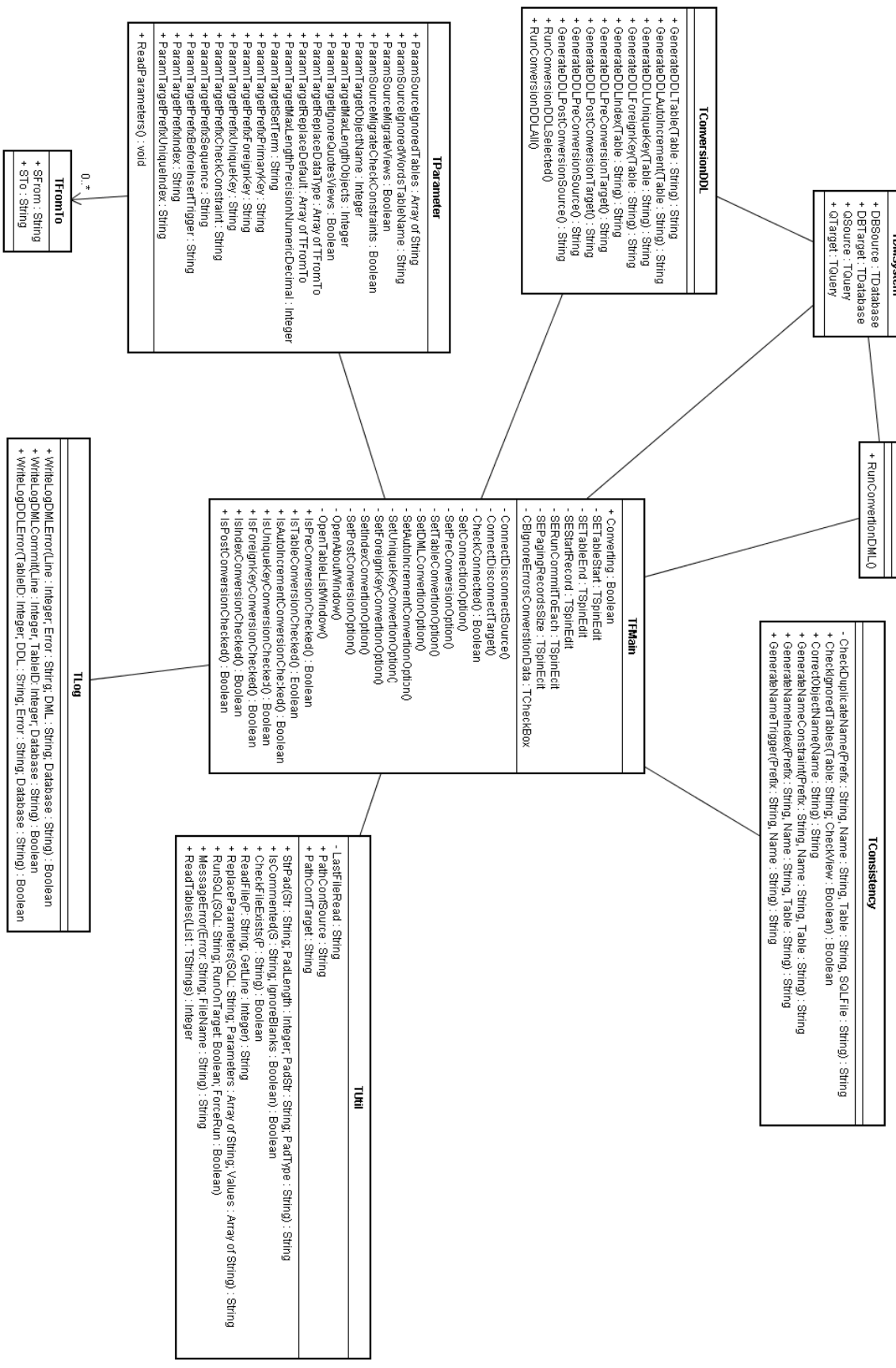


Figura 6: Modelo conceitual do sistema.

Fonte: Própria.

- ***TLog***: Implementará métodos para a gravação de *logs*[5] de dados para os erro nas operações de conversão de DML e DDL, além de prover *log* para as operações DML dos registros que sofrem um comando "COMMIT" no banco;
- ***TParameter***: Responsável pelos parâmetros do sistema;
- ***TFromTo***: Na verdade não se trata especificamente de uma classe e sim de um "*Record*"[6] utilizado pelo projeto para definir uma lista "de" e "para" em alguns parâmetros da classe "*TParameter*";
- ***TUtil***: Classe que implementará métodos úteis para outras classes do projeto.

Todas as classes, exceto as "*TDMSystem*", "*TFAbout*", "*TFMain*", "*TFTableList*", "*TToFrom*" serão automaticamente instanciadas na inicialização da aplicação e finalizadas ao término da execução do sistema, através do recurso do *Delphi*, "*initialization*" e "*finalization*" [Leão, 2001]. Isto se fará assim porque estas classes terão a utilidade de servir como auxílio uma da outra, não havendo a necessidade de instanciá-las somente ao seu uso.

3.2.4 Interação entre as classes do sistema

Os diagramas de seqüência ilustram cada cenário dos casos de uso essenciais. Os eventos que os atores externos geram e as respostas do sistema são ênfase neste diagrama [Medeiros, 2006].

Para mostrar os diagramas de seqüência de atividades não foram utilizadas todas as classes e também somente foram referenciadas as principais ações. Isto para facilitar a leitura dos diagramas, pois a maioria dos métodos e atributos são para controle interno, como por

---

[5] *Log* é um arquivo onde ficam registrados os eventos ocorridos com um sistema [Borrie, 2006].

[6] *Records* (ou registros em português) é um recordo da linguagem *Object Pascal* e são utilizados para agrupar vários atributos de diferentes tipos (ou não), criando um novo tipo de dados considerado como heterogêneo [Leão, 2001].

exemplo, o método “*SetConnectionOption*” da classe “*TFMain*” responsável por trocar a tela para a página de conexão.

A figura 7 mostra como procederá internamente o caso de uso “*Conectar com os bancos de dados origem e destino*”. Primeiramente será feita a conexão com as bases de dados de origem e destino, após isso o aplicativo irá ler os parâmetros do sistema. Também ocorre nesta seqüência típica a desconexão das bases de dados.

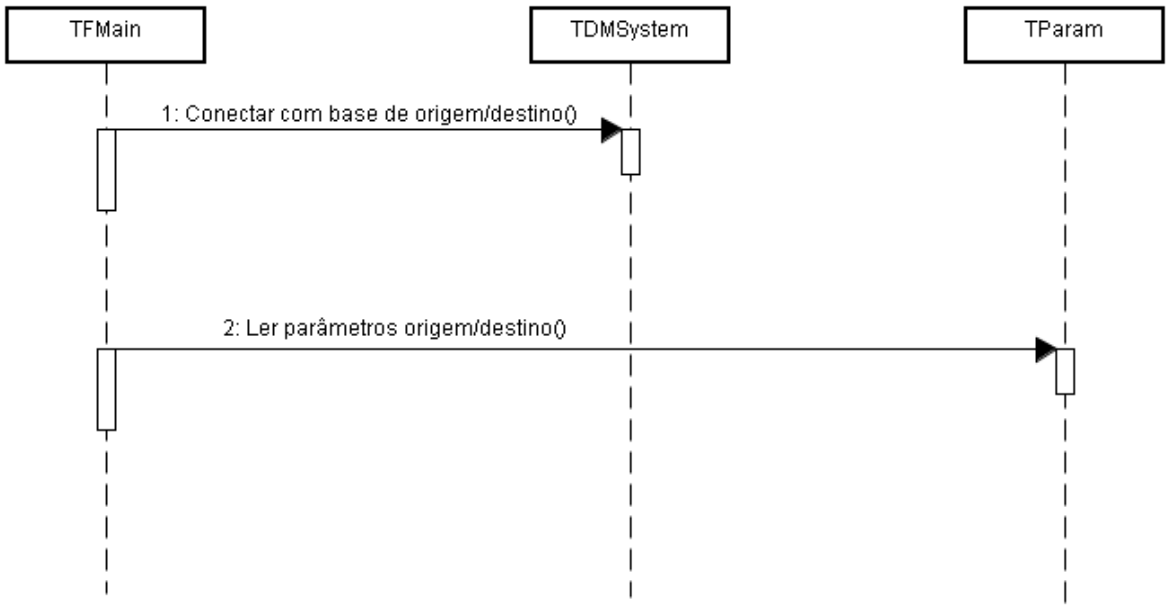


Figura 7: Seqüência para “*Conectar com os bancos de dados*”.
Fonte: Própria.

O caso de uso “*Pré-conversão de origem e destino*” é visto na figura 8. E este mesmo processo serve para o caso de uso “*Pós-conversão de origem e destino*”. Primeiramente a classe “*TFMain*” irá disparar a requisição de conversão para a classe “*TConversionDDL*”. Esta por sua vez irá obter os comandos para execução através da leitura dos *templates* e por fim submeter para execução nos bancos de dados através da classe “*TDMSystem*”.

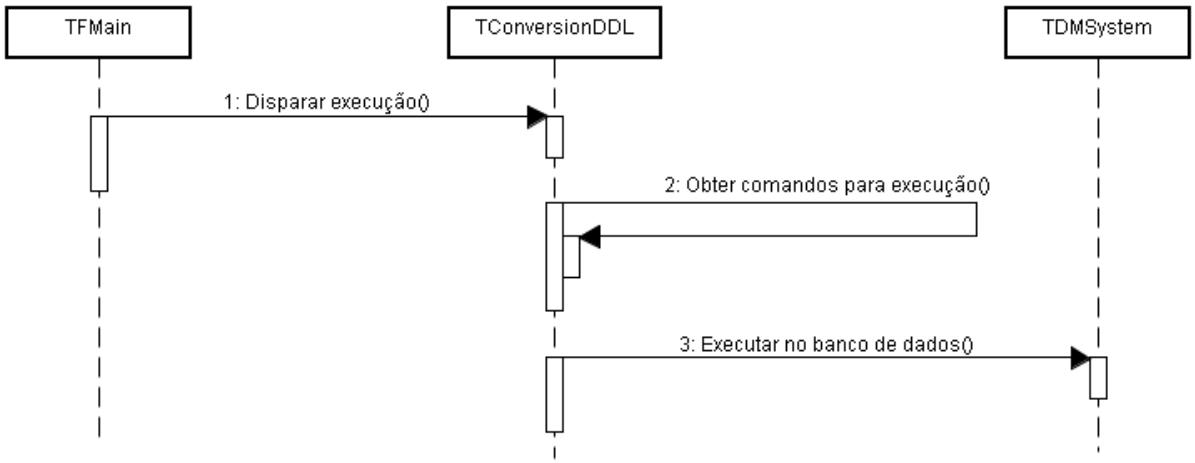


Figura 8: Seqüência para "*Pré-conversão de origem e destino*".

Fonte: Própria.

A figura 9 referente ao caso de uso de "*Conversão de tabelas*" mostra como procede a comunicação entre as classes para este processo. Entretanto as mesmas operações também se aplicam para os casos de uso de "*Conversão de visões*", "*Conversão de colunas auto-incremento*", "*Conversão de chaves únicas*", "*Conversão de chaves estrangeiras*" e "*Conversão de índices*".

Primeiramente a classe "*TFMain*" disparará a execução para a classe "*TConversionDDL*", após isto serão obtidos as informações sobre os metadados do banco de origem, em seguida será gerado os comandos conforme a sintaxe da base de destino através dos arquivos *templates* referentes e por fim a execução destes comandos na base de dados através da classe "*TDMSystem*".

Durante o processo de conversão se algum erro ocorrer um arquivo de *log* é gravado, chamado pelo método "*WriteLogDDLError*" da classe "*TLog*". Isto inclui as operações executadas pelo no caso de uso "*Pré-conversão de origem e destino*" e "*Pós-conversão de origem e destino*".

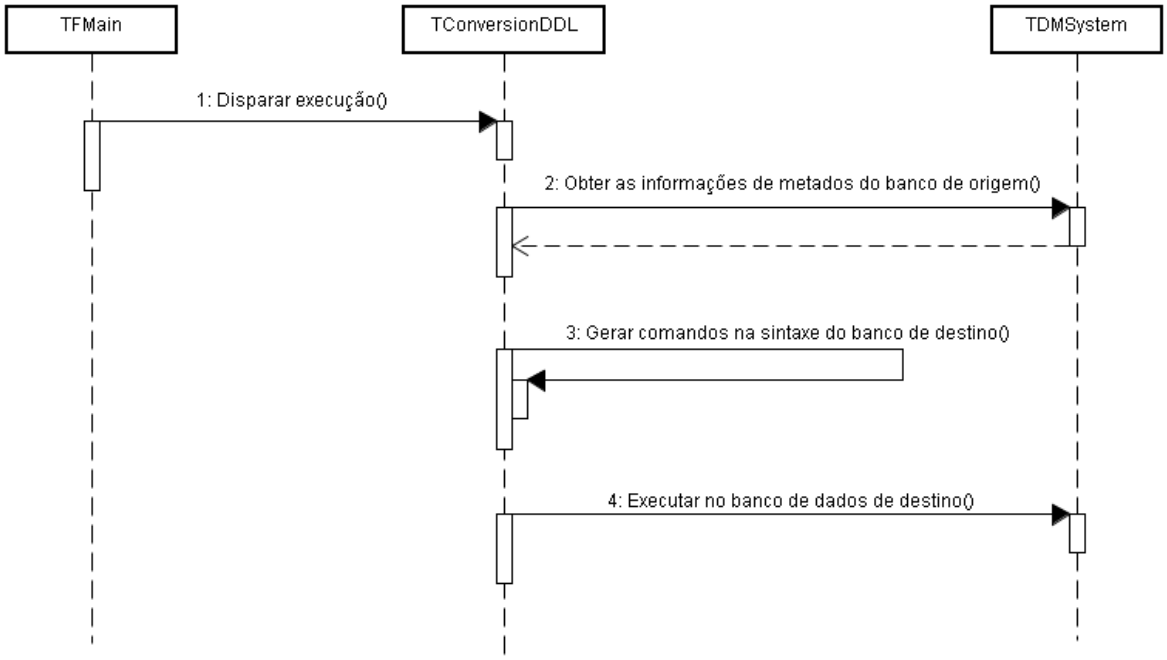


Figura 9: Seqüência para "*Conversão de tabelas*".

Fonte: Própria.

A seqüência de eventos para o caso de uso de "*Conversão de dados*" é exemplificado na figura 10. Primeiramente é disparado pela classe "*TFMain*" a operação para a classe "*TConversionDML*", após é lido as informações dos dados tabela de origem a partir da classe "*TDMSystem*". Conseqüentemente é gerado os comandos com a sintaxe do banco de dados de destino e enviado para execução novamente para a classe "*TDMSytem*".

Também durante o processo de conversão se algum erro ocorrer um arquivo de *log* é gravado, chamado pelo método "*WriteLogDMLError*" da classe "*TLog*". Neste *log* será informada qual a linha que ocorreu o problema, para que em conjunto com o *log* criado pelo método "*WriteLogDMLCommit*", onde é informado a última linha que obteve sucesso na execução de um comando "COMMIT", seja possível continuar a execução após um erro.

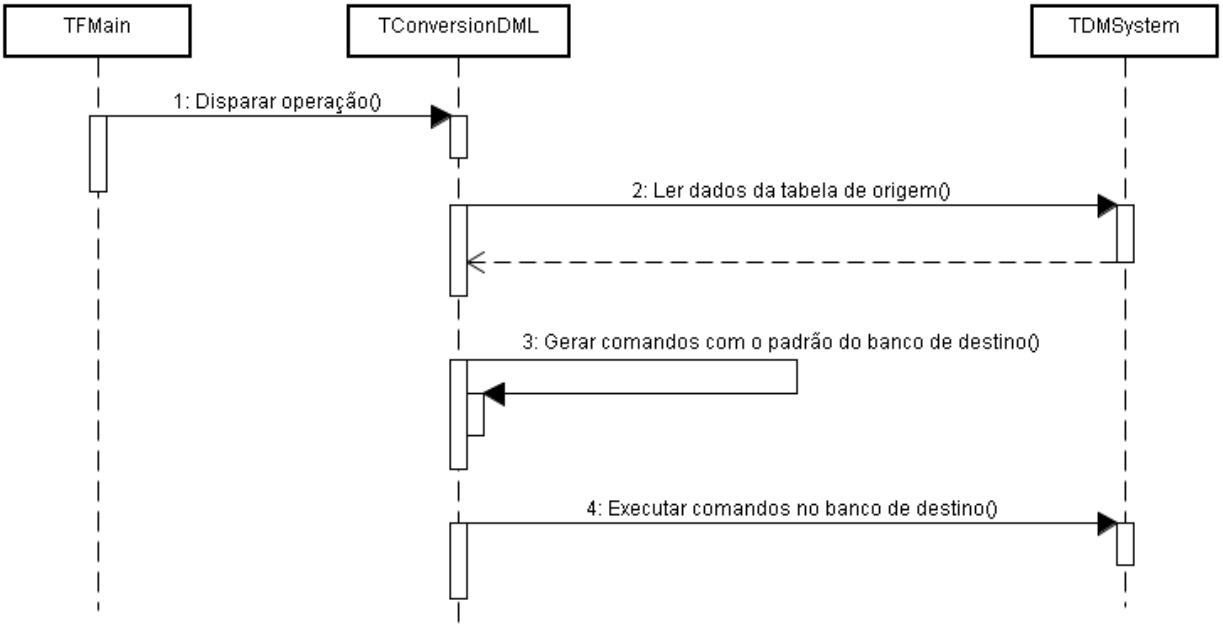


Figura 10: Seqüência para "*Conversão de dados*".

Fonte: Própria.

3.3 Protótipo do aplicativo

Para a operação de conexão com os bancos de dados de origem e destino foi desenvolvidas telas como apresenta a figura 11 na ferramenta *Delphi*, solicitando qual será o SGBD de origem e qual o SGBD para o qual será feita a conexão. E o mesmo aconteceria para o banco de destino, especificando qual será o SGBD de destino e qual é a fonte ODBC do banco de dados vazio para conexão de destino.

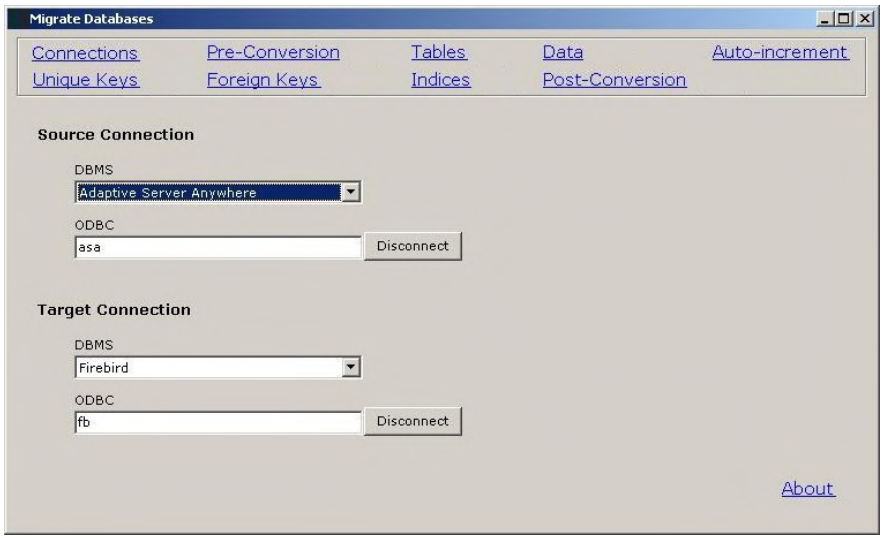


Figura 11: Tela do protótipo para conexão.

Fonte: Própria.

Para as operações de pré-conversão e pós-conversão pretende-se construir uma tela como especificado na figura 12. Haverá uma opção de escolha para onde as operações serão executadas, na origem ou no destino e conforme a escolha o *template* específico será lido.

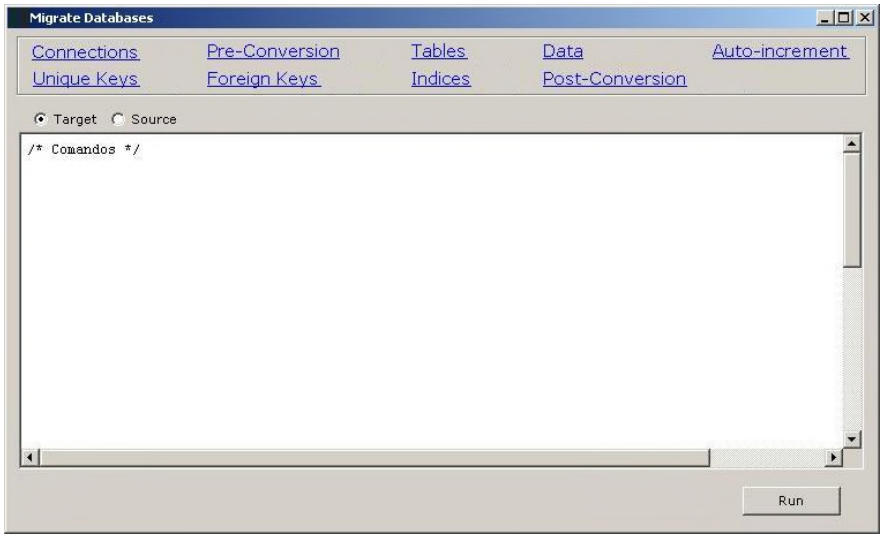


Figura 12: Tela do protótipo para pré-conversão e pós-conversão.

Fonte: Própria.

Para as operações de conversão de tabelas (incluindo chaves primárias e *constraints* do tipo *check*), visões, colunas auto-incremento, chaves únicas, chaves estrangeiras e índices a

tela será semelhante a figura 13. Na esquerda do protótipo serão listadas todas as tabelas que poderão ser migradas, e no lado direito os comandos referentes a estas. Caso selecionado alguma tabela, apenas a migração desta será feita.

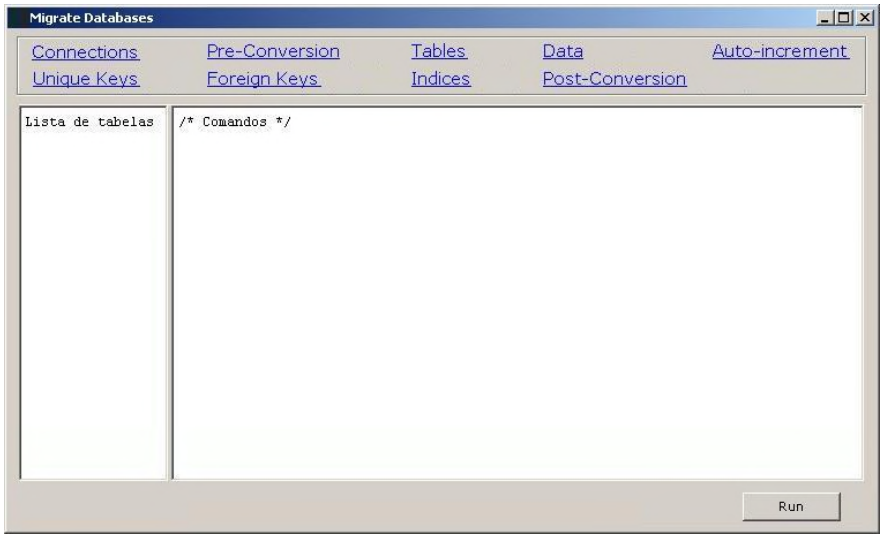


Figura 13: Tela do protótipo para conversão de metadados.

Fonte: Própria.

E por fim, para a migração dos dados pretende-se construir uma tela como visualizada na figura 14, e nela serão contido as seguintes opções:

- **“*Tables n to n*” (“Tabelas da *n* até *n*”)**: Caso queira-se fazer uma migração individual das tabelas é necessário informar aqui um intervalo de códigos, no qual é possível obter na opção “*Table list*” (“Lista de Tabelas”), onde será especificada a lista de tabelas disponíveis para a conversão e os seus respectivos códigos;
- **“*Start first table since the record n*” (“Iniciar primeira tabela a partir do registro n”)**: Para falhas em conversões, também será possível iniciar uma conversão inicial de determinado registro de uma tabela;
- **“*Run "Commit" to each*” (“Executar “COMMIT” a cada *n* registros”)**: Especifica ao conversor a cada quantos registros convertidos deve ser efetuado um comando “COMMIT” para efetivar os dados já convertidos. Será criado um arquivo de *log* para cada “COMMIT” executado, podendo assim em caso

de falha, corrigir o problema e continuar do último registro gravado, especificando na opção anterior (“*Start first table since the record n*”);

- ***“Paging records to each”* (“Paginar a cada *n* registros”)**: O componente de dados de acesso que será utilizado, o BDE possui um limite do número de registros que pode ser carregado em memória, este limite é cerca de 357000 (dependo o tamanho das linhas carregadas) [Leão, 2001]. Então internamente será efetuado um controle de paginação dos registros a serem convertidos, isto significa que o aplicativo irá carregar por vez a quantidade de registros especificada;
- ***“Ignore errors”* (“Ignorar erros”)**: Irá especificar ao sistema para ignorar qualquer erro durante o processo de conversão. Esta opção deverá ser usada com cuidado porque os erros não serão reportados para o usuário e somente gravados no *log* de erros ocorridos no processo de conversão. Isto será útil para forçar um processo de atualização.

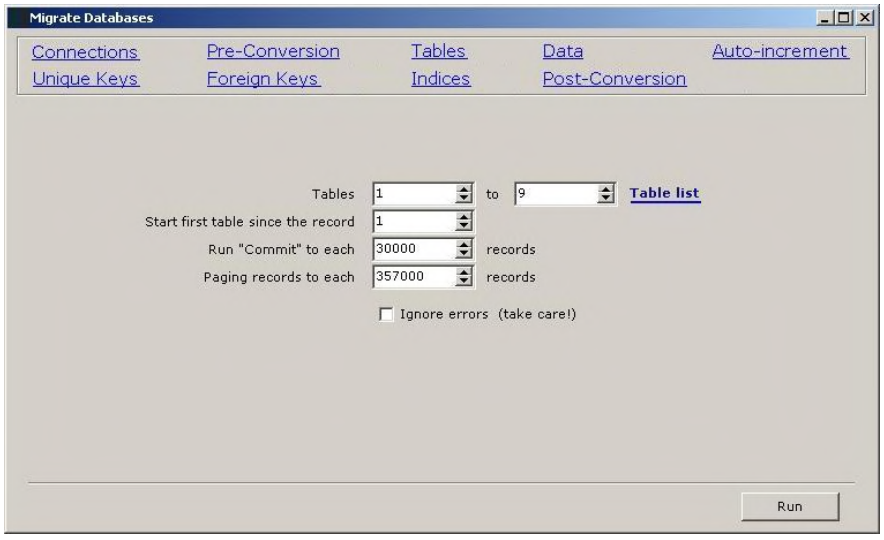


Figura 14: Tela do protótipo para conversão de dados.

Fonte: Própria.

3.4 Ambiente para desenvolvimento

Para o desenvolvimento do *Migrate Databases* pretende-se utilizar uma ferramenta que possibilite a utilização de uma linguagem orientada a objetos, já que foram utilizados diagramas UML para a definição do aplicativo. É também necessário que a ferramenta utilizada ofereça conexão com diversos bancos de dados diferentes para cumprir com o objetivo do trabalho.

3.4.1 Ferramenta para desenvolvimento

A ferramenta escolhida para o desenvolvimento do aplicativo proposto no trabalho de conclusão de curso foi o *Delphi*. A escolha desta foi realizada por maior familiaridade e a possibilidade de desenvolver o aplicativo proposto de forma orientada a objetos. Entretanto, nada impediria que o sistema fosse escrito em outra linguagem que possibilitasse a conexão com múltiplos bancos de dados, como por exemplo, *Powerbuilder* [Wood, 1995] e *Java* [Horstmann, 2005]. Entretanto existem algumas vantagens no desenvolvimento da aplicação em *Delphi*, possivelmente talvez também presentes em outras ferramentas.

O *Delphi* é uma ferramenta IDE, isto é, possui um ambiente integrado para desenvolvimento de *softwares*, utilizando uma linguagem orientada a objetos de fácil assimilação, o *Object Pascal* (conhecido também como *Delphi Language*). O desenvolvimento é rápido com aplicações desenvolvidas em *Delphi* pela existência de uma forte quantidade de componentes prontos para utilização em sua biblioteca. Dentre estes existem diversos componentes para conexão com bancos de dados, incluindo o DBX e o BDE. Além de ser possível criar com o *Delphi* aplicações que rodem nativamente na plataforma *Windows* através da biblioteca conhecida como VCL [Leão, 2001].

Por se tratar de um sistema que será *open-source*, poderá ser utilizada a versão totalmente gratuita do *Delphi*, o Turbo *Delphi*, disponível no endereço http://www.turboexplorer.com/delphi. Esta versão, além de ser livre de custos, também permite até o desenvolvimento de produtos comerciais.

3.4.2 Acesso aos bancos de dados

As conexões com os sistemas gerenciadores de bancos de dados serão estabelecidas pelo *driver* BDE, pois este permite a utilização da fonte de dados ODBC da *Microsoft,* provendo acesso a qualquer sistema gerenciador de banco de dados que disponibilize um *driver* compatível.

Isto é possível por que a implementação do ODBC permite comunicar-se com diversas interfaces de armazenamento de maneira transparente, funcionando como uma camada intermediária entre o sistema gerenciador de banco de dados e a aplicação. O *driver* é responsável por fazer tradução dos distintos protocolos que cada fabricante de banco de dados faz. Isto habilita o programa cliente comunicar-se com mais de um servidor de bancos de dados sem grandes alterações.

O ODBC possibilita até a persistência de gravação em arquivos textos, obviamente com restrições de funcionalidade do *driver*. Entretanto a tradução de protocolo do banco de dados para a aplicação impacta diretamente na performance do sistema, isto é, a comunicação com o banco de dados se fará mais lentamente. E a utilização do ODBC implica na existência de *drivers* ODBC para os bancos que se deseja acessar [Leão, 2001].

3.5 Restrições do escopo

Pretende-se desenvolver no aplicativo as operações de conversões de tabelas, visões, índices, etc., pois o escopo definido do projeto foi bancos de dados relacionais [Elmasri *et al*., 2005]. Entretanto a arquitetura do aplicativo permite que seja possível a conversão de qualquer produto que possa através de uma consulta SQL devolver um conjunto de dados com as informações referentes aos metadados. Obviamente estas consultas devem ser aceitas pelo sistema, e pelo *driver* utilizado internamente na aplicação, o BDE e por sua vez o ODBC.

Apesar do sistema permitir que outros tipos de bancos que atendam as necessidades recém especificadas, os bancos de dados relacionais também deve atender estas necessidades, isto é, um banco de dados que não possua driver ODBC ou do qual não seja possível obter informações dos metadados através de consultas SQL, não pode ser utilizado.

3.6 Distribuição e licença

Como mencionado no início deste capítulo (página 36) pretende-se desenvolver o sistema no idioma inglês, pois no mundo da computação é o mais utilizado, permitindo que outros desenvolvedores possam dar continuidade no projeto. Esta continuidade também pode ser possível através da publicação e distribuição do aplicativo em site de desenvolvimento mútuo, como é o caso do *SourceForge.net*[7].

O licenciamento do software deve permitir isto também, para tal foi criada uma licença baseada na licença do *Mozilla Public Licence*, a *Migrate Databases Public Licence*, disponível "Apêndice A".

A MPL original obriga que produtor disponibilize o código-fonte original do produto, além de permitir a inclusão de código de *softwares* que contenham outro tipo de licença, entretanto o código sob esta licença deve continuar como ela. Assim futuros trabalhos podem incorporar outros tipos de licença que atendam o desejo de outros programadores. É também possível utilizar o *software* para fins comerciais, não invalidando a idéia do projeto, de justamente poder ajudar em situações de *softwares* comerciais. A definição da MPL original está disponível em http://www.mozilla.org/MPL/MPL-1.1.html.

[7] *SourceForge.net* é um *site* direcionado para desenvolvedores de *softwares* de código aberto para controlar e manter o desenvolvimento destes, funcionando como um repositório de código-fonte. Informações disponíveis em http://sourceforge.net/.

“I don't want to be second best

I don't want to stand in line

I don't want to fall behind

I don't want to get caught out

I don't want to do without

And the lesson I must learn

Is that I've got to wait my turn.”

KT Tunstall

4 IMPLEMENTAÇÃO DA FERRAMENTA

Este capítulo irá tratar da implementação da ferramenta, que ocorreu praticamente como descrito no capítulo “Proposta de desenvolvimento”, sendo apenas necessárias algumas alterações para adaptá-lo a situações inusitadas de conversão. Estas alterações ocorreram durante o desenvolvimento da ferramenta, fase que também foram definidos todos os *templates* necessários e a forma de armazenamento destes arquivos de configuração. Também será mencionado pelo capítulo a publicação da ferramenta no *site SourceForge.net*. Sendo, por fim, mencionado as alternativas em situações de conversões não previstas até o momento.

4.1 Alterações na proposta inicial de desenvolvimento

Inicialmente houve adição de novos parâmetros para configurações de destino além dos mencionados na parte “**Configurações do aplicativo**” do capítulo “**Proposta de desenvolvimento**” (página 39). Estes novos parâmetros foram incluídos por necessidade do *Microsoft SQL Server* e são eles:

- **Substituir palavras**: Esta configuração irá substituir expressões determinadas por outras, em toda instrução SQL que serão geradas. Por exemplo, o *Microsoft* SQL *Server* 2005 não possui função TRIM, sendo necessária esta funcionalidade ser criada pelo usuário. Na hora de sua utilização, como esta é uma função de usuário, é necessário especificar o *schema* desta função. Desta maneira é necessário especificar “DBO.TRIM(expressão)” ao invés de “TRIM(expressão)” [Dewson, 2006; Machanic, 2007];
- **Palavra-chave para coluna auto-incremento**: O *Microsoft* SQL *Server* 2005 possibilita apenas a criação de colunas auto-incremento através da especificação desta na criação da tabela, e o conversor previa apenas a criação de colunas auto-incremento após a criação da tabela. Assim sendo, esta opção

será utilizada para identificar a palavra-chave na definição da coluna auto-incremento, que no caso do *Microsoft* SQL *Server* 2005 é a palavra "IDENTIFY". Esta palavra será informada no lugar do parâmetro "<*default*>" nos *template* "*sql_create_table.conf*" do "Apêndice D" (página 167) [Dewson, 2006; Machanic, 2007];

- ***Templates* de usuário**: Os *templates* de usuários serão responsáveis por permitir a criação de conversões com critérios do próprio usuário. Será criado um *template* de leitura para o banco de dados de origem, especificando uma consulta e através deste serão lidas as informações desta consulta e gerados para o destino. Os parâmetros para os *templates* de origem serão sempre o nome das tabelas, possibilitando assim uma conversão individual por tabela, e para os parâmetros do *template* de destino, serão as próprias colunas da consulta retornada pelo *template* de origem. Mais informações na seção 5.1.3 página 92;
- **Executar *Scripts* pré-conversão e pós-conversão de dados**: A partir de necessidades encontradas em uma migração de um banco de dados qualquer para outro no *Microsoft* SQL *Server*, houve a necessidade de executar *scripts* antes e depois da conversão dos dados. Esta necessidade surgiu pelo fato que para inserir valores em uma coluna que tenha sido criada como auto-incremento ("IDENTIFY") é necessário desabilitar a função de auto-incremento, fazer as operações de inserção e habilitar novamente (comandos dos exemplo 46 e exemplo 47 da página 92). Isto poderia ser resolvido simplesmente pelos "*templates* de usuário" caso fosse possível desabilitar todas as tabelas que contivessem colunas auto-incremento, entretanto, é somente possível desabilitar uma por vez. Assim, será configurado dois dos "*templates* de usuário", um para a pré-execução e outro pra a pós-execução. Exemplo deste recurso poderá ser visto na seção 5.1.3 da página 92.

Os arquivos de configurações criados podem ser visto no "Apêndice D", com o nome de *"default.ini"*, um para os bancos de dados de origem e outro para os bancos de dados de destino.

Com estas alterações também houve modificações nos diagramas de caso de uso dos "Requisitos do sistema" (página 42), no diagrama de atividades do "Fluxo de atividades do

sistema” (página 43) e no diagrama de classes do “Modelo conceitual” (página 44). Os novos diagramas podem ser visualizados no “Apêndice B”.

Para os diagramas de seqüência, a última alteração seria a descrição para o caso de uso “*Templates* do usuário”, entretanto esta vem a ser igual ao diagrama da figura 8 (página 48).

Houve também a adição do recurso de salvar as conversões em arquivo texto, isto é, ao invés dos comandos serem executados no banco de dados de destino serão salvos em arquivo. Isso irá possibilitar com que, no caso de uma conversão, o banco de dados de destino não precise estar instalado, ou na situação de não haver a licença para a instalação do banco. Assim podem ser copiados os arquivos SQL gerados para outra máquina e executados. Isto também possibilitará um possível ganho de desempenho nas conversões caso as ferramentas de submissão de consultas dos próprios sistemas gerenciadores forem mais rápidos do que uma conexão ODBC.

Apesar de a ferramenta sofrer alterações não vistas no capítulo anterior (“Proposta de desenvolvimento”), estas deixaram a ferramenta mais flexível, podendo melhor encaixar-se em outras situações. É necessário também tomar conhecimento que outros bancos que ainda não foram migrados, podem não ser atendidos pelas situações previstas. Isto é principal fator do sistema ser além de gratuito, ser código-aberto.

4.2 Desenvolvimento do aplicativo

A ferramenta foi implementada em *Delphi*, com cerca de 3.517 linhas de código, incluindo declarações de métodos, variáveis, corpo de código, comentários, etc. Uma lista dos arquivos fontes pode ser vista no “Apêndice F”. O componente utilizado de acesso das bases foi o ODBC, assim como descrito no capítulo 3 (página 55).

4.3 Armazenamento dos arquivos de configurações e *templates*

Para armazenamento dos arquivos de configuração foi utilizado a estrutura de arquivos INI (*Initialization* ou Inicialização em português). A estrutura foi escolhida pela sua facilidade de uso para manutenção e a fácil integração com a ferramenta *Delphi* utilizada para desenvolver o projeto.

Os arquivos INI são divididos em “seções” e cada seção possui “entradas” com definições de “valores” para estas “entradas”. É possível comentar linhas utilizando o carácter ponto-e-vírgula duas vezes. Um exemplo de um arquivo INI é visto no exemplo 4 [Leão, 2001].

```
;;comentário
[seção1]
entrada1=valor1
entrada2=valor2
[seção2]
entrada1=valor1
entrada2=valor2
```

Exemplo 4: Exemplo de arquivo INI.

Existem dois arquivos de configurações para cada banco de dados utilizado no sistema, sendo um para destino e origem. Os arquivos se chamam “*default.ini*” por indicarem a configuração padrão. Estes arquivos podem ser visto no “Apêndice D” e estão localizados respectivamente em:

- “.*Configuration**Source**banco_de_dados**default.ini*”;
- “.*Configuration**Target**banco_de_dados**default.ini*”.

Também estão armazenados nestes diretórios os *templates* do sistema. Os *templates* não passam de arquivos texto convencionais contendo as informações necessárias para o funcionamento do sistema. Mais informações sobre eles podem ser vistas na seção 5.1 da página 64. Mais detalhes dos arquivos armazenados nestes diretórios são encontrados no “Apêndice C”.

4.4 Distribuição

O sistema foi disponibilizado no endereço http://migratedatabase.sourceforge.net/ do site *SourceForge.net* como mencionado no capítulo 3, onde é possível baixar os fontes ou uma versão compilada do projeto.

O *SourceForge.net* fornece diversos recursos, entre eles pode-se citar o espaço de armazenamento para o projeto, compartilhando o código com outros desenvolvedores através do *software* CVS[8]. Há também disponibilidade do serviço conhecido como *tracker*, onde é possível reportar problemas, novos recursos ou melhorias. Também é possível construir um site próprio para o projeto, utilizando a linguagem PHP em conjunto com o banco de dados *MySQL*. O site possui facilidades para os usuários procurarem projetos, contando com cerca de dois milhões de desenvolvedores cadastrados. Informações disponíveis em www.sourceforge.net.

4.5 Situações não previstas

O projeto do *software* foi idealizado com base nas principais necessidades de conversão de um problema real, o caso da Núcleo Sistemas [Ferraça, 2007]. Entretanto, os atuais *softwares* de gerenciamento de banco de dados existentes provêem outras características que podem ser importantes na hora de realizar uma migração de uma base de

[8] CVS (*Concurrent Version System* ou Sistema de Versões Concorrentes em português) é um *software* que controla versões de código-fonte de projetos, possibilitando o desenvolvimento mutuo de um sistema. Informações disponíveis em http://savannah.nongnu.org/projects/cvs/.

dados para outra, e que ainda não são providas pelo migrador. Para tal, o *software* é gratuito e de código-fonte aberto.

Contudo, se o usuário necessitar na migração de um recurso não previsto é possível utilizar os *templates* de usuário. Estes *templates* possibilitam migrações totalmente personalizadas. Por exemplo, o sistema na sua atual versão, não possibilita a migração de domínios, mas através deste recurso é possível fazer esta conversão. Um exemplo personalizado de migração de domínios pode ser visto na seção 5.1.3.1 da página 94.

5 CONFIGURAÇÃO DA FERRAMENTA

Este capítulo tem o objetivo de explicar a configuração da ferramenta, definido os *templates* de origem e destino necessários para o funcionamento da mesma. Serão descritas também as configurações personalizadas de usuários.

5.1 Templates do sistema

Os *templates* do sistema podem ser configurados para diferentes SGBD e até o término deste trabalho foram configurados os bancos de dados *Firebird* 2.1, *Microsoft* SQL *Server* 2005, *PostgreSQL* 8.3 *e Sybase Adaptive Server Anywhere* 8. Por questões de tempo outros *templates* não foram configurados, entretanto, estes já são suficientes para mostrar a finalidade do trabalho e provar o seu funcionamento.

Os *templates* neste projeto são arquivos de texto convencionais, e estão basicamente divididos em dois tipos. Um deles é o *template* de leitura, isto é, arquivos responsáveis por conter comandos que serão submetidas à consulta, visando o retorno de determinadas informações fundamentais para o motor do sistema. Estas *templates* são comuns para os bancos de origem, pois são utilizados para a obtenção das estruturas necessárias para a criação do novo banco. Porém, existem alguns destes tipos de *templates* que podem ser configurados para os bancos de destino, que possuem a finalidade de procurar possíveis objetos repetidos na criação de novos.

O outro tipo de *template* é o *template* de escrita, existente basicamente para os bancos de destino. Estes são os responsáveis por conter a estrutura de comandos que serão aplicados aos bancos de destino, com o propósito de criação de objetos do banco de dados, tais como tabelas, chaves estrangeiras, chaves únicas, índices, entre outros. Também existem *templates* de escrita que são responsáveis por executar operações nos bancos de dados, como por exemplo, a criação de uma coluna temporária na base em questão.

Para uma configuração de um banco de origem são fundamentais as criações de cerca de 20 *templates* para origem e 16 *templates* para destino. Estes números podem variar por existirem os *templates* definidos pelo usuário. A tabela 8 e a tabela 9 listam os *templates* básicos para o funcionamento do sistema e o tipo de cada um deles, entretanto, uma listagem de todos os *templates* criados até o momento pode ser vistos no “Apêndice D”.

Tabela 8: Listagem dos arquivos de *templates* de origem.

Fonte: Própria.

Localização dos *templates* de origem: .\Configuration\Source\Banco_de_dados\	**Tipo**
default.ini	Arquivo de configuração (não é um *template*, mas faz parte dos arquivos do diretório)
sql_definition_autoincrement.conf	*Template* de leitura
sql_definition_check.conf	*Template* de leitura
sql_definition_foreign_key.conf	*Template* de leitura
sql_definition_foreign_key_2.conf	*Template* de leitura
sql_definition_index.conf	*Template* de leitura
sql_definition_table.conf	*Template* de leitura
sql_definition_unique_key.conf	*Template* de leitura
sql_exists_view.conf	*Template* de leitura
sql_format_column_date.conf	*Template* de leitura
sql_format_column_time.conf	*Template* de leitura
sql_format_column_timestamp.conf	*Template* de leitura
sql_retrieve_data.conf	*Template* de leitura
sql_retrieve_data_2.conf	*Template* de leitura
sql_script_post-conversion.conf	*Template* de escrita
sql_script_pre-conversion.conf	*Template* de escrita
sql_table_columns.conf	*Template* de leitura
sql_table_count.conf	*Template* de leitura
sql_table_list.conf	*Template* de leitura
sql_temp_id_create.conf	*Template* de escrita

sql_temp_id_drop.conf	*Template* de escrita

Tabela 9: Listagem dos arquivos de *templates* de destino.

Fonte: Própria.

Localização dos *templates* de destino: .\Configuration\Target\Banco_de_dados\	**Tipo**
default.ini	Arquivo de configuração (não é um *template*, mas faz parte dos arquivos do diretório)
sql_create_autoincrement.conf	*Template* de escrita
sql_create_foreign_key.conf	*Template* de escrita
sql_create_index.conf	*Template* de escrita
sql_create_table.conf	*Template* de escrita
sql_create_unique_key.conf	*Template* de escrita
sql_exists_index.conf	*Template* de escrita
sql_exists_index_2.conf	*Template* de escrita
sql_exists_table.conf	*Template* de escrita
sql_find_constraints.conf	*Template* de leitura
sql_find_indices.conf	*Template* de leitura
sql_find_sequences.conf	*Template* de leitura
sql_find_triggers.conf	*Template* de leitura
sql_insert_table.conf	*Template* de escrita
sql_script_post-conversion.conf	*Template* de escrita
sql_script_pre-conversion.conf	*Template* de escrita

Todas as colunas dos *templates* que serão vistas nas próximas seções devem necessariamente se chamar como constam, assim como os parâmetros, pois são fundamentais para o funcionamento do sistema. O tamanho de colunas e parâmetros de entrada não são informados, pois estes não importam, o que é importante são os conteúdos retornados pelas colunas e informados para os parâmetros. Quanto ao tipo, fica explícito pelo seu uso se será numérico ou um conjunto de caracteres.

5.1.1 *Templates* de origem

A metodologia de uso dos templates de origem foi dividida conforme os casos de uso da seção 3.2.1 da página 42, e estas serão apresentadas nos próximos tópicos.

Primeiramente é necessário definir o *template* "*sql_table_list.conf*" (exemplo 5), utilizado para a listar por ordem alfabética todas as tabelas e visões através da coluna "*table_name*". Primeiramente deverão ser retornadas todas as tabelas e depois todas as visões. Nenhum parâmetro de entrada é utilizado.

```
SELECT
t.table_name

FROM
INFORMATION_SCHEMA.TABLES t

ORDER BY
(CASE WHEN t.table_type = 'VIEW' THEN 1 ELSE 0 END),
t.table_name;
```

Exemplo 5: *Template* "*sql_table_list.conf*" no SGBD *PostgreSQL*.

5.1.1.1 *Templates* para pré-conversão e pós-conversão

Para o *template* de pré-conversão foi definido o arquivo "*sql_script_pre-conversion.conf*" (exemplo 6), responsável por executar qualquer conjunto de operações que seja necessário antes de executar uma conversão para um banco de dados de origem. E para o *template* de pós-conversão o arquivo "*sql_script_post-conversion.conf*", responsável por executar operações para um banco de dados de origem após a conversão. Estes *templates* não utilizam nenhum parâmetro de entrada.

```
SET TERM ^;
```

```
CREATE OR REPLACE FUNCTION _list(text, text) RETURNS text AS $$
  SELECT CASE
           WHEN $2 IS NULL THEN $1
           WHEN $1 IS NULL THEN $2
           ELSE $1 OPERATOR(pg_catalog.||) ',' OPERATOR(pg_catalog.||) $2
         END
$$ IMMUTABLE LANGUAGE SQL^

CREATE AGGREGATE list (
  BASETYPE = text,
  SFUNC = _list,
  STYPE = text
)^
```

Exemplo 6: *Template* "*sql_script_pre-conversion.conf*" no SGBD *PostgreSQL*.

5.1.1.2 *Templates* para conversão de tabelas e visões

Primeiramente o *template* utilizado para a conversão de tabelas e visões foi o arquivo "*sql_definition_table.conf*" (exemplo 7), responsável por retornar todas as informações de uma tabela do banco de dados de origem.

```
SELECT DISTINCT
SYSCOLUMN.column_id, --This column is here to be used at the order by clause
CASE WHEN (SYSTABLE.view_def IS NOT NULL) AND (SYSTABLE.table_type = 'VIEW') THEN
   SYSTABLE.view_def || ';'
ELSE
   NULL
END AS view_def,
SYSCOLUMN.pkey,
SYSCOLUMN.nulls,
SYSCOLUMN.width,
SYSCOLUMN.scale,
SYSCOLUMN.column_name,
CAST(TRIM(SYSCOLUMN."default") AS VARCHAR(254)) AS default_column,
TRIM(SYSCOLUMN."check") AS check_column,
CASE TRIM(SYSDOMAIN.domain_name)
  WHEN 'double' THEN 'double precision'
  WHEN 'float' THEN 'real'
  WHEN 'long varchar' THEN 'text'
  ELSE TRIM(SYSDOMAIN.domain_name)
END AS data_type,
(CASE WHEN SYSCOLUMN."default" = 'autoincrement' THEN 'Y' ELSE 'N' END) AS is_autoincrement,
(CASE WHEN SYSTABLE.table_type = 'VIEW' THEN 'VIEW' ELSE 'TABLE' END) AS table_type

FROM
SYSCOLUMN

JOIN SYSTABLE ON
SYSTABLE.table_id = SYSCOLUMN.table_id

JOIN SYSDOMAIN ON
SYSDOMAIN.domain_id = SYSCOLUMN.domain_id

WHERE
SYSTABLE.table_name = '<table_name>'
```

```
ORDER BY
SYSCOLUMN.column_id;
```

Exemplo 7: *Template* "*sql_definition_table.conf*" no SGBD *Sybase Adaptive Server Anywhere*.

Este *template* utiliza o parâmetro de entrada "<*table_name*>", útil para colocar na cláusula "WHERE" da consulta e será substituído pelo nome tabela ao ser executado. A consulta deve ser ordenada pela ordem das colunas (posição) na tabela e deverá retornar as seguintes campos, trazendo ou não resultados:

- **"view_def"**: Se o resultado da consulta for uma visão, a definição desta deve ser retornada neste campo;
- **"pkey"**: Se a coluna faz parte da chave primária o valor de retorno deverá ser "Y", caso contrário "N";
- **"nulls"**: Se a coluna possibilita o estado nulo, os valores de retorno deverão ser "Y" para sim ou "N" para não;
- **"width"**: Referente ao tamanho das colunas, quando aplicáveis a esta. Por exemplo, será utilizado para colunas do tipo VARCHAR, CHAR, NUMERIC, NUMBER, entre outras possíveis. Para colunas NUMERIC e NUMBER, estes valores são referentes à precisão e para tipos, como por exemplo, o INTEGER, este valor para esta coluna deve ser zero.
- **"scale"**: O tamanho da escala para campos como, por exemplo, NUMERIC e NUMBER;
- **"column_name"**: O nome da coluna;
- **"default_column"**: O valor padrão para a coluna;
- **"check_column"**: As *check constraints* para a coluna. Alguns bancos podem armazenar as instruções individualmente por campo referido, como é o caso do *Sybase* ASA, outros o armazenamento é por tabela (visto no *template* "*sql_definition_check.conf*");
- **"data_type"**: Tipo de dados da coluna;
- **"is_autoincrement"**: Para caso o campo tenha valor padrão auto-incremento;

- **“table_type”**: Para caso o resultado da consulta tenha vindo seja uma tabela ou uma visão.

Para exemplificar, será utilizada a criação de uma tabela do exemplo 8, o retorno das informações por este *template* pode ser vistos na tabela 10.

```
CREATE TABLE TEST01 (
  A INTEGER NOT NULL PRIMARY KEY,
  B INTEGER DEFAULT 1 NOT NULL CHECK (B > 0),
  C VARCHAR(50) DEFAULT 'MAYCON',
  D NUMERIC(5,2),
  E TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  F TIME DEFAULT CURRENT_TIME,
  G DATE DEFAULT CURRENT_DATE,
  H REAL,
    CONSTRAINT CKC_CHECK_D CHECK (D > 0));
```

Exemplo 8: Criação de uma tabela “TEST01” no *Firebird*.

Tabela 10: Resultado do *template* “*sql_definition_table.conf*” sobre a tabela “TEST01”.

Fonte: Própria.

VIEW_DEF	PKEY	NULLS	WIDTH	SCALE	COLUMN_NAME	DEFAULT_COLUMN	CHECK_COLUMN	DATA_TYPE	IS_AUTOINCREMENT	TABLE_TYPE
	Y	N	0	0	A			INTEGER	N	**TABLE**
	N	N	0	0	B	1		INTEGER	N	**TABLE**
	N	Y	50	0	C	‘MAYCON’		VARCHAR	N	**TABLE**
	N	Y	5	2	D			NUMERIC	N	**TABLE**
	N	Y	0	0	E	CURRENT_TIMESTAMP		TIMESTAMP	N	**TABLE**
	N	Y	0	0	F	CURRENT_TIME		TIME	N	**TABLE**
	N	Y	0	0	G	CURRENT_DATE		DATE	N	**TABLE**

	N	Y	0	0	H			REAL	N	**TABLE**

Para o resultado do *template* aplicado sobre uma visão (exemplo 9), o retorno é igual da na tabela 10, exceto que a primeira coluna (“view_def”) terá comando para a criação da visão e a última coluna (“table_type”) estará com o valor de “VIEW”.

```
CREATE VIEW TEST02 AS SELECT * FROM TEST01;
```

Exemplo 9: Criação de uma visão teste no *Firebird*.

Na conversão de tabelas também é utilizado o *template* “*sql_definition_check.conf*” (exemplo 10), o qual é responsável por retornar as informações referentes as *check constraints* que não foram retornadas no *template* “*sql_definition_table.conf*” pela coluna “*check_column*”.

```
SELECT
CHK_K.RDB$CONSTRAINT_NAME AS check_name,
TRG.RDB$TRIGGER_SOURCE AS check_rule

FROM
RDB$TRIGGERS TRG

JOIN RDB$CHECK_CONSTRAINTS CHK_K ON
TRG.RDB$TRIGGER_NAME = CHK_K.RDB$TRIGGER_NAME

WHERE
TRG.RDB$RELATION_NAME = '<table_name>' AND
TRG.RDB$TRIGGER_TYPE = 1 AND
EXISTS (SELECT RDB$CONSTRAINT_NAME FROM RDB$RELATION_CONSTRAINTS REL_K
        WHERE CHK_K.RDB$CONSTRAINT_NAME = REL_K.RDB$CONSTRAINT_NAME);
```

Exemplo 10: *Template* “*sql_definition_check.conf*” no SGBD *Firebird*.

Esta consulta faz uso do parâmetro de entrada “*<table_name>*” deverá ser retornado as colunas “*check_name*” para o nome da *constraint* e a coluna “*check_rule*”: para a definição da *constraint*. Nenhuma ordenação padrão é necessária para esta consulta. A tabela 11 traz o retorno do *template* para a criação do objeto do exemplo 8.

Tabela 11: Resultado do *template* “*sql_definition_check.conf*” sobre a tabela “TEST01”.

Fonte: Própria.

CHECK_NAME	CHECK_RULE
INTEG_3083	CHECK (B > 0)
CKC_CHECK_D	CHECK (D > 0)

Por último o *template* “*sql_exists_view.conf*” (exemplo 11) é responsável por retornar se uma visão existe ou não no sistema e o seu parâmetro de entrada é “<*view_name*>”. O arquivo também é utilizado internamente pelo sistema em determinadas rotinas para verificar se uma tabela é uma visão ou não, sendo que na lista de tabelas as visões são incluídas também.

```
SELECT
COUNT(*) AS view_exists

FROM
RDB$RELATIONS

WHERE
RDB$RELATION_NAME = '<view_name>' AND
RDB$VIEW_SOURCE IS NOT NULL;
```

Exemplo 11: *Template* “*sql_exists_view.conf*” no SGBD *Firebird*.

5.1.1.3 *Templates* para conversão dos dados

Primeiramente é utilizado o *template* “*sql_table_count.conf*” (exemplo 12), responsável por retornar a quantidade total de registros de uma tabela com a coluna “*total*”. Esta consulta receberá como parâmetro de entrada o nome da tabela, substituída por “<*table_name*>”. Este *template* é útil em uma migração para dados estatísticos dos números de registros convertidos sobre a quantidade total. Também possui a finalidade de verificar a necessidade do uso dos *templates* responsáveis pelas paginações dos registros, o “*sql_temp_id_create.conf*” (exemplo 13) e o “*sql_temp_id_drop.conf*” (exemplo 14).

```
SELECT COUNT(*) AS total FROM <table_name>;
```

Exemplo 12: *Template* “*sql_table_count.conf*” no SGBD *Firebird*.

```
--The column must be called id_converter, but the index can have any name
ALTER TABLE <table_name> ADD temp_id BIGINT;
CREATE INDEX index_temp_id ON <table_name> (temp_id);
COMMIT;

SET TERM ^;
EXECUTE BLOCK AS
  DECLARE i BIGINT;
  DECLARE aux BIGINT;
BEGIN
  i = 0;
  FOR SELECT temp_id FROM <table_name> INTO :aux AS CURSOR loop DO
  BEGIN
    i = i + 1;
    UPDATE <table_name> SET temp_id = :i WHERE CURRENT OF loop;
  END
END^
SET TERM ;^
```

Exemplo 13: *Template* "*sql_temp_id_create.conf*" no SGBD *Firebird*.

```
--The column must be called id_converter, but the index can have any name
DROP INDEX index_temp_id;
ALTER TABLE <table_name> DROP temp_id;
```

Exemplo 14: *Template* "*sql_temp_id_drop.conf*" no SGBD *Firebird*.

O "*sql_temp_id_create.conf*" é responsável por criar a coluna "*temp_id*" na tabela que será migrada, caso o número de registros passar da quantidade estabelecida na hora da migração (página 100 da seção 6.1). Primeiramente o *script* precisa preencher a coluna com um valor auto-incremento, e após, criar um índice sobre o campo, visando o aumento de performance ao acesso por esta, pois esta coluna será utilizada na cláusula "WHERE" da consulta do *template* "*sql_retrieve_data_2.conf*" (exemplo 16) em conjunto com o *template* "*sql_retrieve_data.conf*" (exemplo 15).

```
SELECT <columns_list> FROM <table_name>
```

Exemplo 15: *Template* "*sql_retrieve_data.conf*" no SGBD *Firebird*.

```
WHERE temp_id BETWEEN <start> AND <end>
```

Exemplo 16: *Template* "*sql_retrieve_data_2.conf*" no SGBD *Firebird*.

O arquivo “*sql_temp_id_create.conf*” é extremamente útil para converter os dados por partes, evitando assim que tabelas enormes consumam grandes quantidades de memória quando abertas. A finalidade do *template* “*sql_temp_id_drop.conf*” é apagar a coluna e o índice temporários criados.

O *template* “*sql_retrieve_data.conf*” deverá montar uma consulta SQL para obter os dados a serem convertidos, tendo como parâmetro de entrada a lista de colunas através de “*<columns_list>*” e o nome da tabela através de “*<table_name>*”. A lista de colunas será obtida através do *template* “*sql_table_columns.conf*”. O *template* “*sql_retrieve_data_2.conf*” é o qual é utilizado em conjunto com o “*sql_temp_id_create.conf*”, substituindo a faixa de de registros à serem obtidos pelos parâmetros de entrada “*<start>*” e “*<end>*”.

O arquivo “*sql_table_columns.conf*” (exemplo 17) deverá retornar a lista de colunas de uma tabela, apenas informando o nome e o tipo da coluna, informações identicas às da tabela 11 através das colunas “*column_name*” e “*datatype_name*”. Este *template* receberá o como parâmetro de entrada o nome da tabela, substituída por “*<table_name>*”. A ordem dos elementos desta consulta deve ser pela posição dos campos na tabela.

```
SELECT DISTINCT
SYSCOLUMN.column_id, --This column is here to be used at the order by clause
SYSCOLUMN.column_name AS column_name,
CASE TRIM(SYSDOMAIN.domain_name)
  WHEN 'double' THEN 'double precision'
  WHEN 'float' THEN 'real'
  WHEN 'long varchar' THEN 'text'
  ELSE TRIM(SYSDOMAIN.domain_name)
END AS data_type

FROM
SYSCOLUMN

JOIN SYSTABLE ON
SYSTABLE.table_id = SYSCOLUMN.table_id

JOIN SYSDOMAIN ON
SYSDOMAIN.domain_id = SYSCOLUMN.domain_id

WHERE
SYSTABLE.table_name = '<table_name>'

ORDER BY
SYSCOLUMN.column_id;
```

Exemplo 17: *Template* “*sql_table_columns.conf*” no SGBD *Sybase Adaptive Server Anywhere*.

Quando um tipo de dados *date*, *time* ou *timestamp* é encontrado no retorno da consulta do *template* “*sql_table_columns.conf*”, estes tipos de dados são formatados pelos *templates*, “*sql_format_column_date.conf*” (exemplo 18), “*sql_format_column_time.conf*” (exemplo 19)

ou “*sql_format_column_timestamp.conf*” (exemplo 20). Isto visa transformá-los um uma *string* com o formato definido pelo padrão ANSI para serem reconhecidos na instrução de inserção no banco de destino. É definido pelo padrão a formatação para o tipo *date* como “*yyyy-mm-dd*”, o padrão para *time* como “*hh:mm:ss.zzz*” e para o tipo *timestamp* como “*yyyy-mm-dd hh:mm:ss.zzz*”, aonde “*yyyy*” é referente ao ano “*mm*” ao mês, “*dd*” ao dia, “*hh*” as horas, “*mm*” aos minutos, “*ss*” aos segundos e “*zzz*” ao milissegundos. [ANSI-SQL 92; ANSI-SQL 99].

```
DATEFORMAT(<column_name>, 'yyyy-mm-dd') AS <column_name>
```

Exemplo 18: *Template* “*sql_format_column_date.conf*” no SGBD *Sybase Adaptive Server Anywhere*.

```
DATEFORMAT(<column_name>, 'hh:mm:ss.sss') AS <column_name>
```

Exemplo 19: *Template* “*sql_format_column_time.conf*” no SGBD *Sybase Adaptive Server Anywhere*.

```
DATEFORMAT(<column_name>, 'yyyy-mm-dd hh:mm:ss.sss') AS <column_name>
```

Exemplo 20: *Template* “*sql_format_column_timestamp.conf*” no SGBD *Sybase Adaptive Server Anywhere*.

5.1.1.4 *Template* para conversão de colunas auto-incremento

Para alguns bancos a definição de colunas auto-incremento é feito por gatilhos como é o caso do *Firebird* [Borrie, 2006], o *Microsoft* SQL *Server* pela cláusula “IDENTIFY” na definição da tabela [Dewson, 2006], o *PostgreSQL* [Matthew *et al*., 2005] e *Sybase* ASA [*Sybase*, 2001] pela cláusula “DEFAULT”. De qualquer forma, o *template* “*sql_definition_autoincrement.conf*” (exemplo 21) deverá retornar através do campo “*column_name*” todas as colunas de uma tabela que são auto-incremento, tendo como parâmetro de entrada o parâmetro “<*table_name*>”. A ordem recomendada para esta consulta é a ordem da coluna na tabela, caso existam mais de uma coluna auto-incremento por objeto.

```
SELECT
column_name

FROM
information_schema.columns

WHERE
table_name = '<table_name>' AND
SUBSTR(column_default, 1, 7) = 'nextval'

ORDER BY
column_name;
```

Exemplo 21: *Template* "*sql_definition_autoincrement.conf*" no SGBD *PostgreSQL*.

5.1.1.5 *Template* para conversão de chaves únicas

O *template* definido pelo arquivo "*sql_definition_unique_key.conf*" (exemplo 22) é responsável por retornar as informações referentes a chaves únicas no banco de dados de origem.

```
SELECT
REL_CON.RDB$CONSTRAINT_NAME AS constraint_name,
TRIM((SELECT
      TRIM(LIST(' ' || TRIM(ISG.RDB$FIELD_NAME)))
      FROM (SELECT * FROM RDB$INDEX_SEGMENTS I ORDER BY I.RDB$FIELD_POSITION) ISG
      WHERE ISG.RDB$INDEX_NAME = IX.RDB$INDEX_NAME)) AS constraint_columns,
IIF(IX.RDB$INDEX_TYPE = 1, 'DESC', 'ASC') AS index_order

FROM
RDB$RELATION_CONSTRAINTS REL_CON

JOIN RDB$INDICES IX ON
REL_CON.RDB$INDEX_NAME = IX.RDB$INDEX_NAME

WHERE
REL_CON.RDB$CONSTRAINT_TYPE = 'UNIQUE' AND
REL_CON.RDB$RELATION_NAME = '<table_name>'

ORDER BY
REL_CON.RDB$CONSTRAINT_NAME;
```

Exemplo 22: *Template* "*sql_definition_unique_key.conf*" no SGBD *Firebird*.

Esta consulta deve retornar as informações:

- ***"constraint_name"***: O nome da *constraint*;

- ***"constraint_columns"***: A listagem das colunas envolvidas, separadas por vírgula;
- ***"index_order"***: A ordenação do índice referente à chave única, sendo os valores possíveis "ASC" para ascendente e "DESC" para descendente.

Esta consulta é executada para uma tabela específica utilizando o parâmetro de entrada "<*table_name*>". É possível verificar um exemplo do resultado desta consulta visualizando a tabela 12 sobre a criação do objeto do exemplo 23

```
ALTER TABLE TEST01
  ADD CONSTRAINT UK_TEST01_E UNIQUE (E)
  USING ASC INDEX XU_TEST01_E;
```

Exemplo 23: Criação de uma chave única para teste no *Firebird*.

Tabela 12: Resultado do *template* "*sql_definition_unique_key.conf*" sobre a chave única teste.

Fonte: Própria.

CONSTRAINT_NAME	CONSTRAINT_COLUMNS	INDEX_ORDER
UK_TEST01_E	E	ASC

A ordem recomendada para esta consulta é nome da *constraint*.

5.1.1.6 *Template* para conversão de chaves estrangeiras

Existem duas maneiras de retornar as informações necessárias, a primeira delas é definida apenas pelo *template* do arquivo "*sql_definition_foreign_key.conf*" (modelo 1) (exemplo 24) responsável por fornecer todas as informações referentes a chaves estrangeiras no banco de dados de origem.

```
SELECT
RC2.RDB$RELATION_NAME AS table_name_dest,
RC.RDB$CONSTRAINT_NAME AS role,
```

```
TRIM(IIF(REFC.RDB$UPDATE_RULE = 'RESTRICT', '', 'ON UPDATE ' || TRIM(REFC.RDB$UPDATE_RULE)) ||
     IIF(REFC.RDB$DELETE_RULE = 'RESTRICT', '', 'ON DELETE ' || TRIM(REFC.RDB$DELETE_RULE)))
AS actions,
TRIM((SELECT
      TRIM(LIST(' ' || TRIM(XS2.RDB$FIELD_NAME)))
      FROM (SELECT * FROM RDB$INDEX_SEGMENTS XS2 ORDER BY XS2.RDB$FIELD_POSITION) XS2
      WHERE XS2.RDB$INDEX_NAME = RC.RDB$INDEX_NAME)) AS columns_origin,
TRIM((SELECT
      TRIM(LIST(' ' || TRIM(XS.RDB$FIELD_NAME)))
      FROM (SELECT * FROM RDB$INDEX_SEGMENTS XS ORDER BY XS.RDB$FIELD_POSITION) XS
      WHERE XS.RDB$INDEX_NAME = RC2.RDB$INDEX_NAME)) AS columns_destination

FROM
RDB$RELATION_CONSTRAINTS RC

JOIN RDB$REF_CONSTRAINTS REFC ON
REFC.RDB$CONSTRAINT_NAME = RC.RDB$CONSTRAINT_NAME

JOIN RDB$RELATION_CONSTRAINTS RC2 ON
REFC.RDB$CONST_NAME_UQ = RC2.RDB$CONSTRAINT_NAME

WHERE
RC.RDB$CONSTRAINT_TYPE = 'FOREIGN KEY' AND
RC.RDB$RELATION_NAME = '<table_name>'

ORDER BY
RC.RDB$CONSTRAINT_NAME;
```

Exemplo 24: *Template* "*sql_definition_foreign_key.conf*" (modelo 1) no SGBD *Firebird*.

Os campos desta consulta devem ser:

- ***"table_name_dest"***: Nome da tabela de destino;
- ***"role"***: Nome da *constraint*;
- ***"actions"***: As ações de atualização e deleção, como definida no padrão ANSI [ANSI-SQL 92; ANSI-SQL 99], seguindo a seguinte sintaxe do exemplo 25;
- ***"columns_origin"***: A listagem de colunas de origem separadas por vírgula. Estas colunas devem estar ordenadas de acordo com a relação às colunas de destino (coluna "*columns_destination*"), por exemplo, as colunas de uma tabela de origem "tabela_a_coluna_a, tabela_a_coluna_b", devem estar ordenadas na lista de colunas de destino, como "tabela_b_coluna_a, tabela_b_coluna_b" e não como "tabela_b_coluna_b, tabela_b_coluna_a";
- ***"columns_destination"***: A listagem de colunas de destino separadas por vírgula.

Esta consulta é executada para uma tabela específica utilizando o parâmetro de entrada "<*table_name*>". É possível verificar um exemplo do resultado desta consulta visualizando a tabela 13 sobre a criação do objeto do exemplo 26.

```
[ON DELETE {NO ACTION|CASCADE|SET DEFAULT|SET NULL}]
[ON UPDATE {NO ACTION|CASCADE|SET DEFAULT|SET NULL}]
```

Exemplo 25: Definição das ações de uma chave estrangeira.

```
ALTER TABLE TEST01
  ADD CONSTRAINT FK_TEST01_TEST01 FOREIGN KEY (B, B2)
    REFERENCES TEST01 (A, A2) ON DELETE SET NULL ON UPDATE CASCADE
  USING INDEX X_TESTE01_TESTE01;
```

Exemplo 26: Criação de uma chave estrangeira para teste no *Firebird*.

Tabela 13: Resultado do *template* "*sql_definition_foreign_key.conf*" (modelo 1) sobre a chave estrangeira teste.

Fonte: Própria.

TABLE_NAME_ DEST	ROLE	ACTIONS	COLUMNS_ ORIGIN	COLUMNS_ DESTINATION
TEST01	FK_TEST01_ TEST01	ON UPDATE CASCADE ON DELETE SET NULL	B, B2	A, A2

A ordem padrão para este *template* (modelo 1) é nome da *constraint*.

A segunda maneira de se obter as informações, modelo 2 (exemplo 27), foi especialmente desenvolvida por necessidades da versão usada do SGBD *Sybase* ASA, onde por motivos técnicos foi necessário dividir o arquivo em dois, primeiramente o arquivo "*sql_definition_foreign_key.conf*" trazendo as colunas:

- ***"foreign_table_id"***: Código identificador da tabela de destino, ou o próprio nome desta, que será utilizado como parâmetro de entrada para o *template* "*sql_definition_foreign_key_2.conf*";
- ***"foreign_key_id"***: Código identificador da chave estrangeira, ou o próprio nome desta, que será utilizado como parâmetro de entrada para o *template* "*sql_definition_foreign_key_2.conf*";
- ***"primary_table_id"***: Código identificador da tabela de origem, ou o próprio nome desta, que será utilizado como parâmetro de entrada para o *template* "*sql_definition_foreign_key_2.conf*";

- ***"table_name_dest"***: Nome da tabela de destino;
- ***"role"***: Nome da *constraint*;
- ***"actions"***: As ações de atualização e deleção, como definida no padrão ANSI [ANSI-SQL 92; ANSI-SQL 99], seguindo a seguinte sintaxe do exemplo 25;

```
SELECT
fk.foreign_table_id,
fk.foreign_key_id,
fk.primary_table_id,
pk_tab.table_name as table_name_dest,
fk.role,
TRIM(((SELECT
      (CASE t.referential_action
         WHEN 'C' THEN 'ON UPDATE CASCADE'
         WHEN 'D' THEN 'ON UPDATE SET DEFAULT'
         WHEN 'N' THEN 'ON UPDATE SET NULL'
         WHEN 'R' THEN '' --ON UPDATE NO ACTION
       END)
       FROM systrigger t
       WHERE
       t.foreign_key_id = fk.foreign_key_id AND
       t.foreign_table_id = fk_tab.table_id AND
       t.table_id = pk_tab.table_id AND
       t."event" IN ('C', 'U')) ||
     (SELECT
      (CASE t.referential_action
         WHEN 'C' THEN 'ON DELETE CASCADE'
         WHEN 'D' THEN 'ON DELETE SET DEFAULT'
         WHEN 'N' THEN 'ON DELETE SET NULL'
         WHEN 'R' THEN '' --ON DELETE NO ACTION
       END)
       FROM systrigger t
       WHERE
       t.foreign_key_id = fk.foreign_key_id AND
       t.foreign_table_id = fk_tab.table_id AND
       t.table_id = pk_tab.table_id AND
       t."event" = 'D'))) AS actions

FROM
SYS.SYSFOREIGNKEY AS fk

JOIN SYS.SYSTABLE AS fk_tab ON
fk_tab.table_id = fk.foreign_table_id

JOIN SYS.SYSUSERPERM AS fk_up ON
fk_up.user_id = fk_tab.creator

JOIN SYS.SYSTABLE AS pk_tab ON
pk_tab.table_id = fk.primary_table_id

JOIN SYS.SYSUSERPERM AS pk_up ON
pk_up.user_id = pk_tab.creator

WHERE
fk_tab.table_name = '<table_name>'

ORDER BY
fk.role;
```

Exemplo 27: *Template* "*sql_definition_foreign_key.conf*" (modelo 2) no SGBD *Sybase Adaptive Server Anywhere*.

Esta consulta é executada para uma tabela específica utilizando o parâmetro de entrada "<*table_name*>". A ordem recomendada para este *template* é o nome da *constraint*.

O outro arquivo necessário é o "*sql_definition_foreign_key_2.conf*" (exemplo 28), que deve trazer apenas as informações das colunas utilizadas na chave estrangeira, em uma linha para cada coluna. As informações necessárias de retorno são:

- ***"column_name_origin"***: Todas as colunas que fazem parte da chave estrangeira de origem;
- ***"column_name_destination"***: Todas as colunas que fazem parte da chave estrangeira de destino.

```
SELECT DISTINCT
fkc.primary_column_id, --This column is here to be used at the order by clause
TRIM(fk_col.column_name) AS column_name_origin,
TRIM(pk_col.column_name) AS column_name_destination

FROM
SYS.SYSFKCOL AS fkc

JOIN SYS.SYSCOLUMN AS fk_col ON
fk_col.table_id = fkc.foreign_table_id AND
fk_col.column_id = fkc.foreign_column_id

JOIN SYS.SYSCOLUMN AS pk_col ON
pk_col.column_id = fkc.primary_column_id

WHERE
fkc.foreign_table_id = <foreign_table_id> AND
fkc.foreign_key_id = <foreign_key_id> AND
pk_col.table_id = <primary_table_id>

ORDER BY
fkc.primary_column_id;
```

Exemplo 28: *Template* "*sql_definition_foreign_key_2.conf*" (modelo 2) no SGBD *Sybase Adaptive Server Anywhere*.

Os parâmetros de entrada para este *template* são "<*foreign_table_id*>", "<*foreign_key_id*>" e "<*primary_table_id*>" obtidos da consulta do arquivo "*sql_definition_foreign_key.conf*". É possível ver um exemplo do resultado dos dois *templates* do modelo 2 na tabela 15 e tabela 16. A ordem para este *template* deve ser a ordem da chave primaria na *constraint*.

Tabela 14: Resultado do *template* "*sql_definition_foreign_key.conf*" (modelo 2) sobre a chave estrangeira teste.

Fonte: Própria.

TABLE_NAME_ DEST	ROLE	ACTIONS	COLUMNS_ ORIGIN	COLUMNS_ DESTINATION
TEST01	FK_TEST01_ TEST01	ON UPDATE CASCADE ON DELETE SET NULL	B, B2	A, A2

Tabela 15: Resultado do *template* "*sql_definition_foreign_key_2.conf*" sobre a chave estrangeira teste.

Fonte: Própria.

COLUMNS_ ORIGIN	COLUMNS_ DESTINATION
B	A
B2	A2

O motor do sistema saberá quais tipos modelos de *templates* pegar pela existência do "*sql_definition_foreign_key_2.conf*" na pasta de configuração de origem.

O problema técnico que ocorreu com a geração do modelo 1 para o *Sybase* foi decorrente pela não possibilidade de ordenar sub-consultas em comando "SELECT". Este recurso foi extremamente necessária para possibilitar que a listagem das colunas de origem e destino da chave estrangeira viessem na mesma ordem do seu relacionamento.

5.1.1.7 *Template* para conversão de índices

O *template* definido pelo arquivo "*sql_definition_index.conf*" (exemplo 29) é responsável por retornar as informações referentes aos índices de uma tabela no banco de dados de origem.

```
SELECT
IX.RDB$INDEX_NAME AS index_name,
```

```
IIF(IX.RDB$UNIQUE_FLAG = 1, 'U', 'N') AS index_unique,
IIF(IX.RDB$INDEX_TYPE = 1, 'DESC', 'ASC') AS index_order,
TRIM((SELECT
      TRIM(LIST(' ' || TRIM(ISG.RDB$FIELD_NAME)))
      FROM (SELECT * FROM RDB$INDEX_SEGMENTS I ORDER BY I.RDB$FIELD_POSITION) ISG
      WHERE ISG.RDB$INDEX_NAME = IX.RDB$INDEX_NAME)) AS index_columns,

TRIM((SELECT
      TRIM(LIST(' ' || TRIM(ISG.RDB$FIELD_NAME) || ' ' || IIF(IX.RDB$INDEX_TYPE = 1, 'DESC',
'ASC')))
      FROM (SELECT * FROM RDB$INDEX_SEGMENTS I ORDER BY I.RDB$FIELD_POSITION) ISG
      WHERE ISG.RDB$INDEX_NAME = IX.RDB$INDEX_NAME)) AS index_columns_order

FROM
RDB$INDICES IX

JOIN RDB$RELATIONS R ON
IX.RDB$RELATION_NAME = R.RDB$RELATION_NAME

WHERE
R.RDB$RELATION_NAME = '<table_name>' AND
NOT EXISTS (SELECT * FROM RDB$RELATION_CONSTRAINTS REL_CON
            WHERE REL_CON.RDB$INDEX_NAME = IX.RDB$INDEX_NAME)

ORDER BY
IIF(IX.RDB$UNIQUE_FLAG = 1, 0, 1),
IX.RDB$INDEX_NAME;
```

Exemplo 29: *Template* "*sql_definition_index.conf*" no SGBD *Firebird*.

Esta consulta deve retornar como campos:

- ***"index_name"***: Nome do índice;
- ***"index_unique"***: Indicativo se o índice é do tipo *unique* ou não, sendo os valores possíveis para esta coluna "U" para caso seja única ou "N" para caso não seja;
- ***"index_order"***: A principal ordem do índice, sendo os valores possíveis "ASC" para ascendente ou "DESC" para descendente, para ser utilizado por bancos que não permitem a ordenação individual por colunas;
- ***"index_columns"***: Lista das colunas do índice, separadas por vírgula;
- ***"index_columns_order"***: Lista das colunas do índice incluindo após cada coluna se o campo é ordenado ascendentemente ou descendentemente, através dos valores "ASC" e "DESC" e também separadas por vírgula. Existe a listagem de colunas com ordenação e sem ordenação, pois alguns bancos como o *Firebird* permitem somente a ordem geral por índice [Borrie, 2006] e outros como o ASA, permitem a ordenação do índice individualmente pelas colunas que o compõe [*Sybase*, 2001].

Esta consulta é executada para uma tabela específica utilizando o parâmetro de entrada "<*table_name*>". É possível verificar um exemplo do resultado desta consulta visualizando a tabela 16 sobre a criação do objeto do exemplo 30.

```
CREATE UNIQUE ASC INDEX XU_TEST01_C ON TEST01 (C);
```

Exemplo 30: Criação de um índice teste no *Firebird*.

Tabela 16: Resultado do *template* "*sql_definition_index.conf*" sobre o índice teste.

Fonte: Própria.

INDEX_ NAME	INDEX_ UNIQUE	INDEX_ ORDER	INDEX_ COLUMNS	INDEX_COLUMNS_ ORDER
X_TEST01_C	U	ASC	C	C ASC

A consulta deve trazer primeiramente todos os índices únicos e depois os não-únicos, considerando como segundo critério a ordem alfabética dos nomes dos objetos.

5.1.2 *Templates* de destino

A maior parte dos *templates* de destino define a sintaxe dos comandos para criação dos objetos no banco de dados de destino. Os parâmetros de entrada nestes *templates* irão ser substituídos por nomes de tabelas, índices, entre outros, gerados pelo sistema. Alguns outros *templates* são responsáveis por fazer procuras de objetos para verificar se estes já existem ou não no banco de destino.

5.1.2.1 *Templates* para pré-conversão e pós-conversão

O arquivo “*sql_script_pre-conversion.conf*” (exemplo 30) é responsável por executar qualquer conjunto de operações que seja necessário antes de rodar uma conversão para um banco de dados de destino. E o arquivo “*sql_script_post-conversion.conf*” (exemplo 32), é responsável por executar operações após a conversão para um banco de dados de destino. Estes *templates* não utilizam nenhum parâmetro de entrada.

```
CREATE FUNCTION dbo.TRIM(@string VARCHAR(MAX)) RETURNS VARCHAR(MAX)
BEGIN
  RETURN LTRIM(RTRIM(@string))
END;
```

Exemplo 31: *Template* “*sql_script_pre-conversion.conf*” no SGBD *Microsoft* SQL *Server*.

```
RECREATE VIEW DUAL (FIELD) AS SELECT NULL AS FIELD FROM RDB$DATABASE;
```

Exemplo 32: *Template* “*sql_script_post-conversion.conf*” no SGBD *Firebird*.

5.1.2.2 *Template* para criação de tabelas

O *template* responsável por definir sintaxe padrão para criação de tabela é o arquivo “*sql_create_table.conf*” (exemplo 33) e os seus parâmetros de entrada são:

- **“<*table_name*>”**: Nome da tabela;
- **“<*column_name*>”**: Nome da coluna;
- **“<*data_type*>”**: Tipo do dado;
- **“<*default*>”**: Valor padrão para a coluna, substituída por exemplo, por “*DEFAULT CURRENT_DATE*” para campos data, “*DEFAULT 0*” para campos inteiros e assim por diante;
- **“<*not_null*>”**: Definição se a coluna possibilita ou não estados nulos;
- **“<*check_name*>”**: Nome da *constraint* do tipo *check* para caso a tabela de origem possua;

- **“<*check_rule*>”**: Definição da *constraint* do tipo *check*;
- **“<*primary_key_name*>”**: Nome da chave primária da tabela;
- **“<*primary_key_columns*>”**: Lista separada por vírgula das colunas pertencentes à chave primaria;
- **“<*index_name*>”**: Nome do índice referente à chave primaria para bancos que permitem a definição na sua criação, como o caso do *Firebird* [Borrie, 2006].

```
CREATE TABLE <table_name> (
	<column_name> <data_type> <default> <not_null>,
		CONSTRAINT <check_name> <check_rule>,
		CONSTRAINT <primary_key_name> PRIMARY KEY (<primary_key_columns>) USING INDEX
<index_name>
);
```

Exemplo 33: *Template* “*sql_create_table.conf*” no SGBD *Firebird*.

5.1.2.3 *Template* para inserção dos dados

O arquivo “*sql_insert_table.conf*” (exemplo 34) é o *template* que define o padrão de inserção no banco de dados de destino a partir dos dados lidos do banco de dados de origem. Os parâmetros de entrada são:

“<*table_name*>”: Nome tabela;

“<*columns_list*>”: Lista das colunas separadas por vírgula;

“<*values_list*>”: Lista dos valores separados por vírgula.

```
INSERT INTO <table_name> (<columns_list>) VALUES (<values_list>);
```

Exemplo 34: *Template* “*sql_insert_table.conf*” no SGBD *Firebird*.

5.1.2.4 *Template* para criação de colunas auto-incremento

O arquivo "*sql_create_autoincrement.conf*" define a sintaxe para a criação de colunas auto-incremento. Sejam elas por comandos de alteração de tabela, como é o caso do *Sybase* ASA (exemplo 35) ou por criação de *sequences* em conjunto com a criação de gatilhos, como é o caso do *Firebird* (exemplo da página 164 do "Apêndice D"). Como parâmetros de entrada para este *template* são definidos:

- **"<*sequence_name*>"**: O nome do objeto *sequence* que poderá ser criado em situações que se fizerem necessárias;
- **"<*table_name*>"**: Nome da tabela em questão;
- **"<*column_name*>"**: Nome da coluna que receberá a definição de auto-incremento;
- **"<*trigger_name*>"**: Nome do gatilho que poderá ser criado caso necessário.

```
ALTER TABLE <table_name> ALTER <column_name> SET DEFAULT AUTOINCREMENT;
CREATE VARIABLE @aux BIGINT;
SET @aux = (SELECT COALESCE(MAX(<column_name>) + 1, 1) FROM <table_name>);
CALL sa_reset_identity('<table_name>', 'dba', @aux);
DROP VARIABLE @aux;
```

Exemplo 35: *Template* "*sql_create_autoincrement.conf*" no SGBD *Sybase Adaptive Server Anywhere*.

5.1.2.5 *Template* para criação de chaves únicas

O *template* definido pelo arquivo "*sql_create_unique_key.conf*" (exemplo 36) é responsável por criar as chaves únicas no banco de dados de destino, sendo os parâmetros de entrada:

- **"<*table_name*>"**: Nome da tabela para a chave única;
- **"<*unique_name*>"**: Nome do *constraint*;

- **“<*columns_list*>”**: Lista de colunas separadas por vírgula;
- **“<*index_order*>”**: Ordenação dos índices para banco de dados que possibilitem esta operação, por exemplo, o *Firebird* [Borrie, 2006];
- **“<*index_name*>”**: Nome do índice que será criada para banco de dados que possibilitem esta ação, por exemplo, o *Firebird* [Borrie, 2006].

```
ALTER TABLE <table_name>
  ADD CONSTRAINT <unique_name> UNIQUE (<columns_list>)
  USING <index_order> INDEX <index_name>;
```

Exemplo 36: *Template* “*sql_create_unique_key.conf*” no SGBD *Firebird*.

5.1.2.6 *Template* para criação de chaves estrangeiras

O *template* “*sql_create_foreign_key.conf*” (exemplo 37) define a sintaxe para a criação de chaves estrangeiras e os seus parâmetros de entrada são:

- **“<*table_origin_name*>”**: Nome da tabela que será criada a chave estrangeira, isto é, a tabela de origem;
- **“<*foreign_key_name*>”**: Nome da chave estrangeira;
- **“<*columns_list_origin*>”**: Lista das colunas que farão referência com a tabela de origem, separadas por vírgulas;
- **“<*table_destination_name*>”**: Nome da tabela que fará a referência com a chave estrangeira, isto é, a tabela de destino;
- **“<*columns_list_destination*>”**: Lista das colunas que farão referência com a tabela de destino, separadas por vírgulas;
- **“<*action*>”**: As ações de atualização e deleção, como definida no padrão ANSI [ANSI-SQL 92; ANSI-SQL 99], seguindo a seguinte sintaxe do exemplo 25;

- **“<*index_name*>”**: Alguns gerenciadores de bancos de dados possibilitam a criação ou nomeação dos índices referentes à chave criada, e o sistema fornece um nome conforme os padrões definidos nos arquivos de configurações.

```
ALTER TABLE <table_origin_name>
  ADD CONSTRAINT <foreign_key_name> FOREIGN KEY (<columns_list_origin>)
    REFERENCES <table_destination_name> (<columns_list_destination>) <action>
  USING INDEX <index_name>;
```

Exemplo 37: *Template* “*sql_create_foreign_key.conf*” no SGBD *Firebird*.

5.1.2.7 *Template* para criação de índices

O *template* para criação de índices é definido pelo arquivo “*sql_create_index.conf*” (exemplo 38) e seus parâmetros de entrada são:

“<*index_unique*>”: Definição se o índice é único ou não. Para índices únicos a palavra “UNIQUE” é especificada;

“<*index_order*>”: Ordem geral para todo o índice;

“<*index_name*>”: Nome do índice;

“<*table_name*>”: Nome da tabela que o índice irá ser criado;

“<*columns_list*>”: Lista das colunas separadas por vírgulas;

Para alguns bancos a definição da ordem das colunas é feita individualmente pela coluna, suprindo o parâmetro “*<index_order>*” e ao invés de especificar o parâmetro “*<columns_list>*” é especificado o parâmetro “*<columns_list_order>*”, que conterá a lista das colunas incluindo a ordem individual para cada uma delas. É possível ver exemplo da primeira sintaxe para o banco de dados *Firebird* no exemplo 38 e da segunda sintaxe para os outros bancos configurados no “Apêndice D”.

```
CREATE <index_unique> <index_order> INDEX <index_name> ON <table_name> (<columns_list>);
```

Exemplo 38: *Template* “*sql_create_index.conf*” no SGBD *Firebird*.

5.1.2.8 *Templates* de procura

Os demais *templates* de origem são utilizados para a procura da tabela objetos duplicados. Sendo o arquivo "*sql_exists_index.conf*" (exemplo 39) em conjunto com o arquivo "*sql_exists_index_2.conf*" (exemplo 40) responsáveis por procurar os índices duplicados pela sua definição, isto é, as colunas que compõem um índice e incluindo a sua ordem. Isto possibilita que quando um banco de dados seja migrado para outro, índices duplicados não sejam criados.

```
SELECT
COUNT(DISTINCT col.column_id) AS index_exists

FROM
sys.tables AS tab

JOIN sys.columns AS col ON
tab.object_id = col.object_id

JOIN sys.index_columns AS idx ON
idx.object_id = tab.object_id AND
idx.column_id = col.column_id

WHERE
tab.name = '<table_name>'
```

Exemplo 39: *Template* "*sql_exists_index.conf*" no SGBD *Microsoft* SQL *Server*.

```
OR col.name = '<column_name>'
```

Exemplo 40: *Template* "*sql_exists_index_2.conf*" no SGBD *Microsoft* SQL *Server*.

A duplicação no banco de destino poderia ocorrer por dois fatores, sendo o primeiro por índices criados duas vezes no banco de dados de origem. Ou para sistemas gerenciadores que informem no *template* responsável por retornar as informações dos índices objetos que sejam criados automaticamente por chaves estrangeiras, como é o caso do banco *Firebird* [Borrie, 2006].

O *template* "*sql_exists_index.conf*" é responsável por definir uma consulta SQL no banco de dados de destino para procurar nas tabelas de sistema a existência do índices

duplicados. E o arquivo “*sql_exists_index_2.conf*” responsável por informar um condição lógica para cada coluna do índice. Os parâmetros de entradas são o nome da tabela através de “<*table_name*>” e o nome de cada campo do índice através do parâmetro “<*column_name*>”. Deve ser utilizada como retorno para este parâmetro a coluna “index_exists”, retornado 0 para caso o índice não exista e 1 (ou mais) para caso o índice exista.

O *template* “*sql_exists_table.conf*” (exemplo 41) define uma consulta para verificar a existência de tabelas no banco de dados de destino, sendo necessário a definição da coluna “*table_exists*” retornando 0 para caso a tabela não exista e 1 (ou mais) para caso a tabela exista.

```
SELECT
COUNT(*) AS table_exists

FROM
SYS.SYSTABLE AS tab

WHERE
UPPER(tab.table_name) = UPPER('<table_name>');
```

Exemplo 41: *Template* “*sql_exists_table.conf*” no SGBD *Sybase Adaptive Server Anywhere*.

Os arquivos “*sql_find_constraints.conf*” (exemplo 42), “*sql_find_indices.conf*” (exemplo 43), “*sql_find_sequences.conf*” (exemplo 44) e “*sql_find_triggers.conf*” (exemplo 45) definem *templates* para a procurar *constraints*, índices, *sequences* e gatilhos, sendo necessários que todos definam a coluna “*total*” retornando o número de ocorrências. Estes *templates* podem utilizar os parâmetros de entrada “<*object_name*>”, para o nome do objeto e “<*table_name*>” para o nome da tabela.

```
SELECT
COUNT(*) AS total

FROM
SYS.SYSFOREIGNKEY AS fk

JOIN SYS.SYSTABLE AS fk_tab ON
fk_tab.table_id = fk.foreign_table_id

WHERE
fk_tab.table_name = '<table_name>' AND
fk.role = '<object_name>';
```

Exemplo 42: *Template* “*sql_find_constraints.conf*” no SGBD *Sybase Adaptive Server Anywhere*.

```
SELECT
COUNT(*) AS total

FROM
SYS.SYSTABLE AS tab

KEY JOIN SYS.SYSINDEX AS idx

WHERE
tab.table_name = '<table_name>' AND
idx.index_name = '<object_name>';
```

Exemplo 43: *Template* "*sql_find_indices.conf*" no SGBD *Sybase Adaptive Server Anywhere*.

```
SELECT
COUNT(*) AS total

FROM pg_class c

WHERE
c.relkind = 'S' AND
c.relnamespace IN (SELECT oid
                   FROM pg_namespace
                   WHERE nspname NOT LIKE 'pg_%' AND
                         nspname <> 'information_schema')
                         AND
c.relname = '<object_name>';
```

Exemplo 44: *Template* "*sql_find_sequences.conf*" no SGBD *PostgreSQL*.

```
SELECT COUNT(*) AS total FROM RDB$TRIGGERS WHERE RDB$TRIGGER_NAME = '<object_name>';
/*<table_name>*/
```

Exemplo 45: *Template* "*sql_find_triggers.conf*" no SGBD *Firebird*.

5.1.3 *Templates* de usuário

Os *templates* de usuário permitem migrações totalmente personalizadas e elas não são obrigatórias. Elas auxiliam a migrar objetos não suportados pelo conversor. Para os bancos de dados atualmente homologados, foi necessário o uso de *templates* de usuário para a conversão de destino do *Microsoft* SQL *Server*.

O *Microsoft* SQL *Server* não permite a inserção de valores em uma coluna que tenha sido criada como auto-incremento ("IDENTIFY"), sendo necessário desabilitar a função de auto-incremento para a coluna da tabela, inserir os valores na tabela, e depois habilitar auto-incremento novamente [Dewson, 2006; Machanic, 2007].

Inicialmente foram criados dois arquivos de configuração, um para desabilitar e outro para habilitar, com os nomes de "*sql_user_script_#1.conf*" e "*sql_user_script_#2.conf*", que podem ser vistos no "Apêndice D" ou no exemplo 46 e exemplo 47.

```
SET IDENTITY_INSERT <table_name> ON;
```

Exemplo 46: *Template* "*sql_user_script_#1.conf*" no SGBD *Microsoft* SQL *Server*.

```
SET IDENTITY_INSERT <table_name> OFF;
```

Exemplo 47: *Template* "*sql_user_script_#2.conf*" no SGBD *Microsoft* SQL *Server*..

Para o migrador identificar que estes são os *templates* de usuário a serem utilizados é necessário especificar para o arquivo "*default.ini*" através da entrada "*User Script Target*". A configuração pode ser visualizada no "Apêndice D" ou no exemplo 48.

É necessário configurar a origem das informações que são lidas dos bancos de dados de origem através da entrada "*User Script Source*". Para esta situação o *template* de leitura é informado pelo o arquivo "*sql_definition_autoincrement.conf*", pois as tabelas que precisam sofrer estas operações são somente as que contem definições de colunas auto-incremento. Poderia ser possível configurar um parâmetro totalmente personalizado para os bancos de dados de origem, entretanto, esta configuração deve ser feita para todo banco de origem.

Por fim, é necessário especificar o nome do *script* do usuário através da entrada "*User Script Name*".

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; User Scripts. Configurations to User Scripts.
;;
;; Defaults:
;; Types: String
User Script Name #1=Disable Columns with IDENTIFY Clause
User Script Source #1=sql_definition_autoincrement.conf
User Script Target #1=sql_user_script_#1.conf
User Script Name #2=Enable Columns with IDENTIFY Clause
User Script Source #2=sql_definition_autoincrement.conf
User Script Target #2=sql_user_script_#2.conf

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Run user script before data conversion. To each table that will
;; have the data converted, this script will be executed before
;; the conversion.
;;
;; Defaults:
;; Types: Integer
```

```
Run user Script before data conversion=1

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Run user script after data conversion. To each table that will
;; have the data converted, this script will be executed before
;; the conversion.
;;
;; Defaults:
;; Types: Integer
Run user script after data conversion=2
```

Exemplo 48: Parte do arquivo "*default.ini*" do *Microsoft* SQL *Server*.

O exemplo 48 mostra a configuração dos *templates* de usuário para serem executados antes e depois da conversão de cada tabela com as entradas "*Run user Script before data conversion*" e "*Run user script after data conversion*".

5.1.3.1 Migração de domínios com os *templates* de usuário

É possível migrar praticamente tudo com o conversor, desde que seja possível obter informações a respeito do que se quer migrar através das tabelas de sistema do banco de dados de origem, e a partir de então configurar *templates* de destino.

A seguir será visualizado a utilização de *templates* de usuário para a conversão de domínios de um banco de dados *PostgreSQL* para um banco de dados *Firebird.* Primeiramente para a conversão deve ser configurado o *template* de origem para *PostgreSQL*, conforme no exemplo 49. Depois configurar o *template* de destino para o *Firebird* conforme o exemplo 50. E por fim, configurar o arquivo "*default.ini*" do destino como no exemplo 51.

```
SELECT
d.domain_name,
TRIM(CASE
       WHEN UPPER(d.data_type) = 'MONEY' THEN 'REAL'
       WHEN UPPER(d.data_type) = 'CHARACTER' THEN 'CHAR(' || COALESCE(d.numeric_precision,
d.character_maximum_length) || ')'
       WHEN UPPER(d.data_type) = 'CHARACTER VARYING' THEN 'VARCHAR(' ||
COALESCE(d.numeric_precision, d.character_maximum_length) || ')'
       WHEN UPPER(d.data_type) = 'NUMERIC' THEN 'NUMERIC(' || COALESCE(d.numeric_precision,
d.character_maximum_length) || ',' || COALESCE(d.numeric_scale, 0) || ')'
       WHEN UPPER(d.data_type) = 'DECIMAL' THEN 'DECIMAL(' || COALESCE(d.numeric_precision,
d.character_maximum_length) || ',' || COALESCE(d.numeric_scale, 0) || ')'
       WHEN UPPER(d.data_type) = 'TIME WITH TIME ZONE' THEN 'TIME'
       WHEN UPPER(d.data_type) = 'TIME WITHOUT TIME ZONE' THEN 'TIME'
       WHEN UPPER(d.data_type) = 'TIMESTAMP WITH TIME ZONE' THEN 'TIMESTAMP'
```

```
       WHEN UPPER(d.data_type) = 'TIMESTAMP WITHOUT TIME ZONE' THEN 'TIMESTAMP'
       ELSE
         UPPER(d.data_type)
     END) AS data_type,

 (SELECT CASE WHEN t.typnotnull THEN 'NOT NULL' END
 FROM pg_catalog.pg_type t
 LEFT JOIN pg_catalog.pg_namespace n ON
           n.oid = t.typnamespace
 WHERE n.nspname = d.domain_schema AND
       t.typname = d.domain_name) AS nulls,

'DEFAULT ' ||
 (CASE
  WHEN SUBSTR(UPPER(d.domain_default), 1, 19) = '(' || CHR(39) || 'NOW' || CHR(39) ||
'::TEXT)::DATE' THEN 'CURRENT_DATE'
  WHEN SUBSTR(UPPER(d.domain_default), 1, 19) = '(' || CHR(39) || 'NOW' || CHR(39) ||
'::TEXT)::TIME' THEN 'CURRENT_TIME'
  WHEN SUBSTR(UPPER(d.domain_default), 1, 3) = 'NOW' THEN 'CURRENT_TIMESTAMP'
  WHEN SUBSTR(UPPER(d.domain_default), 1, 7) = 'NEXTVAL' THEN ''
  WHEN SUBSTR(d.domain_default, 1, 1) = CHR(39) THEN SUBSTR(d.domain_default, 1, POSITION('::'
IN d.domain_default) - 1)
ELSE
  d.domain_default
END) AS default_column,

'CHECK ' || REPLACE(cc.check_clause, 'btrim(', 'TRIM(') AS check_rule

FROM
information_schema.domains d

LEFT JOIN information_schema.domain_constraints dc ON
dc.domain_catalog = d.domain_catalog AND
dc.domain_schema = d.domain_schema AND
dc.domain_name = d.domain_name

LEFT JOIN information_schema.check_constraints cc ON
cc.constraint_catalog = dc.constraint_catalog AND
cc.constraint_schema = dc.constraint_schema AND
cc.constraint_name = dc.constraint_name

WHERE
d.domain_name NOT IN ('cardinal_number', 'character_data', 'sql_identifier', 'time_stamp',
'earth', 'lo'); --system domains
```

Exemplo 49: *Template* de usuário "*sql_user_script_domain.conf*" para o *PostgreSQL*.

```
CREATE DOMAIN <domain_name> <data_type> <default_column> <nulls> <check_rule>;
```

Exemplo 50: *Template* de usuário "*sql_user_script_#1.conf*" para o *Firebird*.

```
User Script Name #1=Domain
User Script Source #1=sql_user_script_domain.conf
User Script Target #1=sql_user_script_#1.conf
```

Exemplo 51: Configuração dos *templates* anteriores.

As configurações anteriores serão capazes de migrar domínios como no exemplo 52.

```
CREATE DOMAIN TESTE NUMERIC(1) NOT NULL DEFAULT 1 CHECK (VALUE > 0);
CREATE DOMAIN TESTE2 VARCHAR(32) NOT NULL DEFAULT 'A' CHECK (VALUE <> 'B');
CREATE DOMAIN TESTE3 TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP CHECK (VALUE >
'2000-01-01 00:00:00.000');
CREATE DOMAIN TESTE4 INTEGER NULL;
```

Exemplo 52: Resultado dos *templates* anteriores.

Nesta situação o *template* de origem não utiliza o parâmetro de entrada "<*table_name*>", pois domínios são independentes de tabelas [Date, 2004]. Os parâmetros de entrada do *template* de destino é o resultado da consulta do *template* de origem, sendo o nome dos parâmetros o próprio nome das colunas.

5.1.4 Facilidades das tabelas do "INFORMATION_SCHEMA"

O padrão ANSI-SQL 99 define um *schema* que contém um conjunto de tabelas ou visões que retornam as informações internas dos bancos de dados, tais como nome de objetos, as suas definições e assim por diante. Este *schema* é chamado de "INFORMATION_SCHEMA" [ANSI-SQL 99].

Alguns bancos de dados, como o caso o *PostgreSQL* e o *Microsoft* SQL *Server*, possuem este conjunto de tabelas, tornando assim os *templates* que devem ser desenvolvidos similares. Por diferenças de sintaxe dos bancos de dados as consultas não são iguais, mas com grandes semelhanças. A comprovação do suporte do "INFORMATION_SCHEMA" para estes bancos pode ser vista no "Apêndice D".

Este padrão permite que novos sistemas gerenciadores de bancos de dados, os quais possam necessitar ser configurados para a ferramenta, possuam determinadas facilidades para serem criados, sendo que já existem bancos de dados configurados que utilizam estas tabelas.

6 ESTUDO DE CASOS

Este capítulo irá exemplificar a utilização da ferramenta em situações de migração, também apresentando as alterações no *layout* do projeto para tornar o software mais atrativo. Após será exibido o resultado de alguns testes de migração entre os sistemas gerenciadores de banco de dados homologados para o migrador, mostrando o tempo para cada migração. Isso inclui testes de migração de uma tabela que conterá alguns tipos de colunas definidas pelo padrão ANSI [ANSI-SQL 92; ANSI-SQL 99], e consultas sobre esta tabela para cada banco de dados migrado, visando à conferência das migrações. Por fim, o capítulo irá descrever os principais problemas sobre as migrações feitas para o desenvolvimento e testes do aplicativo, e quais as soluções para estas situações.

6.1 Exemplo de uso

Neste tópico será demonstrada a metodologia de uso do conversor. Foi migrado o banco de dados de testes que acompanha a instalação do *Sybase* ASA (incluindo algumas tabelas de testes) para o gerenciador *Microsoft* SQL *Server*. Entretanto alguns exemplos de migração entre outros bancos de dados serão visualizados, com o intuito de apresentar todas as situações possíveis.

Ao iniciar o aplicativo serão exibidas as conexões com as bases de dados, sendo necessário especificar o banco de dado de origem e o banco de dados de destino como visualizado na figura 15.

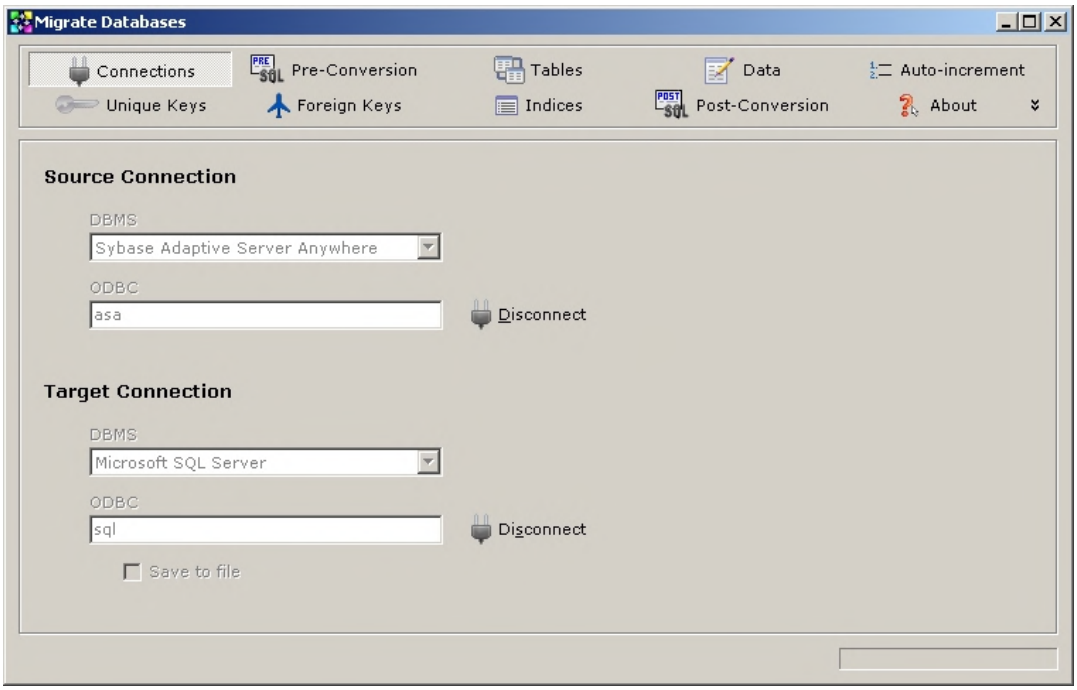


Figura 15: Tela de conexão com os bancos de dados.

Fonte: Própria.

Também é possível ao invés de converter diretamente o banco de dados de origem para destino, gerar comandos em arquivos texto. Para tal é necessário selecionar a opção "*Save to file*" e escolher o diretório aonde serão salvos os arquivos, conforme pode ser visualizado na figura 16.

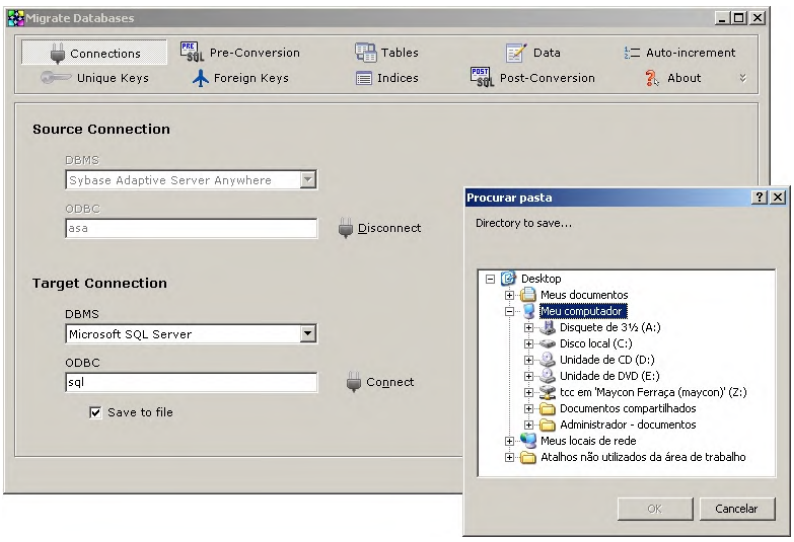


Figura 16: Salvar migração.

Fonte: Própria.

O próximo passo é a execução das pré-conversões de origem e destino caso existam. Para o exemplo da migração em vigor é possível visualizar na figura 17 a criação da função “TRIM” no *Microsoft* SQL *Server*.

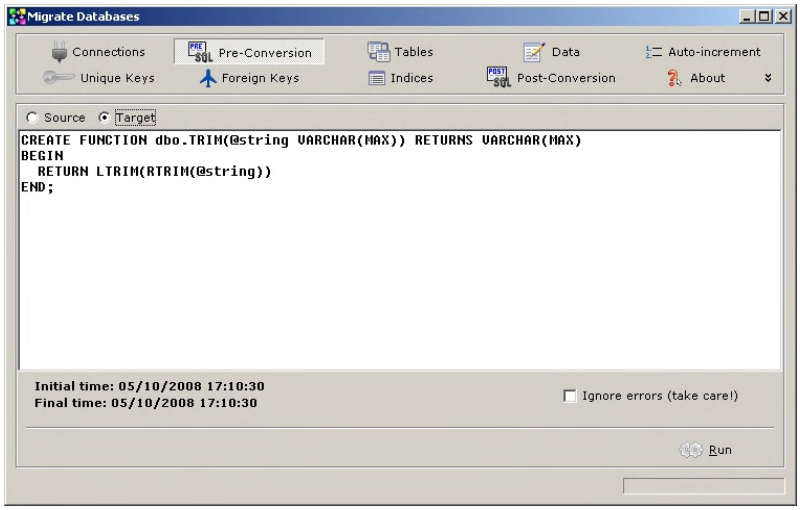


Figura 17: Execução dos comandos pré-conversão.

Fonte: Própria.

O próximo passo são as criações das tabelas no banco de dados de destino ou a gravação desta em arquivo textos (figura 18).

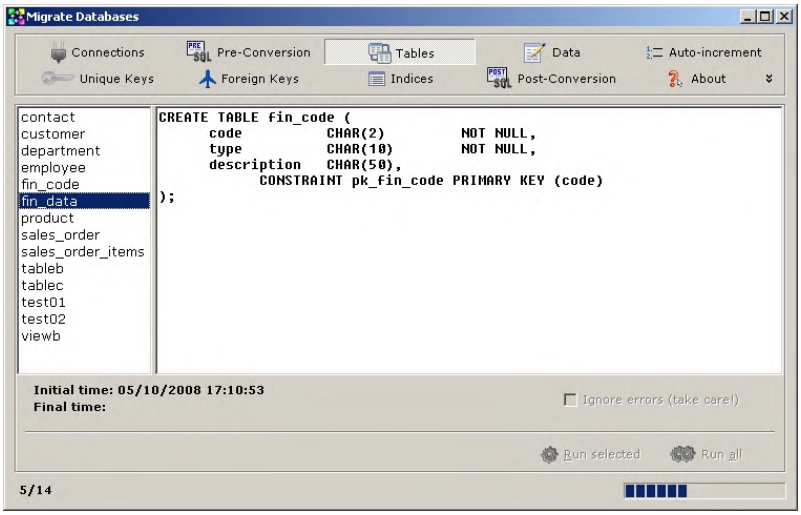


Figura 18: Criação das tabelas no conversor.

Fonte: Própria.

Para a migração dos dados, o procedimento será igual ao da página 52, contendo as seguintes opções (figura 19):

- **“*Tables n to n*” (“Tabelas da *n* até *n*”)**: Caso seja necessário fazer uma migração individual das tabelas é necessário informar aqui um intervalo de códigos, especificando o código inicial e final das tabelas para a conversão. É possível obter a listagem das tabelas na opção “*Table list*” (“Lista de Tabelas”, figura 20);
- **“*Start first table since the record n*” (“Iniciar primeira tabela a partir do registro n”)**: Para falhas em conversões, também será possível iniciar uma conversão de determinado registro de uma tabela;
- **“*Run "Commit" to each*” (“Executar “COMMIT” a cada *n* registros”)**: Especifica ao conversor a cada quantos registros convertidos deve ser efetuado um comando “COMMIT” para efetivar os dados já convertidos. Será criado um arquivo de *log* para cada “COMMIT” executado, podendo assim em caso de falha, corrigir o problema e continuar do último registro gravado, especificando na opção anterior (“*Start first table since the record n*”);
- **“*Paging records to each*” (“Paginar a cada *n* registros”)**: O componente de dados de acesso que será utilizado, o BDE possui um limite do número de registros que pode ser carregado em memória, este limite é cerca de 357000 (dependo o tamanho das linhas carregadas) [Leão, 2001]. Então internamente será efetuado um controle de paginação dos registros a serem convertidos, isto significa que o aplicativo irá carregar por vez a quantidade de registros especificada;
- **“*Ignore errors*” (“Ignorar erros”)**: Irá especificar ao sistema para ignorar qualquer erro durante o processo de conversão. Esta opção deverá ser usada com cuidado porque os erros não serão reportados para o usuário e somente gravados no *log* de erros ocorridos no processo de conversão. Isto será útil para forçar um processo de atualização.

O conversor irá grava um arquivo de *log* sobre cada comando “COMMIT” efetuado. E em caso de uma falha é possível saber qual foi o último registro convertido, especificá-lo na

opção “*Start first table since the record n*” a partir do último confirmado, em conjunto com a opção “*Tables n to n*”. Exemplos de todos os arquivos de *log* do conversor podem ser visto no “Apêndice E”.

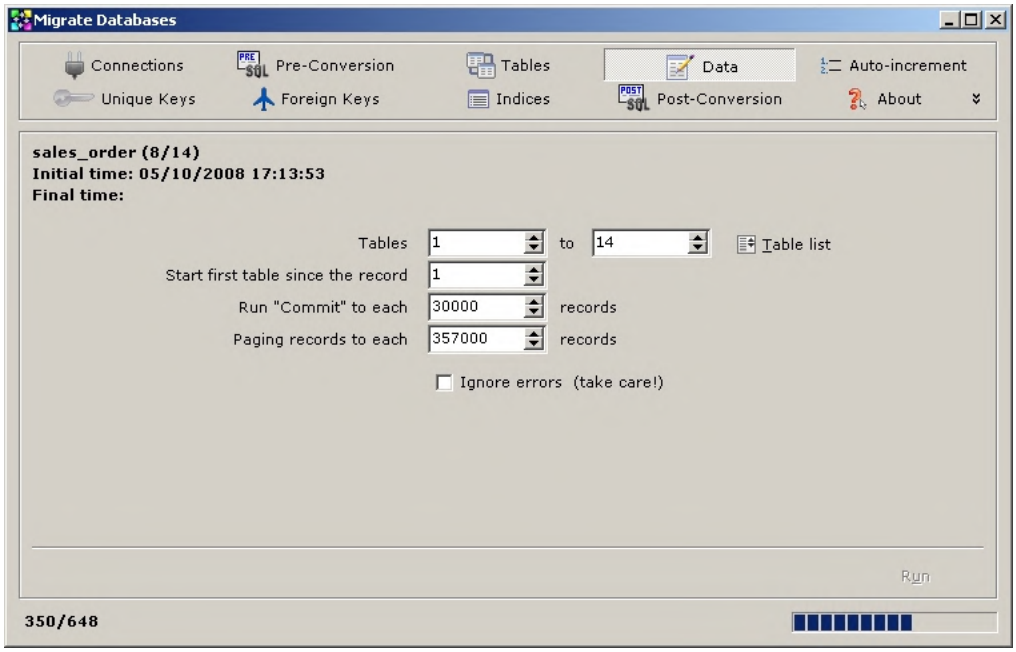


Figura 19: Conversão de dados no conversor.

Fonte: Própria.

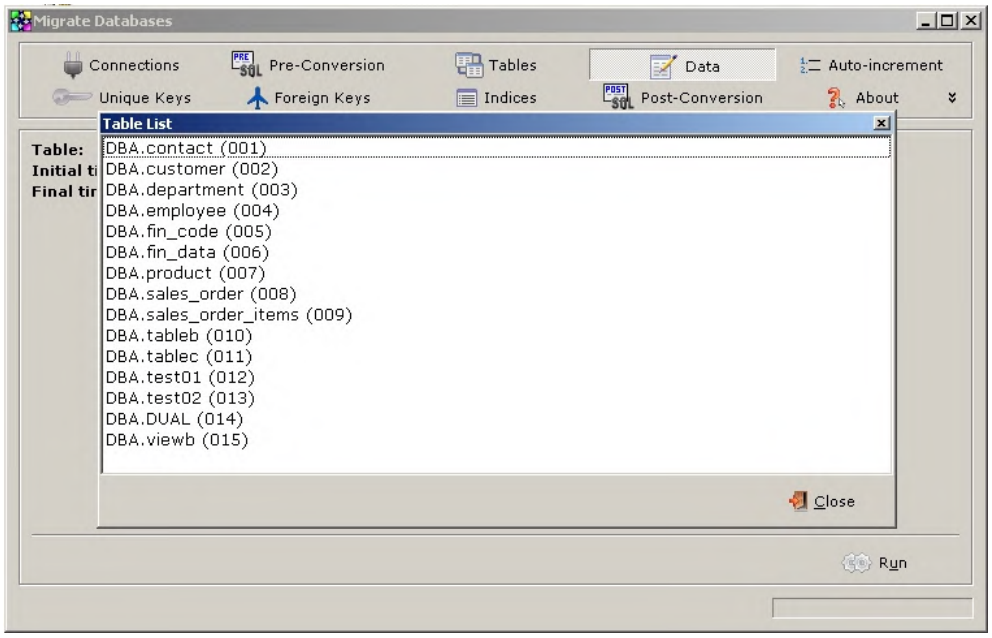


Figura 20: Lista das tabelas no *Converter*.

Fonte: Própria.

O próximo passo é executar a criação das colunas auto-incremento. Entretanto, o *Microsoft SQL Server* necessita que as colunas com incremento automático sejam realizadas na criação da tabelas (opção "*Tables*") [Dewson, 2006]. Para tal a figura 21 exemplifica a criação destas para o *Firebird*.

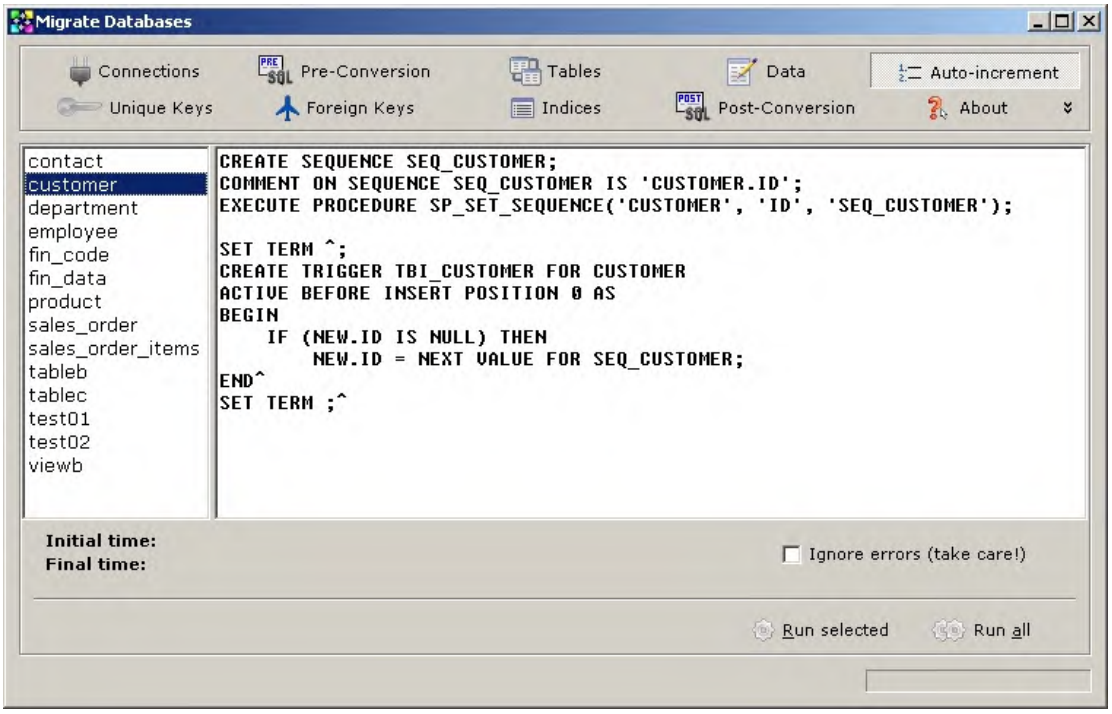


Figura 21: Criação de gatilhos do auto-incremento para colunas no *Converter*.

Fonte: Própria.

Na figura 22 é possível verificar a criação das chaves únicas para o banco de dados de destino.

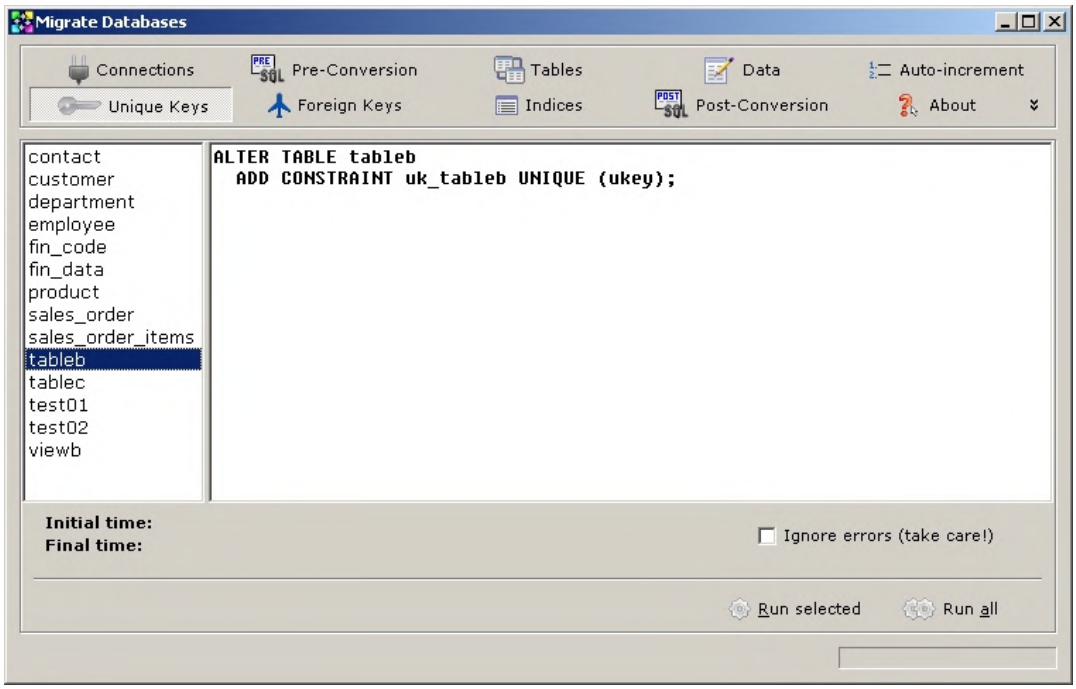


Figura 22: Criação das chaves únicas no conversor.

Fonte: Própria.

A seguir serão criadas as chaves estrangeiras, sendo necessário clicar na opção "*Foreign Keys*" como exibido na figura 23.

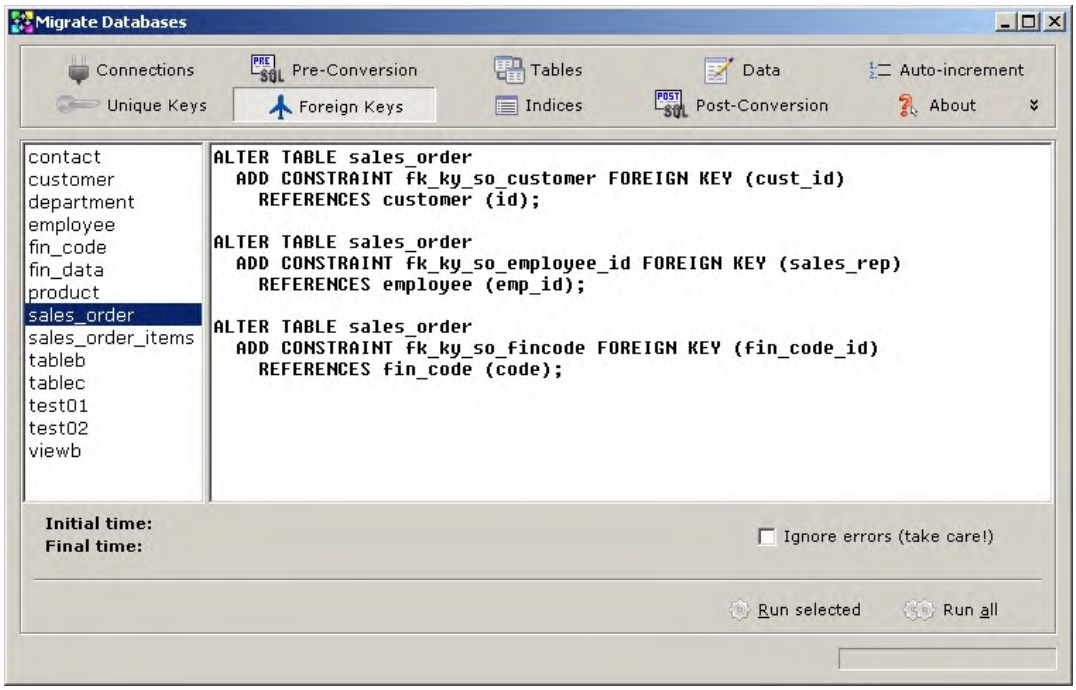


Figura 23: Criação das chaves estrangeiras no conversor

Fonte: Própria.

Para os índices existe a opção “*Indices*” (figura 24).

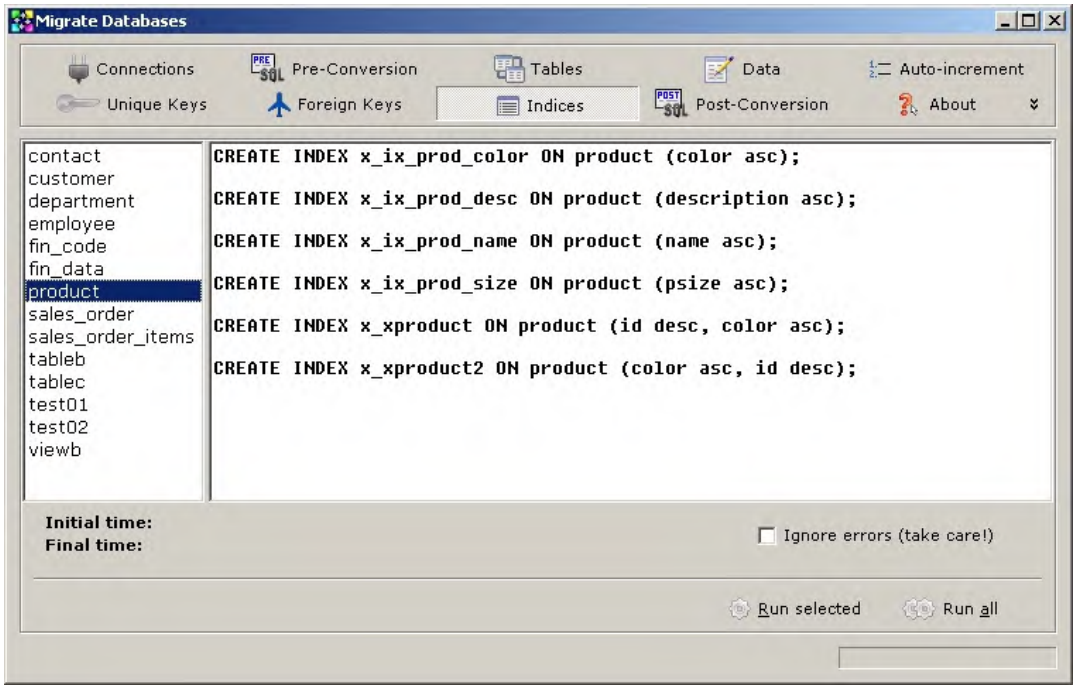


Figura 24: Criação dos índices no conversor.

Fonte: Própria.

O próximo passo é a execução das pós-conversões de origem e destino caso existam. Para o exemplo da migração em vigor é possível visualizar na figura 25 a criação da função da visão “DUAL” no *Microsoft* SQL *Server*.

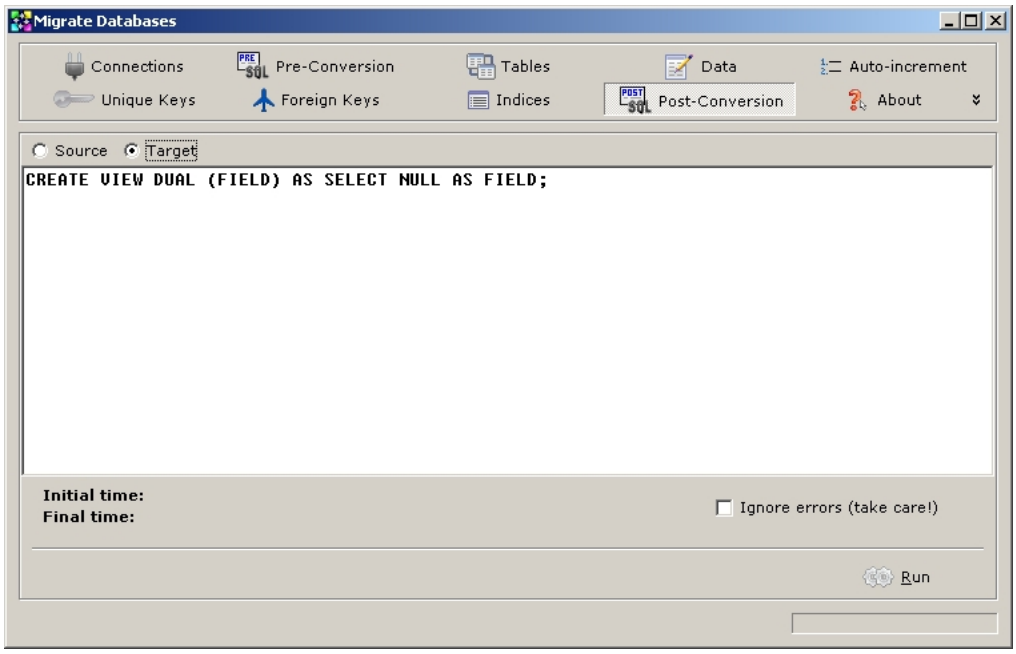


Figura 25: Execução dos comandos pós-conversão.

Fonte: Própria.

Para caso existam configurações personalizadas, é possível executá-las clicando na seta que aponta para baixo conforme figura 26. Neste caso os *templates* de usuários já são executados automaticamente com a conversão de dados.

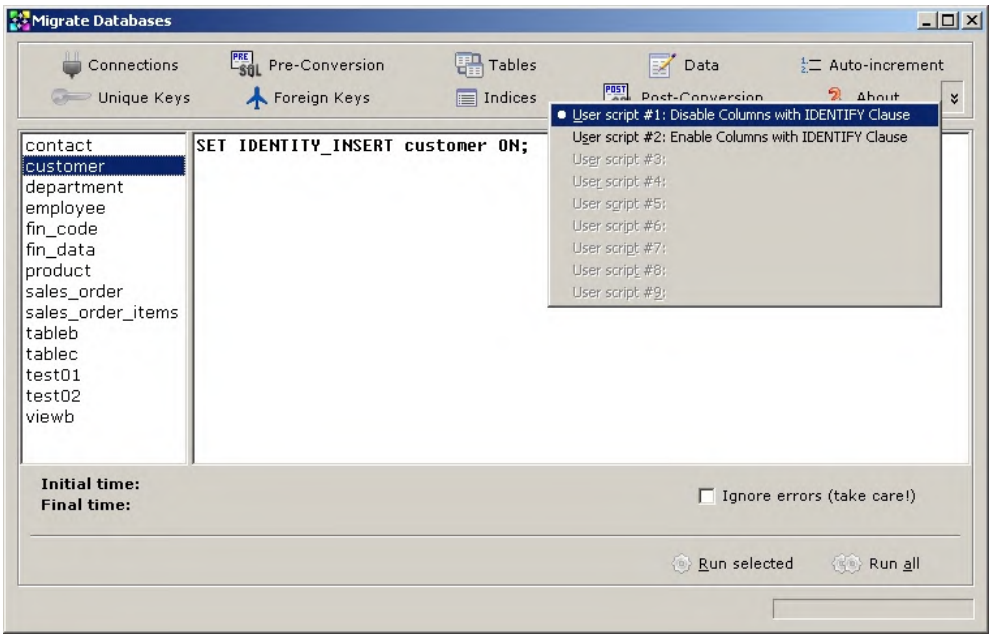


Figura 26: Execução dos *templates* de usuário no conversor.

Fonte: Própria.

A última opção do conversor é o botão “*About*”, figura 27, que mostra informações o sobre sistema.

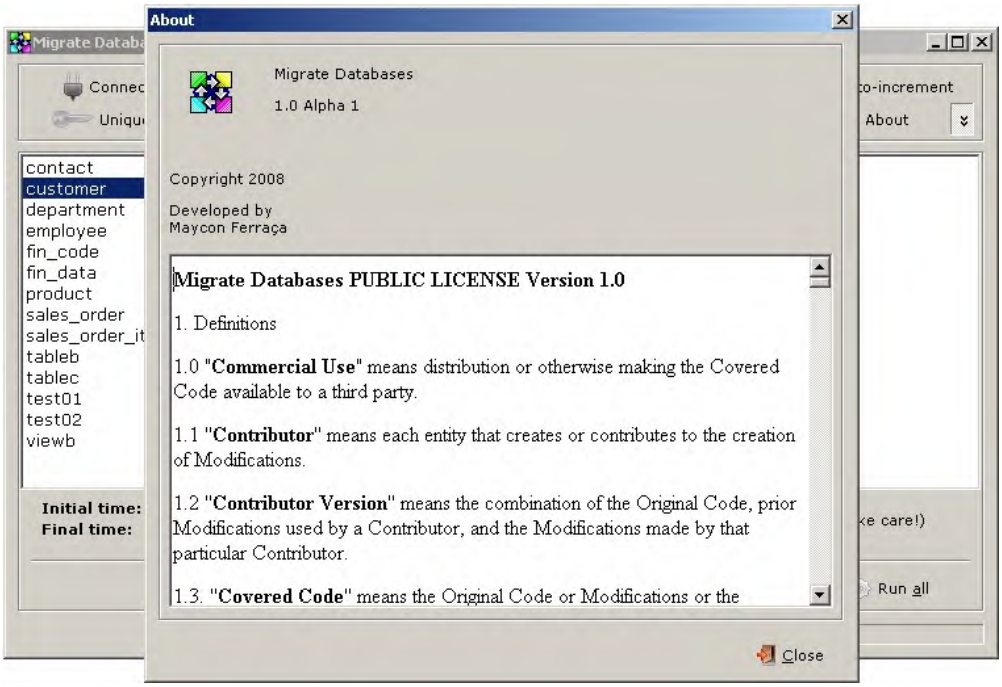


Figura 27: Tela de informações do sistema.

Fonte: Própria.

Caso a opção da conversão for por geração em arquivos, os arquivos gerados seriam:

- ***“01-PreConversion.sql”***: Com o *script* de geração dos comandos de pré-conversão no banco de dados de destino;
- ***“02-Table.sql”***: Comandos referentes a criação de todas a tabelas;
- ***“03-Data-TABELA.sql”***: Um arquivo destes será gerado para cada tabela do sistema. O valor que mudará será o nome da tabela para o arquivo gerado (posição 10 até uma posição antes do ponto da extensão do arquivo);
- ***“04-AutoIncrement.sql”***: Definição de todas colunas auto-incrementos geradas;
- ***“05-UniqueKey.sql”***: Definição de todas a chaves únicas para o banco de dados de destino;

- ***"06-ForeignKey.sql"***: Arquivo com todos os comandos para a criação das chaves estrangeiras;
- ***"07-Index.sql"***: Arquivo responsável para a criação dos índices no banco de dados de destino;
- ***"08-PostConversion.sql"***: *Script* que irá criar os comandos de pós-execução no banco de dados de destino;
- ***"09-01-UserScript.sql"***: Arquivos referentes aos comandos gerados pelos *templates* de usuários, mudando apenas para o nome do arquivo, o número do *template* (posição 4 e 5 do nome).

Estes arquivos devem ser executados no aplicativo correspondente do banco de dados de destino para submissão de consultas.

Alguns bancos podem conter muitas tabelas, tornando inviável à execução individual de todos os *scripts*. Para tal é possível criar um *template* de usuário para gerar um arquivo que faça a chamada de todos os outros, executando todos de uma só vez.

Por exemplo, para o *Sybase* ASA, configurando o *template* de destino visualizado no exemplo 53, o *template* de origem do exemplo 54 (neste caso o *PostgreSQL*) e o arquivo "*default.ini*" como no exemplo 55, é possível obter o arquivo "*09-01-UserScript.sql*" como no exemplo 56 para um banco de dados com as tabelas "A", "B" e "C".

Imaginando um banco com cerca de 500 tabelas, estas configurações poderiam economizar grandes esforços.

```
READ '03-Data-<table_name>.sql';
```

Exemplo 53: *Template* de usuário "*sql_user_script_#1.conf*" para o *Sybase* ASA.

```
SELECT '<table_name>' AS table_name;
```

Exemplo 54: *Template* de usuário "*sql_user_script_table_name.conf*" para o *PostgreSQL*.

```
User Script Name #1=Load tables
User Script Source #1=sql_user_script_table_name.conf
User Script Target #1=sql_user_script_#1.conf
```

Exemplo 55: Configuração dos *templates* anteriores.

```
READ '03-Data-a.sql';
READ '03-Data-b.sql';
READ '03-Data-c.sql';
```

Exemplo 56: Resultado dos *templates* anteriores.

6.2 Testes

Foram executados testes de conversão entre as bases de dados utilizando o conversor, com um banco de dados de 302 MB com mais de 500 tabelas. Cada migração foi executada conforme o especificado no tópico anterior (“Exemplo de uso”).

O intuito dos testes foi de obter possíveis erros não vistos durante o desenvolvimento, e retornar o tempo de conversão entre os bancos de dados. Apesar de serem medidos os tempos, não é escopo dos testes verificar o banco de dados mais rápido.

Da tabela 2 à tabela 22 foram medidos individualmente os tempos para as operações de conversão de tabelas, dados, colunas auto-incremento, chaves únicas, chaves estrangeiras e índices. As operações de pré-conversão e pós-conversão não foram incluídas porque todas as elas duraram menos de um segundo.

Para os testes foi utilizada uma máquina de bom porte, um Intel Core 2 Duo 2 de 40 GHz com 2 GB de RAM utilizando sistema operacional *Windows* XP *Professional Service Pack* 3. As versões dos SGBD utilizados foram o *Firebird* 2.1, *Microsoft* SQL *Server* 2005, *PostgreSQL* 8.3 e *Sybase* ASA 8.0.

Para o *Firebird* e o Sybase ASA o tamanho das páginas criadas foram de 16KB, para os demais SGBD foi utilizado o padrão. Qualquer outra configuração dos bancos também permaneceu padrão.

Foram convertidas no total 540 entidades e o tempo de conversão destas pode ser visualizado na tabela 17.

Tabela 17: Tempos de conversão de tabelas

Fonte: Própria.

		Destino			
		Firebird	***Microsoft* SQL *Server***	***PostgreSQL***	***Sybase* ASA**
Oigem	***Firebird***	00:00:34	00:00:53	00:00:56	00:00:32
	Microsoft* SQL *Server	00:06:36	00:06:29	00:06:36	00:06:18
	PostgreSQL	00:03:39	00:03:21	00:03:24	00:04:17
	***Sybase* ASA**	00:00:43	00:00:36	00:00:43	00:00:22

A tabela 18 mostra o tempo de conversão dos dados.

Tabela 18: Tempos de conversão de dados.

Fonte: Própria.

		Destino			
		Firebird	***Microsoft* SQL *Server***	***PostgreSQL***	***Sybase* ASA**
Oigem	***Firebird***	00:19:00	00:15:59	00:21:09	00:11:03
	Microsoft* SQL *Server	00:18:28	00:16:42	00:20:30	00:09:23
	PostgreSQL	00:33:09	00:26:08	00:24:16	00:20:49
	***Sybase* ASA**	00:25:07	00:21:59	00:29:37	00:19:38

Para tabela 19 é mostrado o tempo das 120 operações ocorridas para definição das colunas que utilizam auto-incremento.

Tabela 19: Tempos de conversão de colunas auto-incremento.

Fonte: Própria.

		Destino			
		Firebird	***Microsoft* SQL *Server***	***PostgreSQL***	***Sybase* ASA**
Oigem	***Firebird***	00:00:07	--	00:00:16	00:00:17
	Microsoft* SQL *Server	00:00:12	--	00:00:20	00:00:17
	PostgreSQL	00:00:06	--	00:00:14	00:00:16
	***Sybase* ASA**	00:00:10	--	00:00:08	00:00:20

A tabela 20 mostra o tempo das 2 chaves únicas criadas.

Tabela 20: Tempos de conversão de chaves únicas.

Fonte: Própria.

		Destino			
		Firebird	***Microsoft* SQL *Server***	***PostgreSQL***	***Sybase* ASA**
Oigem	***Firebird***	00:00:02	00:00:02	00:00:02	00:00:02
	Microsoft* SQL *Server	00:00:57	00:00:57	00:00:57	00:00:57
	PostgreSQL	00:00:06	00:00:06	00:00:07	00:00:06
	***Sybase* ASA**	00:00:10	00:00:10	00:00:10	00:00:11

A tabela 21 faz referência ao tempo de criação das 1.075 chaves estrengeiras.

Tabela 21: Tempos de conversão de chaves estrangeiras.

Fonte: Própria.

		Destino			
		Firebird	***Microsoft* SQL *Server***	***PostgreSQL***	***Sybase* ASA**
Oigem	***Firebird***	00:01:33	00:00:52	00:01:04	00:01:04
	Microsoft* SQL *Server	00:04:25	00:04:30	00:04:51	00:04:58
	PostgreSQL	00:25:38	00:24:26	00:23:49	00:25:55
	***Sybase* ASA**	00:02:21	00:01:52	00:01:03	00:02:25

Por fim, a tabela 22 exibe o tempo da criação dos 228 índices.

Tabela 22: Tempos de conversão de índices.

Fonte: Própria.

		Destino			
		Firebird	***Microsoft* SQL *Server***	***PostgreSQL***	***Sybase* ASA**
Oigem	***Firebird***	00:00:14	00:00:27	00:00:43	00:00:24
	Microsoft* SQL *Server	00:00:57	00:01:17	00:01:45	00:01:09
	PostgreSQL	00:00:14	00:00:31	00:00:43	00:00:28
	***Sybase* ASA**	00:00:19	00:00:32	00:00:53	00:00:37

O tempo total de conversão entre cada banco de dados pode ser visto na tabela 23 e no gráfico da figura 28. É possível ver gráficos individuais para cada conversão realizada no “Apêndice G”.

Tabela 23: Tempos totais de conversão.

Fonte: Própria.

		Destino			
		Firebird	***Microsoft* SQL *Server***	***PostgreSQL***	***Sybase* ASA**
Oigem	***Firebird***	00:21:30	00:18:13	00:24:10	00:13:49
	Microsoft* SQL *Server	00:31:35	00:29:55	00:34:59	00:23:02
	PostgreSQL	01:02:52	00:54:32	00:52:33	00:51:51
	***Sybase* ASA**	00:28:50	00:25:09	00:32:34	00:23:33

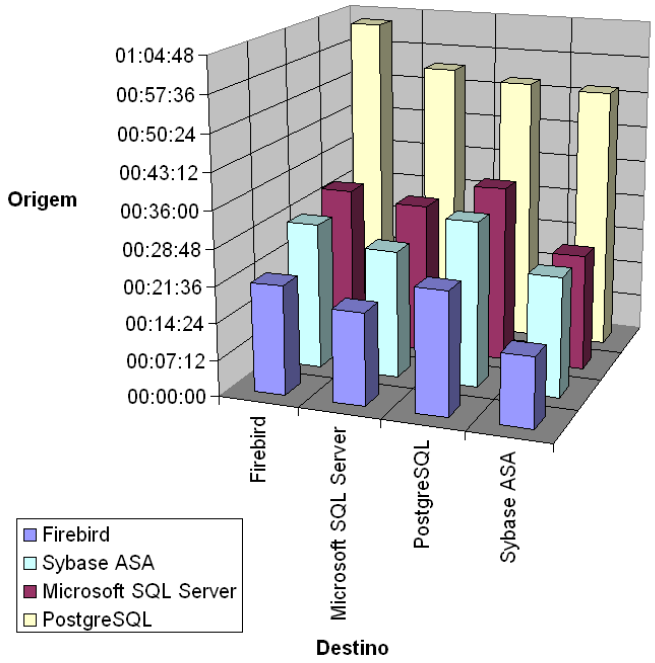


Figura 28: Gráfico de tempos totais de conversão.

Fonte: Própria.

O motivo de diferenças tão grandes dos tempos é devido a diversos fatores, velocidades individuais de cada operação para cada banco de dados. Entretanto, o ponto maior que se pode perceber é que principalmente as consultas dos *templates* desenvolvidos para o *Microsoft* SQL *Server* e o *PostgreSQL* necessitam otimizações. Estas otimizações ficarão para uma próxima etapa, que vai além deste trabalho.

Para testes de confiabilidade dos dados foi criada a tabela do exemplo 1, carregado com a inserção de 100 linhas aleatórias como no exemplo 58, e a partir de então foi submetido em cada banco de dados a consulta do exemplo 59. A tabela criada contém alguns tipos de dados definidos pelo padrão ANSI [ANSI-SQL 92; ANSI-SQL 99].

```
CREATE TABLE TESTE (
      A   BIGINT NOT NULL PRIMARY KEY,
      B   INTEGER,
      C   SMALLINT,
      D   DECIMAL(18,4),
      E   DECIMAL(10,2),
      F   NUMERIC(18,2),
      G   NUMERIC(10,2),
      H   DOUBLE PRECISION,
      I   REAL,
      L   TEXT,
      M   VARCHAR(10),
      N   CHAR(10),
      O   DATE,
      P   TIME,
      Q   TIMESTAMP
);
```

Exemplo 57: Criação de uma tabela para testes confiabilidade de migração.

```
INSERT INTO TESTE (A, B, C, D, E, F, G, H, I, L, M, N, O, P, Q) VALUES (-
1924761163, -1899383670, -16553, 5015.0617, 76225.96, 599735.11, 357701.04,
5539.7737, 0.339300006628036,
'ABCEABCEABCEABCEABCEABCEABCEABCEABCEABCEABCEABCEABCEABCEABCE', 'CBCT',
'KTIDBB', '2006-08-11', '05:19:48.000', '2007-10-11 14:52:12.000');
```

Exemplo 58: Carga da tabela de testes de confiabilidade de migração.

```
SELECT
MAX(A) AS MAX_A,
MIN(A) AS MIN_A,
MAX(B) AS MAX_B,
MIN(B) AS MIN_B,
MAX(C) AS MAX_C,
MIN(C) AS MIN_C,
SUM(D) AS SUM_D,
SUM(E) AS SUM_E,
SUM(F) AS SUM_F,
SUM(G) AS SUM_G,
SUM(H) AS SUM_H,
SUM(I) AS SUM_I,
MAX(CHAR_LENGTH(L)) AS MAX_CHAR_LENGTH_L,
MIN(CHAR_LENGTH(L)) AS MIN_CHAR_LENGTH_L,
MAX(CHAR_LENGTH(M)) AS MAX_CHAR_LENGTH_M,
MIN(CHAR_LENGTH(M)) AS MIN_CHAR_LENGTH_M,
MAX(CHAR_LENGTH(N)) AS MAX_CHAR_LENGTH_N,
MIN(CHAR_LENGTH(N)) AS MIN_CHAR_LENGTH_N,
MAX(O) AS MAX_O,
MIN(O) AS MIN_O,
```

```
MAX(P) AS MAX_P,
MIN(P) AS MIN_P,
MAX(Q) AS MAX_Q,
MIN(Q) AS MIN_Q

FROM
TESTE;
```

Exemplo 59: Consulta submetida na tabela de testes.

O resultado obtido foi igual para todas as consultas executadas, mostrando que a migração da tabela em questão ocorreu adequadamente. Entretanto, caso isto não ocorresse, não quer dizer necessariamente que o conversor não migre os dados corretamente, e sim que, as configurações no sistema podem não ter sido feitas corretamente para a conversão de um banco de dados para outro, já que diferentes bancos possuem características particulares, sejam elas em questões de sintaxe ou semântica.

6.3 Problemas e soluções das conversões

Durante o desenvolvimento e os testes vistos anteriormente foi possível obter uma série de erros de migração, derivadas das diferenças de características entre cada SGBD. Para cada problema encontrado foi apontado uma solução referente.

Irá perceber-se nos erros relatados que cada banco de dados possui as suas características específicas, e que a migração de um banco de dados para outro vai além dos recursos oferecidos pelo aplicativo desenvolvido. Os cuidados vão também aos estudos dos bancos que serão configurados.

6.3.1 Problemas relacionados ao *Firebird*

O SGBD *Firebird* faz restrição de um valor em uma coluna do tipo “NUMERIC” ou “DECIMAL” para a escala[9] declarada, mas a maneira como a precisão[10] é restrita é definida o campo como um tipo inteiro e somente haverá *overflow* se o valor restringir este tipo. A tabela 24 descreve quais tipos são utilizados internamente em quais situações.

Tabela 24: Declaração interna para campos NUMERIC e DECIMAL no *Firebird*.

Fonte: Borrie, 2006.

Tipo declarado	Precisão	Tipo armazenado
DECIMAL	1 a 4	SMALLINT
NUMERIC	1 a 4	INTEGER
NUMERIC e DECIMAL	5 a 9	INTEGER
NUMERIC e DECIMAL	10 a 18	BIGINT

Para identificar os decimais é gravado o fator de escala do campo. Por exemplo, o valor 1,603 é disposto no banco como 1603 e com a escala de -3 (i.e. 1603 * 10^-3) [Borrie, 2006].

Entretanto, isto foge do que declara o padrão ANSI [ANSI-SQL 92; ANSI-SQL 99] e outros bancos de dados por seguirem o padrão, não permitem valores do mesmo tipo de colunas do *Firebird*, ocorrendo erros como no exemplo 60. O exemplo 60 demonstra a inserção em uma tabela “teste” após a conversão de um banco de dados *Firebird* para *PostgreSQL*. A coluna “valor” do exemplo 60 é um “NUMERIC(5,2)”. O erro ocorrido faz referência ao *Firebird* aceitar valores maiores do que a precisão declarada e o *PostgreSQL* não.

```
INSERT INTO teste (valor) VALUES (900000);

General SQL error.
ERROR: numeric field overflow
A field with precision 5, scale 2 must round to an absolute value less than 10^3.;
```

Exemplo 60: Erro ao inserir um valor fora de faixa em campo do tipo “NUMERIC” no *PostgreSQL*.

[9] Escala são dígitos à direita da vírgula [Borrie, 2006].

[10] Precisão são dígitos à esquerda da vírgula [Borrie, 2006].

6.3.2 Problemas relacionados ao *Microsoft* SQL *Server*

Primeiramente houve problemas na configuração dos *templates* do *Microsoft* SQL *Server*. Os tipos de dados retornados para as colunas que continham campos texto não eram reconhecidos pelos componentes de acesso ao banco de dados utilizado pelo migrador. Como solução todas estas colunas sofreram uma conversão de dados para o tipo “TEXT” através do comando “CAST”.

Outro problema foi que no *Microsoft* SQL *Server* não é possível criar duas colunas como “IDENTIFY” ocasionando o erro visualizado no exemplo 61, sendo necessário evitar esta operação.

```
CREATE TABLE teste (
  a INTEGER IDENTITY,
  b INTEGER IDENTITY,
);

General SQL error.
[Microsoft][SQL Native Client][SQL Server]Multiple identity columns specified for
table 'tableb'. Only one identity column per table is allowed.
```

Exemplo 61: Erro ao criar uma tabela com duas colunas “IDENTIFY”.

Também não é possível inserir ou alterar valores das colunas “IDENTIFY”, sendo necessário desabilitar este recurso para a tabela com o comando do exemplo 46 (página 93), inserir os dados e habilitar novamente com o comando do exemplo 47 (página 93). Caso não for desabilitada a tabela, o erro do exemplo 62 ocorrerá.

```
INSERT INTO teste (a, b) VALUES (1, 1);

General SQL error.
[Microsoft][SQL Native Client][SQL Server] Cannot insert explicit value for
identity column in table 'teste' when IDENTITY_INSERT is set to OFF.
[Microsoft][SQL Native Client][SQL Server]The statement has been terminated.
```

Exemplo 62: Erro ao atribuir um valor para uma coluna “IDENTIFY”.

Entretanto, não é possível desabilitar duas tabelas (ou todas as tabelas) de uma vez só. Caso contrário erro do exemplo 63 irá ocorrer.

```
SET IDENTITY_INSERT teste ON;

General SQL error.
[Microsoft][SQL Native Client][SQL Server]IDENTITY_INSERT is already ON for table
'banco.dbo.teste2'. Cannot perform SET operation for table 'teste'.
```

Exemplo 63: Erro desabilitar mais de uma tabela que contenha uma coluna “IDENTIFY”.

O *Microsoft* SQL *Server* trata os identificadores como caso-sensitivo, não sendo possível ocorrer operações tais como no exemplo 64, aonde uma coluna é referenciada uma vez totalmente em minúsculo e outra vez totalmente em maiúsculo. Estas situações ocorrem porque o migrador não consegue alterar o nome dos identificadores em visões e *constraints* do tipo *check*, pois estas são obtidas como um comando compilado da base de dados de origem.

```
CREATE TABLE tableb (
  pkey INTEGER NOT NULL,
  nn2  NUMERIC(10,5) DEFAULT 11.99,
             CHECK((NN2 > 0) and (NN2 > 0)),
             CONSTRAINT pk_tableb PRIMARY KEY (pkey)
);

Invalid field name.
[Microsoft][SQL Native Client][SQL Server]Invalid column name 'NN2'.
```

Exemplo 64: Erro ao criar uma tabela referenciando o mesmo identificador diferentemente.

O *Microsoft* SQL *Server* também não permite a criação de chaves estrangeiras onde a relação entre estas contenha tipos de dados diferentes, por exemplo, uma coluna que seja do tipo “VARCHAR” e outra do tipo “CHAR” (exemplo 65).

```
ALTER TABLE teste
  ADD CONSTRAINT fk_teste FOREIGN KEY (cod_teste)
    REFERENCES teste2 (cod_teste);

General SQL error.
[Microsoft][SQL Native Client][SQL Server]Column 'teste2.cod_teste' is not the same
data type as referencing column 'teste.cod_teste' in foreign key 'fk_teste'.
```

Exemplo 65: Erro desabilitar mais de uma tabela que contenha uma coluna “IDENTIFY”.

O *Microsoft* SQL *Server* não permite criação de integridade referencial circular, pois estas, pela semântica do gerenciador, são possíveis de causar laços infinitos se um campo for atualizado em qualquer uma das tabelas referentes. A figura 1 mostra um exemplo de uma relação deste tipo.

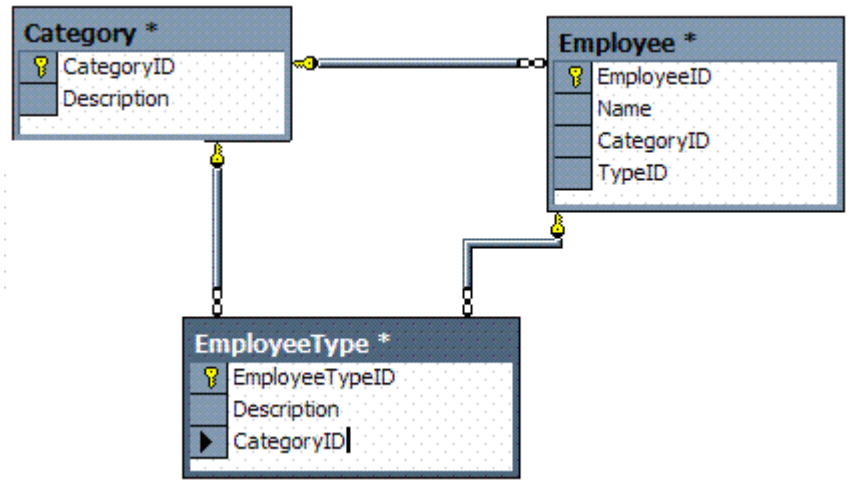


Figura 29: Integridade referencial circular.

Fonte: http://msdn.microsoft.com/pt-br/library/aa902657.aspx.

O exemplo da figura 1, segundo o site oficial da *Microsoft* (http://msdn.microsoft.com/pt-br/library/aa902657.aspx), diz que ao atualizar a coluna "*CategoryID*" da tabela "Category" fará com que o próximo campo "*CategoryID*" da tabela "*EmployeeType*" seja atualizado pela integridade referencial em cascata, fazendo com que o campo "*CategoryID*" da tabela "*Employee*" seja também atualizado e assim sucessivamente, ocasionando um laço infinito.

Entretanto alguns bancos de dados permitem este tipo de operação, como o *Sybase* ASA, e em uma migração de um banco de dados *Sybase* ASA para *Microsoft* SQL *Server* ocasionaria o erro do exemplo 66.

```
CREATE TABLE Category (
  CategoryID INTEGER PRIMARY KEY,
  Description VARCHAR(50));

CREATE TABLE Employee (
  EmployeeID INTEGER PRIMARY KEY,
```

```
  Name VARCHAR(50),
  CategoryID INTEGER,
  TypeID INTEGER);

CREATE TABLE EmployeeType (
  EmployeeTypeID INTEGER PRIMARY KEY,
  Description VARCHAR(50),
  CategoryID INTEGER);

ALTER TABLE Employee
  ADD CONSTRAINT fk_Category_Employee FOREIGN KEY (CategoryID)
    REFERENCES Category (CategoryID) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE Employee
  ADD CONSTRAINT fk_Employee_EmployeeType FOREIGN KEY (TypeID)
    REFERENCES EmployeeType (EmployeeTypeID) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE EmployeeType
  ADD CONSTRAINT fk_Category_EmployeeType FOREIGN KEY (CategoryID)
    REFERENCES Category (CategoryID) ON DELETE CASCADE ON UPDATE CASCADE;

General SQL error.
[Microsoft][SQL Native Client][SQL Server]Introducing FOREIGN KEY constraint
'fk_Category_EmployeeType' on table 'EmployeeType' may cause cycles or multiple
cascade paths. Specify ON DELETE NO ACTION or ON UPDATE NO ACTION, or modify other
FOREIGN KEY constraints.
```

Exemplo 66: Erro criar uma integridade referencial circular no *Microsoft* SQL *Server*.

Para evitar estes erros é necessário evitar a criação destes tipos de integridades referenciais circulares com chaves estrangeiras que possuam as propriedades “ON DELETE CASCADE” e “ON UPDATE CASCADE” na sua definição.

6.3.3 Problemas relacionados ao *PostgreSQL*

Através das conversões executadas foi visto que o *PostgreSQL* converte uma *constraint* do tipo *check* do exemplo 67 para o exemplo 68. O conversor ao ler a instrução gravada e tentar converter para um banco de dados em *Firebird*, *Microsoft* SQL *Server* ou *Sybase* ASA resulta em erro, como apresentado no exemplo 69. Para tal situação foram utilizadas o parâmetro do arquivo “*default.ini*” de origem do *PostgreSQL*, “*Ignored Words*” e o parâmetro dos arquivos “*default..ini*” dos arquivos dos bancos de destino (incluindo o próprio *PostgreSQL*), “*Replace Words*” para atualizar a *check constraint* para o valor de original, passando assim nas conversões.

```
CREATE TABLE TABELA (
  CHAVE INTEGER    NOT NULL,
  TESTE VARCHAR(1) NOT NULL,
    CONSTRAINT CKC_TESTE CHECK (teste IN ('S', 'N', 'E')),
    CONSTRAINT PK_TABELA PRIMARY KEY (CHAVE) USING INDEX XU_TABELA
);
```

Exemplo 67: Criação de uma *check constraint* no *Firebird*.

```
CREATE TABLE TABELA (
  CHAVE INTEGER    NOT NULL,
  TESTE VARCHAR(1) NOT NULL,
    CONSTRAINT CKC_TESTE CHECK ((teste) = ANY ((ARRAY['S'::character varying,
'N'::character varying, 'E'::character varying])[])),
    CONSTRAINT PK_TABELA PRIMARY KEY (CHAVE) USING INDEX XU_TABELA
);
```

Exemplo 68: Criação incorreta de uma *check constraint* no *Firebird*.

```
General SQL error.
            [ODBC Firebird Driver][Firebird]Dynamic SQL Error
            SQL error code = -104
            Token unknown - line 1, column 545
            (
```

Exemplo 69: Erro da execução da criação incorreta de uma *check constraint* no *Firebird*.

Também é encontrado em visões e *check constraint* armazenadas no *PostgreSQL* instruções como "*::numeric*", "*::text*", "*::timestamp without time zone*", "*::character varying*", entre outras. Pelo o que foi identificado, estas instruções foram localizadas ao lado de campos e identificam o seu tipo. Entretanto, para os outros bancos homologados estas instruções não são válidas. Para tal, foi também utilizado a opção "*Ignored Words*" do parâmetro de origem do *PostgreSQL* ("*default.ini*"), com o intuito de ignorar estas expressões.

Percebeu também, através dos testes que instruções como "*CAST(campo AS VARCHAR(25))*" são convertidas para "*(campo)::character varying(25)*". Para corrigir problemas como estes é necessário utilizar a opção "*Ignored Words*", ou alterar a instrução gravada no banco de antes de ser executada, através da opção de pré-conversão de bancos de origem.

6.3.4 Problemas relacionados ao *Sybase Adaptive Server Anywhere*

No *Sybase* ASA, comparações entre colunas do tipo “CHAR” e “VARCHAR” ignoram espaços no início ou no final dos campos, fazendo com que relações entre chaves estrangeiras possam possuir espaços em uma coluna de origem e no destino não, por exemplo. Isto implica diretamente na migração para outros sistemas gerenciadores de banco de dados, ocorrendo erros nas migrações (exemplo 70).

```
ALTER TABLE teste
  ADD CONSTRAINT fk_teste FOREIGN KEY (cod_teste)
    REFERENCES teste2 (cod_teste) ON DELETE SET NULL;

General SQL error.
ERROR: insert or update on table "teste" violates foreign key constraint "fk_teste"
Key (cod_teste)=( ) is not present in table "teste2".
```

Exemplo 70: Erro ao executar a criação de uma chave estrangeira no *Sybase* ASA.

Também ocorrerão problemas na criação de uma chave estrangeira originadas de bancos de dados que tenham o comportamento idêntico ao do *Firebird*. No *Firebird*, chaves únicas e índices únicos permitem que as colunas envolvidas contenham estados nulos (desde que não definidas como “NOT NULL”). Ao contrário do *Sybase* ASA, que somente possibilita esta característica para índices únicos [Ferraça, 2007]. Ao definir-se uma chave única no *Sybase* ASA, originada de uma conversão do *Firebird* que nas colunas envolvidas estejam definidas para aceitarem valores nulos, a criação falha, conforme visualizada no exemplo 71.

```
ALTER TABLE tabela
  ADD CONSTRAINT uk_tabela_chave UNIQUE (chave);

Key violation.
[Sybase][ODBC Driver][Adaptive Server Anywhere]Integrity constraint violation:
Column 'chave' in table 'tabela' cannot be NULL
```

Exemplo 71: Erro ao executar a criação de uma chave única no *Sybase* ASA.

6.3.5 Problemas relacionados aos tipos de dados

Devido a diversos erros de conversão de tipos de dados entre os bancos homologados, foi criada uma subseção específica para relatar estes problemas.

No *Microsoft* SQL *Server*, o tipo de dados de "DATETIME" permite apenas faixas de data entre 1 de janeiro de 1753 até 31 de dezembro de 9999 [Machanic, 2007]. Caso seja inserido um valor fora desta faixa, irá ocorrer o erro do exemplo 72. Como solução para esta situação é necessário corrigir os valores antes de serem inseridos da base de dados de origem para a base de dados de destino.

```
INSERT INTO teste (data) VALUES ('0404-06-17');

General SQL error.
[Microsoft][SQL Native Client][SQL Server]The conversion of a char data type to a
datetime data type resulted in an out-of-range datetime value.
```

Exemplo 72: Erro ao inserir datas fora da faixa permitida pelo *Microsoft* SQL *Server*.

Os bancos de dados *Firebird* [Firebird, 2008], *PostgreSQL* [Matthew *et al*., 2005] e o *Sybase* ASA [*Sybase*, 2001] possuem três tipos de dados, "TIMESTAMP" que armazena a data e hora, "TIME" que armazena somente as horas e o tipo "DATE" que armazena somente data, conforme o que especifica o padrão [ANSI-SQL 92; ANSI-SQL 99]. Entretanto, o *Microsoft* SQL *Server* não possui nenhum destes tipos de dados. O que o banco possui é apenas os tipos "DATETIME" e "SMALLDATIME" que armazenam a data e hora. A diferença entre os dois tipos está apenas na faixa de datas que cada um pode guardar [Dewson, 2006].

Nesta situação uma conversão entre banco de dados *Microsoft* SQL *Server* para outros bancos todas as datas devem ser convertidas para o tipo "TIMESTAMP". Na circunstância oposta (os outros bancos para o *Microsoft* SQL *Server*), um tipo de dados "TIME" será convertido para "DATETIME", guardando apenas informações corretas da hora e não da data.

Também no *Microsoft* SQL *Server* é não possível inserir datas com quatro dígitos nos milisegundos, como na data "2008-07-21 13:41:49.1400". Caso contrário o erro do exemplo 73 poderá ocorrer. O permitido são três dígitos.

```
INSERT INTO teste (data_sistema) VALUES ('2008-07-21 13:41:49.1400');

General SQL error.
[Microsoft][SQL Native Client][SQL Server]Conversion failed when converting
datetime from character string.
```

Exemplo 73: Erro ao inserir uma data com quatro dígitos nos milisegundos no *Microsoft* SQL *Server*.

E isto acontece em conversões vindas do *Firebird*, por o SGBD conter quatro dígitos para milisegundos. Como solução foi alterado os scripts "*sql_format_column_timestamp.conf*" e "*sql_format_column_time.conf*" para: "*LEFT(<column_name> || '', CHAR_LENGTH(<column_name> || '') - 1) AS <column_name>*" ao invés de "*<column_name> || '' AS <column_name>*", para retirar o último zero desnecessário. No caso nenhuma precisão é perdida, pois o *Firebird* traz somente zeros para o último dígito [Firebird, 2008].

No *Microsoft* SQL *Server*, colunas do tipo "VARCHAR" possuem limite de 8000 posições [Dewson, 2006], não sendo possível criar mais do que o limite estabelecido, caso contrário, ocorrerá o erro do exemplo 74:

```
CREATE TABLE teste (
  texto VARCHAR(10000) NOT NULL
);


General SQL error.
[Microsoft][SQL Native Client][SQL Server]The size (10000) given to the column
'texto' exceeds the maximum allowed for any data type (8000).
```

Exemplo 74: Erro criar um campo do tipo "VARCHAR" que seja maior do que 8000 caracteres.

Os demais tipos de dados permaneceram praticamente os mesmo, com mudanças apenas de nomes, por exemplo, campo texto longo para o *Firebird* é chamado de "BLOB" [Borrie, 2006] e para os outros bancos *Microsoft* SQL *Server* como "TEXT" [Dewson, 2006].

6.3.6 Outros problemas

Na migração de algumas bases de dados, a criação de visões e *constraints* do tipo *check* falham por não existirem funções referenciadas no banco de dados de origem no banco de dados de destino, como acontece com a função “TRIM” existentes no Sybase ASA [*Sybase*, 2001] e não existir no *Microsoft* SQL *Server* [Dewson, 2006]. O mesmo problema pode ocorrer na migração de outras bases de dados, como no exemplo 75 de uma migração de um banco de dados *Sybase* ASA para *Firebird*.

```
CREATE VIEW teste_v (texto) AS SELECT SUBSTR(texto, 1, 25) texto FROM teste;

General SQL error.
[ODBC Firebird Driver][Firebird]Dynamic SQL Error
SQL error code = -804
Function unknown
SUBSTR
```

Exemplo 75: Erro ao criar uma visão no *Firebird*.

Por fim, um caractere não tratado pelo conversor era o caractere de código zero pela representação da tabela ASCII. E uma das situações de testes de conversão, um campo continha este caractere e o *Delphi* trocou a *string* neste ponto. Segundo a documentação da linguagem, este é um caractere que identifica término de uma *string*. O exemplo 76 mostra a conversão como ficou o comando que foi executado.

```
INSERT INTO teste (texto) VALUES ('
```

Exemplo 76: Execução de um comando truncado pelo caractere número zero da tabela ASCII.

CONSIDERAÇÕES FINAIS

Pode-se concluir que o trabalho conseguiu atingir o seu objetivo, identificando o problema e desenvolvendo a solução: a construção de um aplicativo de migração independente das bases de dados de origem e destino.

A independência das bases de origem e destino é obtida a partir da definição dos *templates*, uma estrutura flexível e configurável. A desvantagem desta técnica é a configuração do aplicativo, que pode ser no início um pouco complexo e demorado, devido à quantidade de *templates* a serem configurados. Entretanto, espera-se que o uso do aplicativo por mais usuários e a continuidade do seu desenvolvimento, possibilite que mais SGBD sejam homologados.

Como parte da solução foi também definido o projeto, a proposta de desenvolvimento, as restrições de escopo do aplicativo, a distribuição, a licença, a configuração da ferramenta, a metodologia de uso e a homologação de quatro banco de dados, o *Firebird 2.1*, *Microsoft* SQL *Server* 2005, *PostgreSQL* 8.3 e o *Sybase Adaptive Server Anywhere 8*.

Algumas funcionalidades necessárias podem ter sido passadas despercebidas pelo projeto, mas como o sistema é de código aberto, outros desenvolvedores poderão trabalhar em cima do código-fonte original caso necessário. Algumas outras funcionalidades não serão implementadas agora por não haver tempo hábil para o desenvolvimento, e por também não tratar de pendências de grande importância.

Para trabalhos futuros ficarão a homologação de mais bases de dados, entre elas o *Oracle*, *MySQL*, *Ingress*, entre outros; otimizações das consultas dos *templates* já desenvolvidos, para maiores velocidades de conversão; melhoria nas configurações do aplicativo, trazendo interfaces gráficas para estes fins; revisão de todos os tipos de dados utilizados definidos pelo padrão SQL ANSI; além de atender solicitações de futuros usuários da ferramenta.

REFERÊNCIAS BIBLIOGRÁFICAS

ANSI-SQL 92. **Database Language SQL, ANSI/ISO/IEC International Standard (IS)**, ISO/IEC 9075:1992

ANSI-SQL 99. **Database Language SQL – Part 2: Foundation (SQL/Foundation), ANSI/ISO/IEC International Standard (IS)**, ISO/IEC 9075:1999

BORRIE, Helen. **Dominando Firebird**, 1. ed. Rio de Janeiro: Editora Ciência Moderna, 2006

DATE, Christopher J. **Introdução a Sistemas de Banco de Dados**, 8. ed. São Paulo: Editora Campus, 2004

DEWSON, Robin. **Beginning SQL Server 2005 for Developers**, New York: Apress, 2006

ELMASRI, Ramez; NAVATHE, Shamkant. **Sistemas de bancos de dados**, 4. ed. São Paulo: Pearson, 2005.

FERRAÇA, Maycon. **Adaptação do ERP Factory para o Firebird**. Caxias do Sul, RS. Brasil: Barcharelado de Sistemas de Informação, 2007. Relatório de Estágio Curricular.

FIREBIRD. **Firebird 2.0.4 Release Notes**, disponível em: <http://www.firebirdsql.org/rlsnotes/Firebird-2.0.1-ReleaseNotes.pdf>, Acesso em: 15 de outubro de 2008.

FOWLER, Martin. **Padrões de Arquitetura de Aplicações Corporativas**, 1. ed. São Paulo: Bookman, 2006.

GARCIA, Hector Molina; ULLMAN, Jeffrey; WIDOM, Jennifer. **Implementação de Sistemas de Banco de Dados**, Rio de Janeiro: Campus, 2001.

GIUSTINA, Michele Telles Della. **Extração e Integração de Fontes de Dados Heterogêneas para Repositório de Consulta**. Caxias do Sul, RS. Brasil: Barcharelado de Ciências da Computação, 2002. Monografia de Trabalho de Conclusão de Curso.

HORSTMANN, Cay. **Core Java 2**, 1. ed. Rio de Janeiro: Alta Books, 2005.

LARMAN, Craig. **Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos**. Porto Alegre: Bookman, 2000.

LEÃO, Marcelo. **Delphi 6 e Kylix Curso Completo**. 1. ed. Rio de Janeiro: Axcel Books, 2001

MACHANIC, Adam. **Expert SQL Server 2005 Development**, New York: Apress, 2007

MATTHEW, Neil; STONES, Richard. **Beginning Databases with PostgreSQL From Novice to Professional**, New York: Apress, 2005

MEDEIROS, Ernani. **Desenvolvendo software com UML 2.0**, 1. ed. São Paulo: Makron Books, 2006.

RODRIGUES, Anderson Haertel. **Sistemas Multicamadas com Delphi DataSnap e dbExpress**. Florianópolis: Visual Books, 2002.

SYBASE Inc. **SQL Anywhere Studio 8.0.0**, 2001. Manual do Produto.

VANDEVOORDE David; JOSUTTIS Nicolai. **C++ Templates: The Complete Guide**, Boston: Addison Wesley, 2002.

WOOD, Charles. **Usando o Powerbuilder**, 1. ed. Rio de Janeiro: Editora Campus, 1995.

APÊNDICES

APÊNDICE A – LICENÇA DO MIGRATE DATABASES

Migrate Databases PUBLIC LICENSE Version 1.0

1. Definitions

1.0 "**Commercial Use**" means distribution or otherwise making the Covered Code available to a third party.

1.1 "**Contributor**" means each entity that creates or contributes to the creation of Modifications.

1.2 "**Contributor Version**" means the combination of the Original Code, prior Modifications used by a Contributor, and the Modifications made by that particular Contributor.

1.3. "**Covered Code**" means the Original Code or Modifications or the combination of the Original Code and Modifications, in each case including portions thereof.

1.4. "**Electronic Distribution Mechanism**" means a mechanism generally accepted in the software development community for the electronic transfer of data.

1.5. "**Executable**" means Covered Code in any form other than Source Code.

1.6. "**Initial Developer**" means the individual or entity identified as the Initial Developer in the Source Code notice required by Exhibit A.

1.7. "**Larger Work**" means a work which combines Covered Code or portions thereof with code not governed by the terms of this License.

1.8. "**License**" means this document.

1.8.1. "**Licensable**" means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently acquired, any and all of the rights conveyed herein.

1.9. "**Modifications**" means any addition to or deletion from the substance or structure of either the Original Code or any previous Modifications. When Covered Code is released as a series of files, a Modification is:

Any addition to or deletion from the contents of a file containing Original Code or previous Modifications.

Any new file that contains any part of the Original Code or previous Modifications.

1.10. "**Original Code**" means Source Code of computer software code which is described in the Source Code notice required by Exhibit A as Original Code, and which, at the time of its release under this License is not already Covered Code governed by this License.

1.10.1. "**Patent Claims**" means any patent claim(s), now owned or hereafter acquired, including without limitation, method, process, and apparatus claims, in any patent Licensable by grantor.

1.11. "**Source Code**" means the preferred form of the Covered Code for making modifications to it, including all modules it contains, plus any associated interface definition files, scripts used to control compilation and installation of an Executable, or source code differential comparisons against either the Original Code or another well known, available Covered Code of the Contributor's choice. The Source Code can be in a compressed or archival form, provided the appropriate decompression or de-archiving software is widely available for no charge.

1.12. "**You**" (or "**Your**") means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License or a future version of this License issued under Section 6.1. For legal entities, "You" includes any entity w hich controls, is controlled by, or is under common control with You. For purposes of this definition, "control" means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. Source Code License.

2.1. The Initial Developer Grant. The Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license, subject to third party intellectual property claims:

(a) under intellectual property rights (other than patent or trademark) Licensable by Initial Developer to use, reproduce, modify, display, perform, sublicense and distribute the Original Code (or portions thereof) with or without Modifications, and/or as part of a Larger Work; and

(b) under Patents Claims infringed by the making, using or selling of Original Code, to make, have made, use, practice, sell, and offer for sale, and/or otherwise dispose of the Original Code (or portions thereof).

(c) the licenses granted in this Section 2.1(a) and (b) are effective on the date Initial Developer first distributes Original Code under the terms of this License.

d) Notwithstanding Section 2.1(b) above, no patent license is granted:

1) for code that You delete from the Original Code;

2) separate from the Original Code; or

3) for infringements caused by:

i) the modification of the Original Code or

ii) the combination of the Original Code with other software or devices.

2.2. Contributor Grant. Subject to third party intellectual property claims, each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license

(a) under intellectual property rights (other than patent or trademark) Licensable by Contributor, to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof) either on an unmodified basis, with other Modifications, as Covered Code and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of: 1) Modifications made by that Contributor (or portions thereof); and 2) the combination of Modifications made by that Contributor with its Contributor Version (or portions of such combination).

(c) the licenses granted in Sections 2.2(a) and 2.2(b) are effective on the date Contributor first makes Commercial Use of the Covered Code.

(d) Notwithstanding Section 2.2(b) above, no patent license is granted:

1) for any code that Contributor has deleted from the Contributor Version;

2) separate from the Contributor Version;

3) for infringements caused by: i) third party modifications of Contributor Version or

ii) the combination of Modifications made by that Contributor with other software (except as part of the Contributor Version) or other devices; or

4) under Patent Claims infringed by Covered Code in the absence of Modifications made by that Contributor.

3. Distribution Obligations.

3.1. Application of License. The Modifications which You create or to which You contribute are governed by the terms of this License, including without limitation Section 2.2. The Source Code version of Covered Code may be distributed only under the terms of this License or a future version of this License released under Section 6.1, and You must include a copy of this License with every copy of the Source Code You distribute. You may not offer or impose any terms on any Source Code version that alters or restricts the applicable version of this License or the recipients' rights hereunder. However, You may include an additional document offering the additional rights described in Section 3.5.

3.2. Availability of Source Code. Any Modification which You create or to which You contribute must be made available in Source Code form under the terms of this License either on the same media as an Executable version or via an accepted Electronic Distribution Mechanism to anyone to whom you made an Executable version available; and if made available via Electronic Distribution Mechanism, must remain available for at least twelve (12) months after the date it initially became available, or at least six (6) months after a subsequent version of that particular Modification has been made available to such recipients. You are responsible for ensuring that the Source Code version remains available even if the Electronic Distribution Mechanism is maintained by a third party.

3.3. Description of Modifications. You must cause all Covered Code to which You contribute to contain a file documenting the changes You made to create that Covered Code and the date of any change. You must include a prominent statement that the Modification is derived, directly or indirectly, from Original Code provided by the Initial Developer and including the name of the Initial Developer in

(a) the Source Code, and

(b) in any notice in an Executable version or related documentation in which You describe the origin or ownership of the Covered Code.

3.4. Intellectual Property Matters

a) Third Party Claims. If Contributor has knowledge that a license under a third party's intellectual property rights is required to exercise the rights granted by such Contributor under Sections 2.1 or 2.2, Contributor must include a text file with the Source Code distribution titled "LEGAL" which describes the claim and the party making the claim in sufficient detail that a recipient will know whom to contact. If Contributor obtains such knowledge after the Modification is made available as described in Section 3.2, Contributor shall promptly modify

the LEGAL file in all copies Contributor makes available thereafter and shall take other steps (such as notifying appropriate mailing lists or newsgroups) reasonably calculated to inform those who received the Covered Code that new knowledge has been obtained.

(b) Contributor APIs. If Contributor's Modifications include an application programming interface and Contributor has knowledge of patent licenses which are reasonably necessary to implement that API, Contributor must also include this information in the LEGAL file.

(c) Representations. Contributor represents that, except as disclosed pursuant to Section 3.4(a) above, Contributor believes that Contributor's Modifications are Contributor's original creation(s) and/or Contributor has sufficient rights to grant the rights conveyed by this License.

3.5. Required Notices. You must duplicate the notice in Exhibit A in each file of the Source Code. If it is not possible to put such notice in a particular Source Code file due to its structure, then You must include such notice in a location (such as a relevant directory) where a user would be likely to look for such a notice. If You created one or more Modification(s) You may add your name as a Contributor to the notice described in Exhibit A. You must also duplicate this License in any documentation for the Source Code where You describe recipients' rights or ownership rights relating to Covered Code. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Code. However, You may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear than any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer.

3.6. Distribution of Executable Versions. You may distribute Covered Code in Executable form only if the requirements of Section 3.1-3.5 have been met for that Covered Code, and if You include a notice stating that the Source Code version of the Covered Code is available under the terms of this License, including a description of how and where You have fulfilled the obligations of Section 3.2. The notice must be conspicuously included in any notice in an Executable version, related documentation or collateral in which You describe recipients' rights relating to the Covered Code. You may distribute the Executable version of Covered Code or ownership rights under a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and hat

the license for the Executable version does not attempt to limit or alter the recipient's rights in the Source Code version from the rights set forth in this License. If You distribute the Executable version under a different license You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or any Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer.

3.7. Larger Works. You may create a Larger Work by combining Covered Code with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Code.

4. Inability to Comply Due to Statute or Regulation.

If it is impossible for You to comply with any of the terms of this License with respect to some or all of the Covered Code due to statute, judicial order, or regulation then You must:

(a) comply with the terms of this License to the maximum extent possible; and

(b) describe the limitations and the code they affect. Such description must be included in the LEGAL file described in Section 3.4 and must be included with all distributions of the Source Code. Except to the extent prohibited by statute or regulation, such description must be sufficiently detailed for a recipient of ordinary skill to be able to understand it.

5. Application of this License.

This License applies to code to which the Initial Developer has attached the notice in Exhibit A and to related Covered Code.

6. Versions of the License.

6.1. New Versions. Maycon Ferraça may publish revised and/or new versions of the License from time to time. Each version will be given a distinguishing version number.

6.2. Effect of New Versions. Once Covered Code has been published under a particular version of the License, You may always continue to use it under the terms of that version. You may also choose to use such Covered Code under the terms of any subsequent version of the License published by the Initial Developer. No one other than the Initial Developer has the right to modify the terms applicable to Covered Code created under this License.

6.3. Derivative Works. If You create or use a modified version of this License (which you may only do in order to apply it to code which is not already Covered Code governed by this License), You must

(a) rename Your license so that the phrases "Mozilla", "MOZILLAPL", "MOZPL", "Netscape", "MPL", "NPL", or any confusingly similar phrases do not appear in your license (except to note that your license differs from this License) and

(b) otherwise make it clear that Your version of the license contains terms which differ from the Mozilla Public License and Netscape Public License. (Filling in the name of the Initial Developer, Original Code or Contributor in the notice described in Exhibit A shall not of themselves be deemed to be modifications of this License.)

6.4 Origin of the Migrate Databases Public License. The Migrate Databases Public License is based on the Mozilla Public License V 1.1 with the following changes:

1) The license is published by the Initial Developer of this code. Only the Initial Developer can modify the terms applicable to Covered Code.

2) The license can be modified and used for code which is not already governed by this license. Modified versions of the license must be renamed to avoid confusion with the Migrate Databases Public License and must include a description of changes from the Migrate Databases Public License.

3) The name of the license in Exhibit A is the "Migrate Databases Public License".

4) The reference to an alternative license in Exhibit A has been removed .

5) Amendments I, II, III, V, and VI have been deleted.

6) Exhibit A, Netscape Public License has been deleted

7. DISCLAIMER OF WARRANTY.

COVERED CODE IS PROVIDED UNDER THIS LICENSE ON AN "AS IS" BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED CODE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED CODE IS WITH YOU. SHOULD ANY COVERED CODE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING,

REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED CODE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

8. TERMINATION.

8.1. This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. All sublicenses to the Covered Code which are properly granted shall survive any termination of this License. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.

8.2. If You initiate litigation by asserting a patent infringement claim (excluding declatory judgment actions) against Initial Developer or a Contributor (the Initial Developer or Contributor against whom You file such action is referred to as "Participant") alleging that:

(a) such Participant's Contributor Version directly or indirectly infringes any patent, then any and all rights granted by such Participant to You under Sections 2.1 and/or 2.2 of this License shall, upon 60 days notice from Participant terminate prospectively, unless if within 60 days after receipt of notice You either:

(i) agree in writing to pay Participant a mutually agreeable reasonable royalty for Your past and future use of Modifications made by such Participant, or

(ii) withdraw Your litigation claim with respect to the Contributor Version against such Participant.

If within 60 days of notice, a reasonable royalty and payment arrangement are not mutually agreed upon in writing by the parties or the litigation claim is not withdrawn, the rights granted by Participant to You under Sections 2.1 and/or 2.2 automatically terminate at the expiration of the 60 day notice period specified above.

(b) any software, hardware, or device, other than such Participant's Contributor Version, directly or indirectly infringes any patent, then any rights granted to You by such Participant under Sections 2.1(b) and 2.2(b) are revoked effective as of the date You first made, used, sold, distributed, or had made, Modifications made by that Participant.

8.3. If You assert a patent infringement claim against Participant alleging that such Participant's Contributor Version directly or indirectly infringes any patent where such claim is resolved (such as by license or settlement) prior to the initiation of patent infringement

litigation, then the reasonable value of the licenses granted by such Participant under Sections 2.1 or 2.2 shall be taken into account in determining the amount or value of any payment or license.

8.4. In the event of termination under Sections 8.1 or 8.2 above, all end user license agreements (excluding distributors and resellers) which have been validly granted by You or any distributor hereunder prior to termination shall survive termination.

9. LIMITATION OF LIABILITY.

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL YOU, THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED CODE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO ANY PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTY'S NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

10. U.S. GOVERNMENT END USERS.

The Covered Code is a "commercial item," as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of "commercial computer software" and "commercial computer software documentation," as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Code with only those rights set forth herein.

11. MISCELLANEOUS.

This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License shall be governed by California law

provisions (except to the extent applicable law, if any, provides otherwise), excluding its conflict-of-law provisions. With respect to disputes in which at least one party is a citizen of, or an entity chartered or registered to do business in the United States of America, any litigation relating to this License shall be subject to the jurisdiction of the Federal Courts of the Northern District of California, with venue lying in Santa Clara County, California, with the losing party responsible for costs, including without limitation, court costs and reasonable attorneys' fees and expenses. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License.

12. RESPONSIBILITY FOR CLAIMS.

As between Initial Developer and the Contributors, each party is responsible for claims and damages arising, directly or indirectly, out of its utilization of rights under this License and You agree to work with Initial Developer and Contributors to distribute such responsibility on an equitable basis. Nothing herein is intended or shall be deemed to constitute any admission of liability.

13. MULTIPLE-LICENSED CODE.

Initial Developer may designate portions of the Covered Code as "Multiple-Licensed". "Multiple-Licensed" means that the Initial Devpoeloper permits you to utilize portions of the Covered Code under Your choice of the IDPL or the alternative licenses, if any, specified by the Initial Developer in the file described in Exhibit A.

EXHIBIT A - Migrate Databases Public License.

The contents of this file are subject to the Migrate Databases Public License Version 1.0 (the "License"); you may not use this file except in compliance with the License.

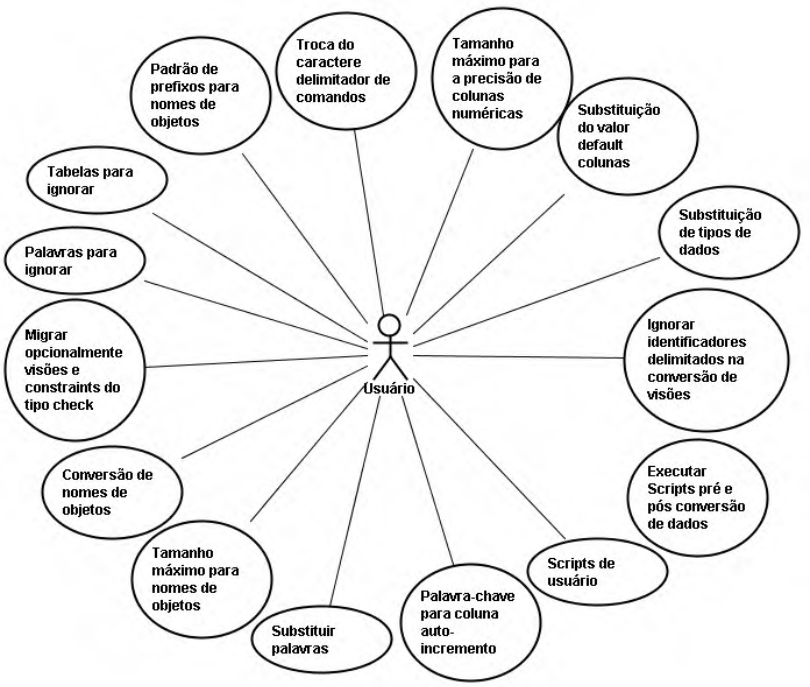
Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code was created by _____.

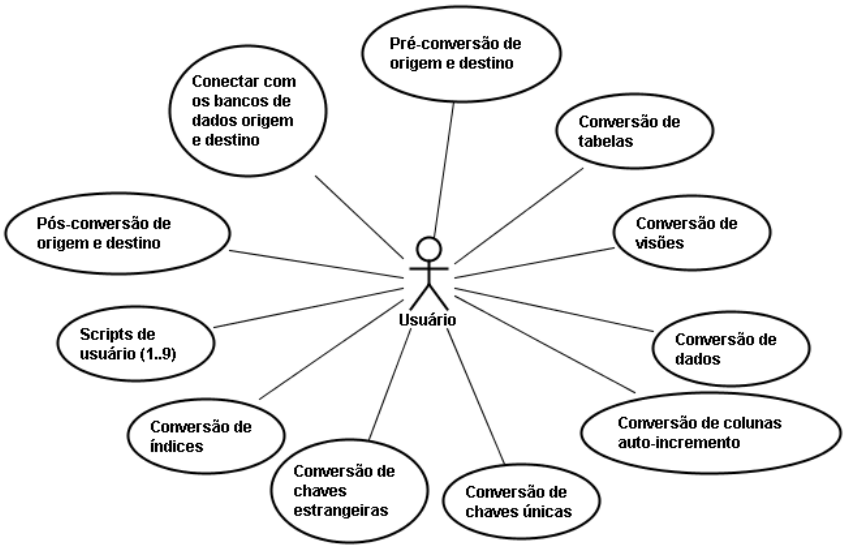
All Rights Reserved.

Contributor(s): _____.

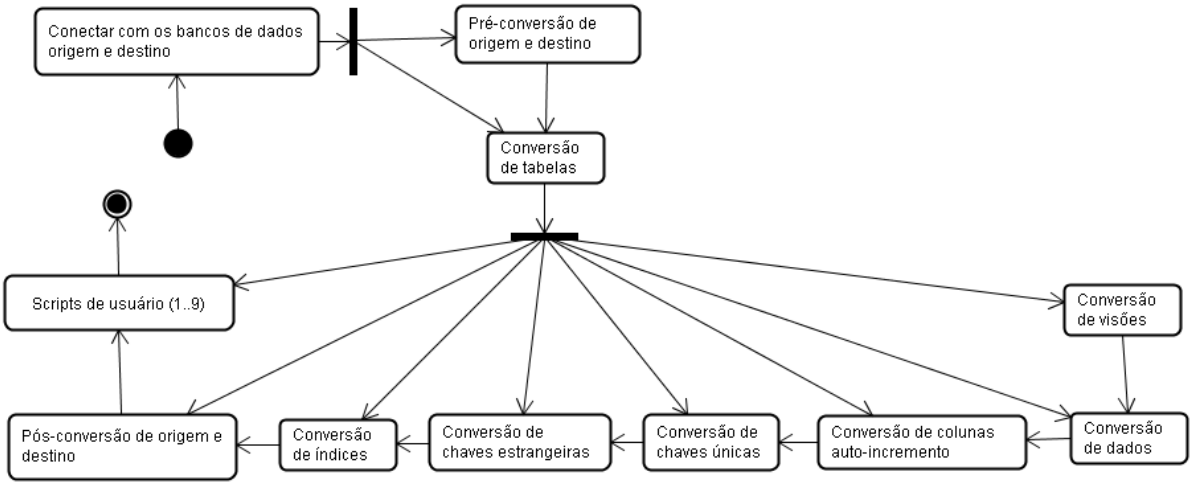
APÊNDICE B – DIAGRAMAS FINAIS



Requisitos para a configuração do sistema.



Requisitos para a utilização do sistema.



Fluxo de atividades do sistema.

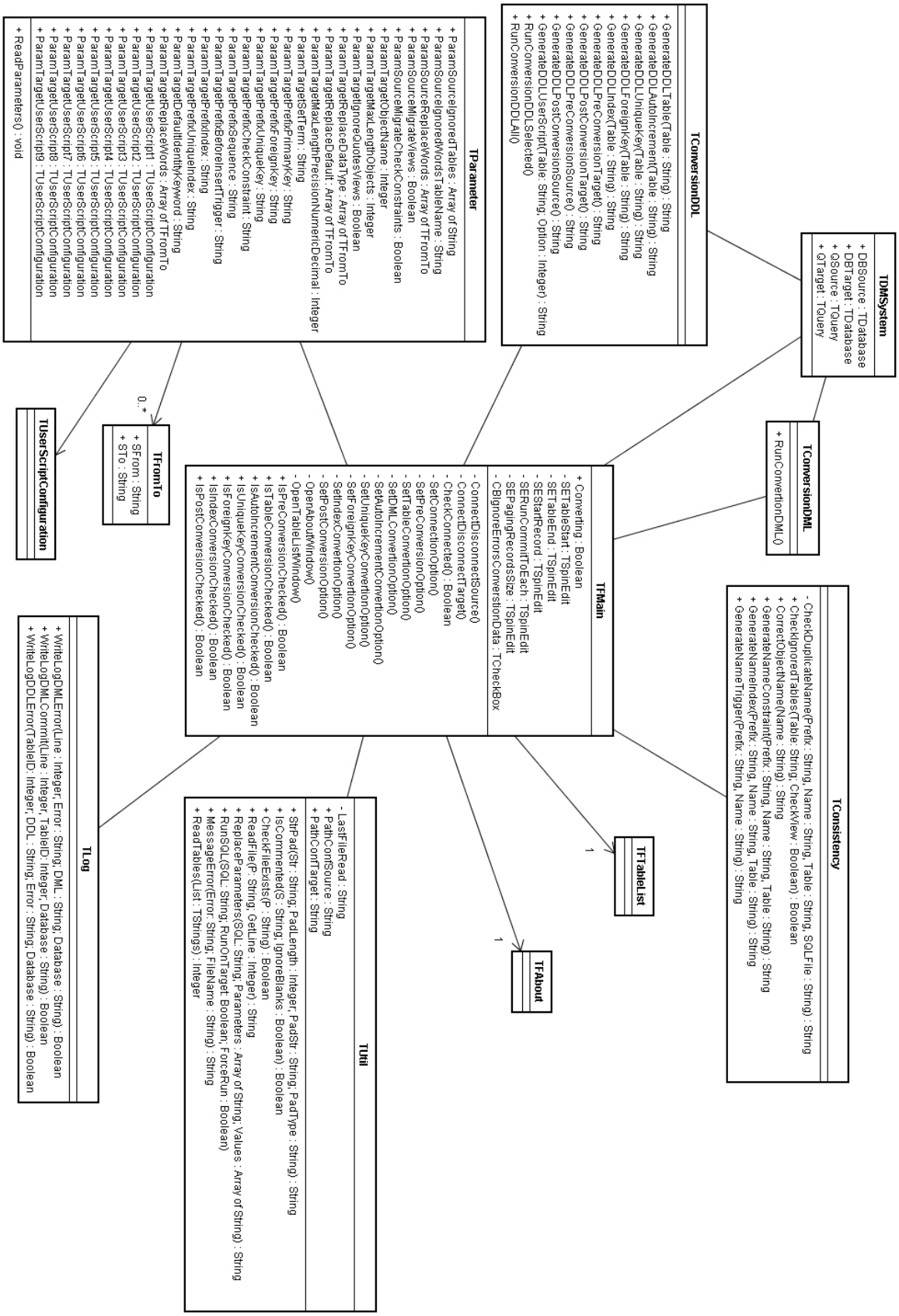


Diagrama de modelo conceitual do sistema.

APÊNDICE C – DISPOSIÇÃO DOS ARQUIVOS DO SISTEMA

```
|   license.rtf
|   Migrate.exe
|
+---BDE
|       *
|
\---Configuration
    |
    +---Source
    |   +---Firebird
    |   |       default.ini
    |   |       sql_definition_autoincrement.conf
    |   |       sql_definition_check.conf
    |   |       sql_definition_foreign_key.conf
    |   |       sql_definition_index.conf
    |   |       sql_definition_table.conf
    |   |       sql_definition_unique_key.conf
    |   |       sql_exists_view.conf
    |   |       sql_format_column_date.conf
    |   |       sql_format_column_time.conf
    |   |       sql_format_column_timestamp.conf
    |   |       sql_retrieve_data.conf
    |   |       sql_retrieve_data_2.conf
    |   |       sql_script_post-conversion.conf
    |   |       sql_script_pre-conversion.conf
    |   |       sql_table_columns.conf
    |   |       sql_table_count.conf
    |   |       sql_table_list.conf
    |   |       sql_temp_id_create.conf
    |   |       sql_temp_id_drop.conf
    |   |
    |   +---Microsoft SQL Server
    |   |       default.ini
    |   |       sql_definition_autoincrement.conf
    |   |       sql_definition_check.conf
    |   |       sql_definition_foreign_key.conf
    |   |       sql_definition_index.conf
    |   |       sql_definition_table.conf
    |   |       sql_definition_unique_key.conf
    |   |       sql_exists_view.conf
    |   |       sql_format_column_date.conf
    |   |       sql_format_column_time.conf
    |   |       sql_format_column_timestamp.conf
    |   |       sql_retrieve_data.conf
    |   |       sql_retrieve_data_2.conf
    |   |       sql_script_post-conversion.conf
    |   |       sql_script_pre-conversion.conf
    |   |       sql_table_columns.conf
    |   |       sql_table_count.conf
    |   |       sql_table_list.conf
    |   |       sql_temp_id_create.conf
    |   |       sql_temp_id_drop.conf
    |   |
    |   +---PostgreSQL
    |   |       default.ini
    |   |       sql_definition_autoincrement.conf
    |   |       sql_definition_check.conf
    |   |       sql_definition_foreign_key.conf
    |   |       sql_definition_index.conf
    |   |       sql_definition_table.conf
    |   |       sql_definition_unique_key.conf
    |   |       sql_exists_view.conf
    |   |       sql_format_column_date.conf
    |   |       sql_format_column_time.conf
    |   |       sql_format_column_timestamp.conf
    |   |       sql_retrieve_data.conf
    |   |       sql_retrieve_data_2.conf
    |   |       sql_script_post-conversion.conf
    |   |       sql_script_pre-conversion.conf
    |   |       sql_table_columns.conf
    |   |       sql_table_count.conf
    |   |       sql_table_list.conf
    |   |       sql_temp_id_create.conf
    |   |       sql_temp_id_drop.conf
    |   |
    |   \---Sybase Adaptive Server Anywhere
    |           default.ini
    |           sql_definition_autoincrement.conf
    |           sql_definition_check.conf
    |           sql_definition_foreign_key.conf
    |           sql_definition_foreign_key_2.conf
    |           sql_definition_index.conf
```

```
|           sql_definition_table.conf
|           sql_definition_unique_key.conf
|           sql_exists_view.conf
|           sql_format_column_date.conf
|           sql_format_column_time.conf
|           sql_format_column_timestamp.conf
|           sql_retrieve_data.conf
|           sql_retrieve_data_2.conf
|           sql_script_post-conversion.conf
|           sql_script_pre-conversion.conf
|           sql_table_columns.conf
|           sql_table_count.conf
|           sql_table_list.conf
|           sql_temp_id_create.conf
|           sql_temp_id_drop.conf
|
\---Target
    +---Firebird
    |       default.ini
    |       sql_create_autoincrement.conf
    |       sql_create_foreign_key.conf
    |       sql_create_index.conf
    |       sql_create_table.conf
    |       sql_create_unique_key.conf
    |       sql_exists_index.conf
    |       sql_exists_index_2.conf
    |       sql_exists_table.conf
    |       sql_find_constraints.conf
    |       sql_find_indices.conf
    |       sql_find_sequences.conf
    |       sql_find_triggers.conf
    |       sql_insert_table.conf
    |       sql_script_post-conversion.conf
    |       sql_script_pre-conversion.conf
    |
    +---Microsoft SQL Server
    |       default.ini
    |       sql_create_autoincrement.conf
    |       sql_create_foreign_key.conf
    |       sql_create_index.conf
    |       sql_create_table.conf
    |       sql_create_unique_key.conf
    |       sql_exists_index.conf
    |       sql_exists_index_2.conf
    |       sql_exists_table.conf
    |       sql_find_constraints.conf
    |       sql_find_indices.conf
    |       sql_find_sequences.conf
    |       sql_find_triggers.conf
    |       sql_insert_table.conf
    |       sql_script_post-conversion.conf
    |       sql_script_pre-conversion.conf
    |       sql_user_script_#1.conf
    |       sql_user_script_#2.conf
    |
    +---PostgreSQL
    |       default.ini
    |       sql_create_autoincrement.conf
    |       sql_create_foreign_key.conf
    |       sql_create_index.conf
    |       sql_create_table.conf
    |       sql_create_unique_key.conf
    |       sql_exists_index.conf
    |       sql_exists_index_2.conf
    |       sql_exists_table.conf
    |       sql_find_constraints.conf
    |       sql_find_indices.conf
    |       sql_find_sequences.conf
    |       sql_find_triggers.conf
    |       sql_insert_table.conf
    |       sql_script_post-conversion.conf
    |       sql_script_pre-conversion.conf
    |
    \---Sybase Adaptive Server Anywhere
            default.ini
            sql_create_autoincrement.conf
            sql_create_foreign_key.conf
            sql_create_index.conf
            sql_create_table.conf
            sql_create_unique_key.conf
            sql_exists_index.conf
            sql_exists_index_2.conf
            sql_exists_table.conf
            sql_find_constraints.conf
            sql_find_indices.conf
            sql_find_sequences.conf
            sql_find_triggers.conf
            sql_insert_table.conf
            sql_script_post-conversion.conf
            sql_script_pre-conversion.conf
```

APÊNDICE D – TEMPLATES E CONFIGURAÇÕES DO SISTEMA PARA OS BANCOS *FIREBIRD*, *MICROSOFT SQL SERVER*, *POSTGRESQL*, *SYBASE ADAPTIVE SERVER ANYWHERE*

default.ini | **Templates de origem (*Source*)** | **Banco de dados *Firebird***

```
[Parameter]
;;DBMS: Firebird 2.1.x

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; A list of tables to be ignored by the system, separated by
;; semicolon. All the tables starting by the words will be
;; ignored
;;
;; Default: ""
;; Type: String
;; Usage example:
;; "Table01;Table02;"
Ignored Tables=DUAL;RDB$;MON$

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Ignored Words
;;
;; Type: String
;; Usage example: "DBA.;::numeric"
Ignored Words=

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Replace Words. Replace Words in the SQL stataments.
;;
;; Type: String
;; Usage example: "TRIM(:dbo.TRIM(;"
Replace Words=

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Migrate Views
;;
;; Default: 1
;; Type: Boolean (1=True/0=False)
Migrate Views=1

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Migrate Check Constraints
;;
;; Default: 1
;; Type: Boolean (1=True/0=False)
Migrate Check Constraints=1
```

sql_definition_autoincrement.conf | **Templates de origem (*Source*)** | **Banco de dados *Firebird***

```
SELECT
s.column_name

FROM (SELECT
      SUBSTRING(S.RDB$DESCRIPTION FROM (POSITION('.', S.RDB$DESCRIPTION) + 1) FOR
CHAR_LENGTH(S.RDB$DESCRIPTION)) AS column_name,
      SUBSTRING(S.RDB$DESCRIPTION FROM 1 FOR IIF((POSITION('.', S.RDB$DESCRIPTION) - 1) < 0, 0,
(POSITION('.', S.RDB$DESCRIPTION) - 1))) AS table_name

      FROM
      RDB$GENERATORS S) S

WHERE
s.table_name = '<table_name>'

ORDER BY
s.column_name;
```

sql_definition_check.conf | **Templates de origem (*Source*)** | **Banco de dados *Firebird***

```
SELECT
CHK_K.RDB$CONSTRAINT_NAME AS check_name,
TRG.RDB$TRIGGER_SOURCE AS check_rule

FROM
RDB$TRIGGERS TRG
```

```
JOIN RDB$CHECK_CONSTRAINTS CHK_K ON
TRG.RDB$TRIGGER_NAME = CHK_K.RDB$TRIGGER_NAME

WHERE
TRG.RDB$RELATION_NAME = '<table_name>' AND
TRG.RDB$TRIGGER_TYPE = 1 AND
EXISTS (SELECT RDB$CONSTRAINT_NAME FROM RDB$RELATION_CONSTRAINTS REL_K
        WHERE CHK_K.RDB$CONSTRAINT_NAME = REL_K.RDB$CONSTRAINT_NAME);
```

sql_definition_foreign_key.conf **Templates de origem (Source)** **Banco de dados Firebird**

```
SELECT
RC2.RDB$RELATION_NAME AS table_name_dest,
RC.RDB$CONSTRAINT_NAME AS role,
TRIM(IIF(REFC.RDB$UPDATE_RULE = 'RESTRICT', '', 'ON UPDATE ' || TRIM(REFC.RDB$UPDATE_RULE)) ||
     IIF(REFC.RDB$DELETE_RULE = 'RESTRICT', '', 'ON DELETE ' || TRIM(REFC.RDB$DELETE_RULE))) AS actions,
TRIM((SELECT
      TRIM(LIST(' ' || TRIM(XS2.RDB$FIELD_NAME)))
      FROM (SELECT * FROM RDB$INDEX_SEGMENTS XS2 ORDER BY XS2.RDB$FIELD_POSITION) XS2
      WHERE XS2.RDB$INDEX_NAME = RC.RDB$INDEX_NAME)) AS columns_origin,
TRIM((SELECT
      TRIM(LIST(' ' || TRIM(XS.RDB$FIELD_NAME)))
      FROM (SELECT * FROM RDB$INDEX_SEGMENTS XS ORDER BY XS.RDB$FIELD_POSITION) XS
      WHERE XS.RDB$INDEX_NAME = RC2.RDB$INDEX_NAME)) AS columns_destination

FROM
RDB$RELATION_CONSTRAINTS RC

JOIN RDB$REF_CONSTRAINTS REFC ON
REFC.RDB$CONSTRAINT_NAME = RC.RDB$CONSTRAINT_NAME

JOIN RDB$RELATION_CONSTRAINTS RC2 ON
REFC.RDB$CONST_NAME_UQ = RC2.RDB$CONSTRAINT_NAME

WHERE
RC.RDB$CONSTRAINT_TYPE = 'FOREIGN KEY' AND
RC.RDB$RELATION_NAME = '<table_name>'

ORDER BY
RC.RDB$CONSTRAINT_NAME;
```

sql_definition_index.conf **Templates de origem (*Source*)** **Banco de dados *Firebird***

```
SELECT
IX.RDB$INDEX_NAME AS index_name,
IIF(IX.RDB$UNIQUE_FLAG = 1, 'U', 'N') AS index_unique,
IIF(IX.RDB$INDEX_TYPE = 1, 'DESC', 'ASC') AS index_order,
TRIM((SELECT
      TRIM(LIST(' ' || TRIM(ISG.RDB$FIELD_NAME)))
      FROM (SELECT * FROM RDB$INDEX_SEGMENTS I ORDER BY I.RDB$FIELD_POSITION) ISG
      WHERE ISG.RDB$INDEX_NAME = IX.RDB$INDEX_NAME)) AS index_columns,

TRIM((SELECT
      TRIM(LIST(' ' || TRIM(ISG.RDB$FIELD_NAME) || ' ' || IIF(IX.RDB$INDEX_TYPE = 1, 'DESC', 'ASC')))
      FROM (SELECT * FROM RDB$INDEX_SEGMENTS I ORDER BY I.RDB$FIELD_POSITION) ISG
      WHERE ISG.RDB$INDEX_NAME = IX.RDB$INDEX_NAME)) AS index_columns_order

FROM
RDB$INDICES IX

JOIN RDB$RELATIONS R ON
IX.RDB$RELATION_NAME = R.RDB$RELATION_NAME

WHERE
R.RDB$RELATION_NAME = '<table_name>' AND
NOT EXISTS (SELECT * FROM RDB$RELATION_CONSTRAINTS REL_CON
            WHERE REL_CON.RDB$INDEX_NAME = IX.RDB$INDEX_NAME)

ORDER BY
IIF(IX.RDB$UNIQUE_FLAG = 1, 0, 1),
IX.RDB$INDEX_NAME;
```

sql_definition_table.conf **Templates de origem (*Source*)** **Banco de dados *Firebird***

```
SELECT
'CREATE VIEW <table_name> AS ' || R.RDB$VIEW_SOURCE || ';' AS view_def,

COALESCE((SELECT 'Y'
FROM RDB$RELATION_CONSTRAINTS RC
LEFT JOIN RDB$INDEX_SEGMENTS I ON
I.RDB$INDEX_NAME = RC.RDB$INDEX_NAME
WHERE RC.RDB$RELATION_NAME = F.RDB$RELATION_NAME AND
```

```
I.RDB$FIELD_NAME = F.RDB$FIELD_NAME AND
RC.RDB$CONSTRAINT_TYPE = 'PRIMARY KEY'), 'N') AS PKEY,
IIF(F.RDB$NULL_FLAG IS NULL, 'Y', 'N') AS nulls,

COALESCE(CASE
  WHEN FS.RDB$FIELD_PRECISION > 0 THEN FS.RDB$FIELD_PRECISION
  WHEN T.RDB$TYPE_NAME IN ('VARYING', 'TEXT') THEN FS.RDB$CHARACTER_LENGTH
END, 0) AS WIDTH,
COALESCE(IIF(FS.RDB$FIELD_PRECISION > 0, FS.RDB$FIELD_SCALE * -1, NULL), 0) AS scale,

F.RDB$FIELD_NAME AS column_name,
CAST(IIF(RIGHT(RIGHT(F.RDB$DEFAULT_SOURCE, CHAR_LENGTH(F.RDB$DEFAULT_SOURCE) - 8), 1) = ',',
LEFT(RIGHT(F.RDB$DEFAULT_SOURCE, CHAR_LENGTH(F.RDB$DEFAULT_SOURCE) - 8),
CHAR_LENGTH(RIGHT(F.RDB$DEFAULT_SOURCE, CHAR_LENGTH(F.RDB$DEFAULT_SOURCE) - 8)) - 1),
RIGHT(F.RDB$DEFAULT_SOURCE, CHAR_LENGTH(F.RDB$DEFAULT_SOURCE) - 8 )) AS VARCHAR(254)) AS default_column,

NULL AS check_column,
TRIM(CASE
       WHEN FS.RDB$FIELD_PRECISION > 0  THEN 'NUMERIC'
       WHEN T.RDB$TYPE_NAME = 'FLOAT'   THEN 'REAL'
       WHEN T.RDB$TYPE_NAME = 'LONG'    THEN 'INTEGER'
       WHEN T.RDB$TYPE_NAME = 'SHORT'   THEN 'SMALLINT'
       WHEN T.RDB$TYPE_NAME = 'VARYING' THEN 'VARCHAR'
       WHEN T.RDB$TYPE_NAME = 'TEXT'    THEN 'CHAR'
       WHEN T.RDB$TYPE_NAME = 'INT64'   THEN 'BIGINT'
       WHEN T.RDB$TYPE_NAME = 'DOUBLE'  THEN 'DOUBLE PRECISION'
       WHEN FS.RDB$FIELD_SUB_TYPE = 1 AND T.RDB$TYPE_NAME = 'BLOB' THEN 'TEXT'
       ELSE
         T.RDB$TYPE_NAME
     END) AS data_type,

IIF(EXISTS (SELECT 1 FROM (SELECT SUBSTRING(S.RDB$DESCRIPTION FROM (POSITION('.', S.RDB$DESCRIPTION) + 1) FOR
CHAR_LENGTH(S.RDB$DESCRIPTION)) AS column_name,
                                  SUBSTRING(S.RDB$DESCRIPTION FROM 1 FOR IIF((POSITION('.',
S.RDB$DESCRIPTION) - 1) < 0, 0, (POSITION('.', S.RDB$DESCRIPTION) - 1))) AS table_name
                           FROM  RDB$GENERATORS S) S
                           WHERE s.table_name = F.RDB$RELATION_NAME AND
                                 s.column_name = F.RDB$FIELD_NAME), 'Y', 'N') AS is_autoincrement,

IIF(R.RDB$VIEW_SOURCE IS NOT NULL, 'VIEW', 'TABLE') AS table_type

FROM
RDB$RELATIONS R

JOIN RDB$RELATION_FIELDS F ON
R.RDB$RELATION_NAME = F.RDB$RELATION_NAME

LEFT JOIN RDB$FIELDS FS ON
FS.RDB$FIELD_NAME = F.RDB$FIELD_SOURCE

LEFT JOIN RDB$TYPES T ON
T.RDB$FIELD_NAME = 'RDB$FIELD_TYPE' AND
FS.RDB$FIELD_TYPE = T.RDB$TYPE

WHERE
F.RDB$RELATION_NAME = '<table_name>'

ORDER BY
F.RDB$FIELD_POSITION;
```

sql_definition_unique_key.conf **Templates de origem (*Source*)** **Banco de dados *Firebird***

```
SELECT
REL_CON.RDB$CONSTRAINT_NAME AS constraint_name,
TRIM((SELECT
      TRIM(LIST(' ' || TRIM(ISG.RDB$FIELD_NAME)))
      FROM (SELECT * FROM RDB$INDEX_SEGMENTS I ORDER BY I.RDB$FIELD_POSITION) ISG
      WHERE ISG.RDB$INDEX_NAME = IX.RDB$INDEX_NAME)) AS constraint_columns,
IIF(IX.RDB$INDEX_TYPE = 1, 'DESC', 'ASC') AS index_order

FROM
RDB$RELATION_CONSTRAINTS REL_CON

JOIN RDB$INDICES IX ON
REL_CON.RDB$INDEX_NAME = IX.RDB$INDEX_NAME

WHERE
REL_CON.RDB$CONSTRAINT_TYPE = 'UNIQUE' AND
REL_CON.RDB$RELATION_NAME = '<table_name>'

ORDER BY
REL_CON.RDB$CONSTRAINT_NAME;
```

sql_exists_view.conf **Templates de origem (*Source*)** **Banco de dados *Firebird***

```
SELECT
COUNT(*) AS view_exists

FROM
RDB$RELATIONS

WHERE
RDB$RELATION_NAME = '<view_name>' AND
RDB$VIEW_SOURCE IS NOT NULL;
```

sql_format_column_date.conf **Templates de origem (*Source*)** **Banco de dados *Firebird***

```
<column_name> || '' AS <column_name>
```

sql_format_column_time.conf **Templates de origem (*Source*)** **Banco de dados *Firebird***

```
LEFT(<column_name> || '', CHAR_LENGTH(<column_name> || '') - 1) AS <column_name>
```

sql_format_column_timestamp.conf **Templates de origem (*Source*)** **Banco de dados *Firebird***

```
LEFT(<column_name> || '', CHAR_LENGTH(<column_name> || '') - 1) AS <column_name>
```

sql_retrieve_data.conf **Templates de origem (*Source*)** **Banco de dados *Firebird***

```
SELECT <columns_list> FROM <table_name>
```

sql_retrieve_data_2.conf **Templates de origem (*Source*)** **Banco de dados *Firebird***

```
WHERE temp_id BETWEEN <start> AND <end>
```

sql_script_post-conversion.conf **Templates de origem (*Source*)** **Banco de dados *Firebird***

sql_script_pre-conversion.conf **Templates de origem (*Source*)** **Banco de dados *Firebird***

sql_table_columns.conf **Templates de origem (*Source*)** **Banco de dados *Firebird***

```
SELECT DISTINCT
F.RDB$FIELD_NAME AS column_name,
TRIM(CASE
       WHEN FS.RDB$FIELD_PRECISION > 0  THEN 'NUMERIC'
       WHEN T.RDB$TYPE_NAME = 'FLOAT'   THEN 'REAL'
       WHEN T.RDB$TYPE_NAME = 'LONG'    THEN 'INTEGER'
       WHEN T.RDB$TYPE_NAME = 'SHORT'   THEN 'SMALLINT'
       WHEN T.RDB$TYPE_NAME = 'VARYING' THEN 'VARCHAR'
       WHEN T.RDB$TYPE_NAME = 'TEXT'    THEN 'CHAR'
       WHEN T.RDB$TYPE_NAME = 'INT64'   THEN 'BIGINT'
       WHEN T.RDB$TYPE_NAME = 'DOUBLE'  THEN 'DOUBLE PRECISION'
       WHEN FS.RDB$FIELD_SUB_TYPE = 1 AND T.RDB$TYPE_NAME = 'BLOB' THEN 'TEXT'
       ELSE
         T.RDB$TYPE_NAME
     END) AS data_type

FROM
RDB$RELATION_FIELDS F

LEFT JOIN RDB$FIELDS FS ON
FS.RDB$FIELD_NAME = F.RDB$FIELD_SOURCE

LEFT JOIN RDB$TYPES T ON
T.RDB$FIELD_NAME = 'RDB$FIELD_TYPE' AND
FS.RDB$FIELD_TYPE = T.RDB$TYPE

WHERE
F.RDB$RELATION_NAME = '<table_name>'

ORDER BY
F.RDB$FIELD_POSITION;
```

sql_table_count.conf **Templates de origem (*Source*)** **Banco de dados *Firebird***

```
SELECT COUNT(*) AS total FROM <table_name>;
```

sql_table_list.conf **Templates de origem (*Source*)** **Banco de dados *Firebird***

```
SELECT
R.RDB$RELATION_NAME AS table_name

FROM
RDB$RELATIONS R
```

```
ORDER BY
IIF(R.RDB$VIEW_SOURCE IS NULL, 0, 1),
table_name;
```

sql_temp_id_create.conf **Templates de origem (*Source*)** **Banco de dados *Firebird***

```
--The column must be called id_converter, but the index can have any name
ALTER TABLE <table_name> ADD temp_id BIGINT;
CREATE INDEX index_temp_id ON <table_name> (temp_id);
COMMIT;

SET TERM ^;
EXECUTE BLOCK AS
  DECLARE i BIGINT;
  DECLARE aux BIGINT;
BEGIN
  i = 0;
  FOR SELECT temp_id FROM <table_name> INTO :aux AS CURSOR loop DO
  BEGIN
    i = i + 1;
    UPDATE <table_name> SET temp_id = :i WHERE CURRENT OF loop;
  END
END^
SET TERM ;^
```

sql_temp_id_drop.conf **Templates de origem (*Source*)** **Banco de dados *Firebird***

```
--The column must be called id_converter, but the index can have any name
DROP INDEX index_temp_id;
ALTER TABLE <table_name> DROP temp_id;
```

default.ini **Templates de origem (*Source*)** **Banco de dados *Microsoft SQL Server***

```
[Parameter]
;;DBMS: Microsoft SQL Server 2005

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; A list of tables to be ignored by the system, separated by
;; semicolon. All the tables starting by the words will be
;; ignored
;;
;; Default: ""
;; Type: String
;; Usage example:
;; "Table01;Table02;"
Ignored Tables=sys;DUAL;INFORMATION_SCHEMA.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Ignored Words
;;
;; Type: String
;; Usage example: "DBA.;::numeric"
Ignored Words=dbo.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Replace Words. Replace Words in the SQL stataments.
;;
;; Type: String
;; Usage example: "TRIM(:dbo.TRIM(;"
Replace Words=

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Migrate Views
;;
;; Default: 1
;; Type: Boolean (1=True/0=False)
Migrate Views=1

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Migrate Check Constraints
;;
;; Default: 1
;; Type: Boolean (1=True/0=False)
Migrate Check Constraints=1
```

sql_definition_autoincrement.conf **Templates de origem (*Source*)** **Banco de dados *Microsoft SQL Server***

```
SELECT
CAST(column_name AS TEXT) AS column_name

FROM
INFORMATION_SCHEMA.COLUMNS
```

```
WHERE
table_name = '<table_name>' AND
COLUMNPROPERTY(OBJECT_ID(table_name), column_name, 'IsIdentity') = 1

ORDER BY
ordinal_position;
```

sql_definition_check.conf **Templates de origem (*Source*)** **Banco de dados *Microsoft SQL Server***

```
SELECT
CAST(sys1.name AS TEXT) AS check_name,
CAST('CHECK ' + REPLACE(REPLACE(REPLACE(c.text, '[', ''), ']', ''), 'dbo.', '') AS TEXT) AS check_rule

FROM
sysobjects sys1

JOIN sysobjects sys2 ON
sys1.parent_obj = sys2.id

JOIN syscomments c ON
sys1.id = c.id

WHERE
sys1.xtype = 'C' AND
sys2.name = '<table_name>';
```

sql_definition_foreign_key.conf **Templates de origem (*Source*)** **Banco de dados *Microsoft SQL Server***

```
SELECT
CAST(f.table_name_dest AS TEXT) AS table_name_dest,
CAST(f.role AS TEXT) AS role,
f.actions,
CAST(SUBSTRING(f.columns_origin, 1, LEN(f.columns_origin) - 2) AS TEXT) AS columns_origin,
CAST(SUBSTRING(f.columns_destination, 1, LEN(f.columns_destination) - 2) AS TEXT) AS columns_destination

FROM
(SELECT DISTINCT
 ccu.table_name AS table_name_dest,
 tc.constraint_name AS role,
LTRIM(RTRIM((CASE WHEN rc.update_rule = 'NO ACTION' THEN '' ELSE 'ON UPDATE ' + rc.update_rule END) + ' ' +
            (CASE WHEN rc.delete_rule = 'NO ACTION' THEN '' ELSE 'ON DELETE ' + rc.delete_rule END))) AS
actions,

 (SELECT kcu.column_name + ', '
  FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE kcu
  WHERE kcu.constraint_catalog = tc.constraint_catalog AND
        kcu.constraint_schema = tc.constraint_schema AND
        kcu.constraint_name = tc.constraint_name
  ORDER BY kcu.ordinal_position
  FOR XML PATH('')) AS columns_origin,

 (SELECT kcu.column_name + ', '
  FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE kcu
  WHERE kcu.constraint_schema = ccu.constraint_schema AND
        kcu.constraint_name = ccu.constraint_name
  ORDER BY kcu.ordinal_position
  FOR XML PATH('')) AS columns_destination

 FROM
 INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc

 LEFT JOIN INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc ON
 tc.constraint_catalog = rc.constraint_catalog AND
 tc.constraint_schema = rc.constraint_schema AND
 tc.constraint_name = rc.constraint_name

 LEFT JOIN INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE ccu ON
 ccu.constraint_catalog = rc.unique_constraint_catalog AND
 ccu.constraint_schema = rc.unique_constraint_schema AND
 ccu.constraint_name = rc.unique_constraint_name

 WHERE
 tc.constraint_type = 'FOREIGN KEY' AND
 tc.table_name = '<table_name>') f

ORDER BY
f.role;
```

sql_definition_index.conf **Templates de origem (*Source*)** **Banco de dados *Microsoft SQL Server***

```
SELECT
CAST(i.index_name AS TEXT) AS index_name,
i.index_unique,
```

```
CAST(SUBSTRING(i.index_columns, 1, LEN(i.index_columns) - 2) AS TEXT) AS index_columns,
i.index_order,
CAST(SUBSTRING(i.index_columns_order, 1, LEN(i.index_columns_order) - 2) AS TEXT) AS index_columns_order

FROM
(SELECT DISTINCT
 idxd.name AS index_name,
 (CASE WHEN idxd.is_unique = 1 THEN 'U' ELSE 'N' END) AS index_unique,

 (SELECT col2.name + ', '
  FROM sys.columns AS col2
  JOIN sys.index_columns AS idx2 ON
  idx2.object_id = col2.object_id AND
  idx2.column_id = col2.column_id
  WHERE idx2.index_id = idxd.index_id AND
        idx2.object_id = idxd.object_id
  ORDER BY idx2.key_ordinal
  FOR XML PATH('')) AS index_columns,

 (SELECT TOP 1 (CASE WHEN idx2.is_descending_key = 1 THEN 'DESC' ELSE 'ASC' END)
  FROM sys.index_columns AS idx2
  WHERE idx2.object_id = idxd.object_id AND
        idx2.index_id = idxd.index_id
  ORDER BY idx2.key_ordinal) AS index_order,

 (SELECT col2.name + ' ' + (CASE WHEN idx2.is_descending_key = 1 THEN 'DESC' ELSE 'ASC' END) +  ', '
  FROM sys.columns AS col2
  JOIN sys.index_columns AS idx2 ON
  idx2.object_id = col2.object_id AND
  idx2.column_id = col2.column_id
  WHERE idx2.index_id = idxd.index_id AND
        idx2.object_id = idxd.object_id
  ORDER BY idx2.key_ordinal
  FOR XML PATH('')) AS index_columns_order

 FROM
 sys.tables AS tab

 JOIN sys.columns AS col ON
 tab.object_id = col.object_id

 JOIN sys.index_columns AS idx ON
 idx.object_id = col.object_id AND
 idx.column_id = col.column_id

 JOIN sys.indexes AS idxd ON
 idxd.index_id = idx.index_id AND
 idxd.object_id = idx.object_id

 WHERE
 tab.name = '<table_name>' AND
 idxd.is_primary_key = 0) i

ORDER BY
i.index_unique DESC,
i.index_name;
```

sql_definition_table.conf — **Templates de origem (*Source*)** — **Banco de dados *Microsoft SQL Server***

```
SELECT

CAST((SELECT view_definition
      FROM INFORMATION_SCHEMA.VIEWS v
      WHERE v.table_catalog = t.table_catalog AND
            v.table_schema = t.table_schema AND
            v.table_name = t.table_name) AS TEXT) AS view_def,

COALESCE((SELECT CAST('Y' AS CHAR(1)) FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc
          JOIN INFORMATION_SCHEMA.KEY_COLUMN_USAGE kcu ON
               tc.constraint_catalog = kcu.constraint_catalog AND
               tc.constraint_schema = kcu.constraint_schema AND
               tc.constraint_name = kcu.constraint_name
          WHERE tc.constraint_type = 'PRIMARY KEY' AND
                tc.table_name = c.table_name AND
                kcu.column_name = c.column_name), 'N') AS pkey,

(CASE WHEN is_nullable = 'YES' THEN 'Y' ELSE 'N' END) AS nulls,
COALESCE(numeric_precision, character_maximum_length) AS width,
COALESCE(numeric_scale, 0) AS scale,

CAST(c.column_name AS VARCHAR(254)) AS column_name,

CAST(CASE
```

```
       WHEN c.column_default = '(getdate())' THEN 'CURRENT_TIMESTAMP'
       WHEN SUBSTRING(c.column_default, 1, 2) =  '(''' THEN SUBSTRING(c.column_default, 2,
LEN(c.column_default) - 2)
       WHEN SUBSTRING(c.column_default, 1, 2) =  '((' THEN SUBSTRING(c.column_default, 3,
LEN(c.column_default) - 4)
       ELSE column_default
     END AS TEXT) AS default_column,

NULL AS check_column,

CAST(CASE c.data_type
       WHEN 'datetime' THEN 'timestamp'
       WHEN 'float' THEN 'double precision'
       WHEN 'int' THEN 'integer'
       WHEN 'long varchar' THEN 'text'
       ELSE c.data_type
     END AS VARCHAR(254)) AS data_type,

(CASE WHEN COLUMNPROPERTY(OBJECT_ID(c.table_name), c.column_name, 'IsIdentity') = 1 THEN 'Y' ELSE 'N' END) AS
is_autoincrement,
(CASE WHEN t.table_type = 'VIEW' THEN 'VIEW' ELSE 'TABLE' END) AS table_type

FROM
INFORMATION_SCHEMA.TABLES t

JOIN INFORMATION_SCHEMA.COLUMNS c ON
c.table_catalog = t.table_catalog AND
c.table_schema = t.table_schema AND
c.table_name = t.table_name

WHERE
c.table_name = '<table_name>'

ORDER BY
c.ordinal_position;
```

sql_definition_unique_key.conf **Templates de origem (*Source*)** **Banco de dados *Microsoft SQL Server***

```
SELECT
CAST(c.constraint_name AS TEXT) AS constraint_name,
CAST(SUBSTRING(c.constraint_columns, 1, LEN(c.constraint_columns) - 2) AS TEXT) AS constraint_columns,
c.index_order

FROM
(SELECT
 tc.constraint_name,

 (SELECT kcu.column_name + ', '
  FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE kcu
  WHERE kcu.constraint_catalog = tc.constraint_catalog AND
        kcu.constraint_schema = tc.constraint_schema AND
        kcu.constraint_name = tc.constraint_name
  ORDER BY kcu.ordinal_position
  FOR XML PATH('')) AS constraint_columns,

 (SELECT TOP 1 (CASE WHEN idx.is_descending_key = 1 THEN 'DESC' ELSE 'ASC' END)
  FROM sys.index_columns AS idx
  JOIN sys.indexes AS idxd ON
      idxd.index_id = idx.index_id AND
      idxd.object_id = idx.object_id
  WHERE OBJECT_NAME(idx.object_id) = tc.table_name AND
        idxd.name = tc.constraint_name
  ORDER BY idx.key_ordinal) AS index_order

 FROM
 INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc

 WHERE
 tc.constraint_type = 'UNIQUE' AND
 tc.table_name = '<table_name>') c

ORDER BY
c.constraint_name;
```

sql_exists_view.conf **Templates de origem (*Source*)** **Banco de dados *Microsoft SQL Server***

```
SELECT
COUNT(*) AS view_exists

FROM
sys.views AS tab

WHERE
```

```
tab.name = '<view_name>';
```

sql_format_column_date.conf **Templates de origem (*Source*)** **Banco de dados *Microsoft SQL Server***

```
REPLACE(CONVERT(VARCHAR(10), <column_name>, 102), '.', '-') AS <column_name>
```

sql_format_column_time.conf **Templates de origem (*Source*)** **Banco de dados *Microsoft SQL Server***

```
SUBSTRING(CONVERT(VARCHAR(23), <column_name>, 121), 12, 23) AS <column_name>
```

sql_format_column_timestamp.conf **Templates de origem (*Source*)** **Banco de dados *Microsoft SQL Server***

```
CONVERT(VARCHAR(23), <column_name>, 121) AS <column_name>
```

sql_retrieve_data.conf **Templates de origem (*Source*)** **Banco de dados *Microsoft SQL Server***

```
SELECT <columns_list> FROM <table_name>
```

sql_retrieve_data_2.conf **Templates de origem (*Source*)** **Banco de dados *Microsoft SQL Server***

```
WHERE temp_id BETWEEN <start> AND <end>
```

sql_script_post-conversion.conf **Templates de origem (*Source*)** **Banco de dados *Microsoft SQL Server***

sql_script_pre-conversion.conf **Templates de origem (*Source*)** **Banco de dados *Microsoft SQL Server***

sql_table_columns.conf **Templates de origem (*Source*)** **Banco de dados *Microsoft SQL Server***

```
SELECT
CAST(c.column_name AS TEXT) AS column_name,

CAST(CASE c.data_type
       WHEN 'datetime' THEN 'timestamp'
       WHEN 'float' THEN 'double precision'
       WHEN 'int' THEN 'integer'
       WHEN 'long varchar' THEN 'text'
       ELSE c.data_type
     END AS TEXT) AS data_type

FROM
INFORMATION_SCHEMA.TABLES t

JOIN INFORMATION_SCHEMA.COLUMNS c ON
t.table_name = c.table_name

WHERE
t.table_name = '<table_name>'

ORDER BY
c.ordinal_position;
```

sql_table_count.conf **Templates de origem (*Source*)** **Banco de dados *Microsoft SQL Server***

```
SELECT COUNT(*) AS total FROM <table_name>;
```

sql_table_list.conf **Templates de origem (*Source*)** **Banco de dados *Microsoft SQL Server***

```
SELECT
CAST(t.table_name AS TEXT) AS table_name

FROM
INFORMATION_SCHEMA.TABLES t

ORDER BY
(CASE WHEN t.table_type = 'VIEW' THEN 1 ELSE 0 END),
t.table_name;
```

sql_temp_id_create.conf **Templates de origem (*Source*)** **Banco de dados *Microsoft SQL Server***

```
ALTER TABLE <table_name> ADD temp_id BIGINT;
CREATE INDEX index_temp_id ON <table_name> (temp_id);

SET NOCOUNT ON

DECLARE crsTemp CURSOR FOR
  SELECT temp_id FROM <table_name> FOR UPDATE OF temp_id
DECLARE @aux INT
DECLARE @count BIGINT

OPEN crsTemp
FETCH NEXT FROM crsTemp INTO @aux
```

```
SELECT @count = 0
WHILE (@@FETCH_STATUS = 0)
BEGIN
  SELECT @count = @count + 1
  UPDATE <table_name> SET temp_id = @count WHERE CURRENT OF crsTemp
  FETCH NEXT FROM crsTemp INTO @aux
END

CLOSE crsTemp
DEALLOCATE crsTemp
```

sql_temp_id_drop.conf **Templates de origem (*Source*)** **Banco de dados *Microsoft SQL Server***

```
--The column must be called id_converter, but the index can have any name
DROP INDEX <table_name>.index_temp_id;
ALTER TABLE <table_name> DROP COLUMN temp_id;
```

default.ini **Templates de origem (*Source*)** **Banco de dados *PostgreSQL***

```
[Parameter]
;;DBMS: PostgreSQL 8.3

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; A list of tables to be ignored by the system, separated by
;; semicolon. All the tables starting by the words will be
;; ignored
;;
;; Default: ""
;; Type: String
;; Usage example :
;; "Table01;Table02;pg_;sql_;"
Ignored Tables=DUAL;pg_;sql_;administrable_;applicable_;attributes;check_;column_;columns;
constraint_;data_type_;domain_;domains;element_;enabled_;information_;key_column_;parameters;
referential_constraints;role_;routine_;routines;schemata;sequences;table_;tables;triggered_;
triggers;usage_;view_;views;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Ignored Words
;;
;; Type: String
;; Usage example: "DBA.;::numeric"
Ignored Words=public.;::numeric;::text;::timestamp without time zone;::character varying;ARRAY[;[;];

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Replace Words. Replace Words in the SQL stataments.
;;
;; Type: String
;; Usage example: "TRIM(:dbo.TRIM(;"
Replace Words== ANY:IN;((':(';')):');)(25):);

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Migrate Views
;;
;; Default: 1
;; Type: Boolean (1=True/0=False)
Migrate Views=1

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Migrate Check Constraints
;;
;; Default: 1
;; Type: Boolean (1=True/0=False)
Migrate Check Constraints=1
```

sql_definition_autoincrement.conf **Templates de origem (*Source*)** **Banco de dados *PostgreSQL***

```
SELECT
column_name

FROM
information_schema.columns

WHERE
table_name = '<table_name>' AND
SUBSTR(column_default, 1, 7) = 'nextval'

ORDER BY
column_name;
```

sql_definition_check.conf **Templates de origem (*Source*)** **Banco de dados *PostgreSQL***

```
SELECT
c.conname AS check_name,
```

```
'CHECK ' || REPLACE(c.consrc, 'btrim(', 'TRIM(') AS check_rule

FROM
pg_constraint c

JOIN pg_class t ON
c.conrelid  = t.oid

WHERE
TRIM(c.consrc) <> '' AND
t.relname = '<table_name>';
```

sql_definition_foreign_key.conf **Templates de origem (*Source*)** **Banco de dados *PostgreSQL***

```
SELECT DISTINCT
ccu.table_name AS table_name_dest,
tc.constraint_name AS role,
TRIM((CASE WHEN rc.update_rule = 'NO ACTION' THEN '' ELSE 'ON UPDATE ' || rc.update_rule END) || ' ' ||
     (CASE WHEN rc.delete_rule = 'NO ACTION' THEN '' ELSE 'ON DELETE ' || rc.delete_rule END)) AS actions,
(SELECT
 TRIM(LIST(' ' || kcu.column_name))
 FROM (SELECT * FROM information_schema.key_column_usage kcu ORDER BY kcu.ordinal_position) kcu
 WHERE kcu.constraint_catalog = tc.constraint_catalog AND
       kcu.constraint_schema = tc.constraint_schema AND
       kcu.constraint_name = tc.constraint_name) AS columns_origin,
(SELECT
 TRIM(LIST(' ' || kcu.column_name))
 FROM (SELECT * FROM information_schema.key_column_usage kcu ORDER BY kcu.ordinal_position) kcu
 WHERE kcu.table_name = ccu.table_name AND
       kcu.constraint_schema = ccu.constraint_schema AND
       kcu.constraint_name = ccu.constraint_name) AS columns_destination

FROM
information_schema.table_constraints tc

LEFT JOIN information_schema.referential_constraints rc ON
tc.constraint_catalog = rc.constraint_catalog AND
tc.constraint_schema = rc.constraint_schema AND
tc.constraint_name = rc.constraint_name

LEFT JOIN information_schema.constraint_column_usage ccu ON
ccu.constraint_catalog = rc.unique_constraint_catalog AND
ccu.constraint_schema = rc.unique_constraint_schema AND
ccu.constraint_name = rc.unique_constraint_name

WHERE
tc.constraint_type = 'FOREIGN KEY' AND
tc.table_name = '<table_name>'

ORDER BY
tc.constraint_name;
```

sql_definition_index.conf **Templates de origem (*Source*)** **Banco de dados *PostgreSQL***

```
SELECT
ic.relname AS index_name,
(CASE WHEN i.indisunique THEN 'U' ELSE 'N' END) AS index_unique,

(SELECT TRIM(LIST(' ' || ta.attname))
 FROM (SELECT * FROM pg_attribute ia ORDER BY ia.attnum) ia,
      (SELECT * FROM pg_attribute ta ORDER BY ta.attnum) ta
 WHERE ia.attrelid = i.indexrelid AND
       ta.attrelid = bc.oid AND
       ta.attrelid = i.indrelid AND
       ta.attnum = i.indkey[ia.attnum - 1]) AS index_columns,

(SELECT
 CASE WHEN (pg_indexes.indexdef like '%(' || (SELECT ta.attname
                                              FROM pg_attribute ia, pg_attribute ta
                                              WHERE ia.attrelid = i.indexrelid AND
                                                    ta.attrelid = bc.oid AND
                                                    ta.attrelid = i.indrelid AND
                                                    ta.attnum = i.indkey[ia.attnum - 1]
                                              LIMIT 1) || ' DESC%') THEN
   'DESC'
 ELSE
   'ASC'
 END FROM pg_indexes WHERE pg_indexes.tablename = bc.relname AND pg_indexes.indexname = ic.relname) AS
index_order,

(SELECT TRIM(LIST(' ' || ta.attname || ' ' ||
        (SELECT CASE WHEN (pg_indexes.indexdef like '%(' || ta.attname || ' DESC%') OR
                          (pg_indexes.indexdef like '%, ' || ta.attname || ' DESC%') THEN
```

```
                'DESC'
              ELSE
                'ASC'
              END FROM pg_indexes WHERE pg_indexes.tablename = bc.relname AND pg_indexes.indexname =
ic.relname)))
 FROM (SELECT * FROM pg_attribute ia ORDER BY ia.attnum) ia,
      (SELECT * FROM pg_attribute ta ORDER BY ta.attnum) ta
 WHERE ia.attrelid = i.indexrelid AND
       ta.attrelid = bc.oid AND
       ta.attrelid = i.indrelid AND
       ta.attnum = i.indkey[ia.attnum - 1]) AS index_columns_order

FROM
pg_class bc

JOIN pg_index i ON
i.indrelid = bc.oid

JOIN pg_class ic ON
ic.oid = i.indexrelid

WHERE
bc.relname = '<table_name>' AND
i.indisprimary IS FALSE

ORDER BY
index_unique DESC,
index_name;
```

sql_definition_table.conf **Templates de origem (*Source*)** **Banco de dados *PostgreSQL***

```
SELECT

'CREATE VIEW <table_name> AS ' ||
(SELECT view_definition
 FROM information_schema.views v
 WHERE v.table_catalog = t.table_catalog AND
       v.table_schema = t.table_schema AND
       v.table_name = t.table_name) AS view_def,

COALESCE((SELECT CAST('Y' AS CHAR(1)) FROM information_schema.table_constraints tc
          JOIN information_schema.key_column_usage kcu ON
               tc.constraint_catalog = kcu.constraint_catalog AND
               tc.constraint_schema = kcu.constraint_schema AND
               tc.constraint_name = kcu.constraint_name
          WHERE tc.constraint_type = 'PRIMARY KEY' AND
                tc.table_name = c.table_name AND
                kcu.column_name = c.column_name), 'N') AS pkey,

CAST((CASE WHEN is_nullable = 'YES' THEN 'Y' ELSE 'N' END) AS CHAR(1)) AS nulls,
COALESCE(numeric_precision, character_maximum_length) AS width,
COALESCE(numeric_scale, 0) AS scale,

c.column_name,

(CASE
  WHEN SUBSTR(UPPER(c.column_default), 1, 19) = '(' || CHR(39) || 'NOW' || CHR(39) || '::TEXT)::DATE' THEN
'CURRENT_DATE'
  WHEN SUBSTR(UPPER(c.column_default), 1, 19) = '(' || CHR(39) || 'NOW' || CHR(39) || '::TEXT)::TIME' THEN
'CURRENT_TIME'
  WHEN SUBSTR(UPPER(c.column_default), 1, 3) = 'NOW' THEN 'CURRENT_TIMESTAMP'
  WHEN SUBSTR(UPPER(c.column_default), 1, 7) = 'NEXTVAL' THEN ''
  WHEN SUBSTR(c.column_default, 1, 1) = CHR(39) THEN SUBSTR(c.column_default, 1, POSITION ('::' IN
c.column_default) - 1)
ELSE
  c.column_default
END) AS default_column,
NULL AS check_column,

CAST(TRIM(CASE
       WHEN UPPER(data_type) = 'MONEY' THEN 'REAL'
       WHEN UPPER(data_type) = 'CHARACTER' THEN 'CHAR'
       WHEN UPPER(data_type) = 'CHARACTER VARYING' THEN 'VARCHAR'
       WHEN UPPER(data_type) = 'TIME WITH TIME ZONE' THEN 'TIME'
       WHEN UPPER(data_type) = 'TIME WITHOUT TIME ZONE' THEN 'TIME'
       WHEN UPPER(data_type) = 'TIMESTAMP WITH TIME ZONE' THEN 'TIMESTAMP'
       WHEN UPPER(data_type) = 'TIMESTAMP WITHOUT TIME ZONE' THEN 'TIMESTAMP'
       ELSE
         UPPER(c.data_type)
     END) AS VARCHAR(254)) AS data_type,

CAST((CASE WHEN SUBSTR(column_default, 1, 7) = 'nextval' THEN 'Y' ELSE 'N' END) AS CHAR(1)) AS
is_autoincrement,
```

```
(CASE WHEN t.table_type = 'VIEW' THEN 'VIEW' ELSE 'TABLE' END) AS table_type

FROM
information_schema.tables t

JOIN information_schema.columns c ON
c.table_catalog = t.table_catalog AND
c.table_schema = t.table_schema AND
c.table_name = t.table_name

WHERE
c.table_name = '<table_name>'

ORDER BY
c.ordinal_position;
```

sql_definition_unique_key.conf **Templates de origem (*Source*)** **Banco de dados *PostgreSQL***

```
SELECT DISTINCT
tc.constraint_name,

(SELECT
 TRIM(LIST(' ' || TRIM(kcu.column_name)))
 FROM (SELECT * FROM information_schema.key_column_usage kcu ORDER BY kcu.ordinal_position) kcu
 WHERE kcu.constraint_catalog = tc.constraint_catalog AND
       kcu.constraint_schema = tc.constraint_schema AND
       kcu.constraint_name = tc.constraint_name) AS constraint_columns,

(SELECT
 CASE WHEN (pg_indexes.indexdef like '%(' || (SELECT kcu.column_name
                                              FROM (SELECT * FROM information_schema.key_column_usage kcu
ORDER BY kcu.ordinal_position) kcu
                                              WHERE kcu.constraint_catalog = tc.constraint_catalog AND
                                                    kcu.constraint_schema = tc.constraint_schema AND
                                                    kcu.constraint_name = tc.constraint_name
                                              LIMIT 1) || ' DESC%') THEN
   'DESC'
 ELSE
   'ASC'
 END FROM pg_indexes WHERE pg_indexes.tablename = tc.table_name AND pg_indexes.indexname =
tc.constraint_name) AS index_order

FROM
information_schema.table_constraints tc

WHERE
tc.constraint_type = 'UNIQUE' AND
tc.table_name = '<table_name>'

ORDER BY
tc.constraint_name;
```

sql_exists_view.conf **Templates de origem (*Source*)** **Banco de dados *PostgreSQL***

```
SELECT
COUNT(*) AS view_exists

FROM
pg_views

WHERE
viewname NOT LIKE 'pg\_' AND
viewname = '<view_name>';
```

sql_format_column_date.conf **Templates de origem (*Source*)** **Banco de dados *PostgreSQL***

```
<column_name> || '' AS <column_name>
```

sql_format_column_time.conf **Templates de origem (*Source*)** **Banco de dados *PostgreSQL***

```
SUBSTR(<column_name> || '', 1, 12) AS <column_name>
```

sql_format_column_timestamp.conf **Templates de origem (*Source*)** **Banco de dados *PostgreSQL***

```
SUBSTR(<column_name> || '', 1, 23) AS <column_name>
```

sql_retrieve_data.conf **Templates de origem (*Source*)** **Banco de dados *PostgreSQL***

```
SELECT <columns_list> FROM <table_name>
```

sql_retrieve_data_2.conf **Templates de origem (*Source*)** **Banco de dados *PostgreSQL***

```
WHERE temp_id BETWEEN <start> AND <end>
```

sql_script_post-conversion.conf **Templates de origem (*Source*)** **Banco de dados *PostgreSQL***

sql_script_pre-conversion.conf **Templates de origem (*Source*)** **Banco de dados *PostgreSQL***

```
SET TERM ^;

CREATE OR REPLACE FUNCTION _list(text, text) RETURNS text AS $$
  SELECT CASE
           WHEN $2 IS NULL THEN $1
           WHEN $1 IS NULL THEN $2
           ELSE $1 OPERATOR(pg_catalog.||) ',' OPERATOR(pg_catalog.||) $2
         END
$$ IMMUTABLE LANGUAGE SQL^

CREATE AGGREGATE list (
  BASETYPE = text,
  SFUNC = _list,
  STYPE = text
)^
```

sql_table_columns.conf **Templates de origem (*Source*)** **Banco de dados *PostgreSQL***

```
SELECT
column_name,
TRIM(CASE
       WHEN UPPER(data_type) = 'MONEY' THEN 'REAL'
       WHEN UPPER(data_type) = 'CHARACTER' THEN 'CHAR'
       WHEN UPPER(data_type) = 'CHARACTER VARYING' THEN 'VARCHAR'
       WHEN UPPER(data_type) = 'TIME WITH TIME ZONE' THEN 'TIME'
       WHEN UPPER(data_type) = 'TIME WITHOUT TIME ZONE' THEN 'TIME'
       WHEN UPPER(data_type) = 'TIMESTAMP WITH TIME ZONE' THEN 'TIMESTAMP'
       WHEN UPPER(data_type) = 'TIMESTAMP WITHOUT TIME ZONE' THEN 'TIMESTAMP'
       ELSE
         UPPER(data_type)
     END) AS data_type

FROM
information_schema.columns

WHERE
table_name = '<table_name>'

ORDER BY
ordinal_position;
```

sql_table_count.conf **Templates de origem (*Source*)** **Banco de dados *PostgreSQL***

```
SELECT COUNT(*) AS total FROM <table_name>;
```

sql_table_list.conf **Templates de origem (*Source*)** **Banco de dados *PostgreSQL***

```
SELECT
t.table_name

FROM
INFORMATION_SCHEMA.TABLES t

ORDER BY
(CASE WHEN t.table_type = 'VIEW' THEN 1 ELSE 0 END),
t.table_name;
```

sql_temp_id_create.conf **Templates de origem (*Source*)** **Banco de dados *PostgreSQL***

```
--The column must be called id_converter, but the index can have any name
ALTER TABLE <table_name> ADD temp_id BIGSERIAL;
CREATE INDEX index_temp_id ON <table_name> (temp_id);
```

sql_temp_id_drop.conf **Templates de origem (*Source*)** **Banco de dados *PostgreSQL***

```
--The column must be called id_converter, but the index can have any name
DROP INDEX index_temp_id;
ALTER TABLE <table_name> DROP temp_id;
```

default.ini **Templates de origem (*Source*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
[Parameter]
;;DBMS: AAdaptive Server Anywhere 8.0.x

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; A list of tables to be ignored by the system, separated by
;; semicolon. All the tables starting by the words will be
;; ignored
```

```
;;
;; Default: ""
;; Type: String
;; Usage example:
;; "Table01;Table02;"
Ignored Tables=jdbc;satmp;ul_referenced;sys;pbcat;migrate_;ml_;ul_;spt_;DUAL;$;DUMMY;EXCLUDEOBJECT;
RowGenerator;rs_lastcommit;rs_threads;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Ignored Words
;;
;; Type: String
;; Usage example: "DBA.;::numeric"
Ignored Words=DBA.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Replace Words. Replace Words in the SQL stataments.
;;
;; Type: String
;; Usage example: "TRIM(:dbo.TRIM(;"
Replace Words=

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Migrate Views
;;
;; Default: 1
;; Type: Boolean (1=True/0=False)
Migrate Views=1

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Migrate Check Constraints
;;
;; Default: 1
;; Type: Boolean (1=True/0=False)
Migrate Check Constraints=1
```

sql_definition_autoincrement.conf **Templates de origem (*Source*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
SELECT DISTINCT
SYSCOLUMN.column_id, --This column is here to be used at the order by clause
SYSCOLUMN.column_name

FROM
SYSCOLUMN

JOIN SYSTABLE ON
SYSTABLE.table_id = SYSCOLUMN.table_id

WHERE
SYSCOLUMN."default" = 'autoincrement' AND
SYSTABLE.table_name = '<table_name>'

ORDER BY
SYSCOLUMN.column_id;
```

sql_definition_check.conf **Templates de origem (*Source*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
SELECT
SYSTABLE.table_name AS check_name,
REPLACE(REPLACE((CASE WHEN SYSTABLE.table_type = 'VIEW' THEN NULL ELSE SYSTABLE.view_def END),  CHAR(13), '
'),  CHAR(10), ' ') AS check_rule

FROM
SYSTABLE

WHERE
SYSTABLE.table_name = '<table_name>' AND
check_rule IS NOT NULL;
```

sql_definition_foreign_key.conf **Templates de origem (*Source*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
SELECT
fk.foreign_table_id,
fk.foreign_key_id,
fk.primary_table_id,
pk_tab.table_name as table_name_dest,
fk.role,
TRIM(((SELECT
      (CASE t.referential_action
         WHEN 'C' THEN 'ON UPDATE CASCADE'
         WHEN 'D' THEN 'ON UPDATE SET DEFAULT'
         WHEN 'N' THEN 'ON UPDATE SET NULL'
         WHEN 'R' THEN '' --ON UPDATE NO ACTION
```

```
       END)
       FROM systrigger t
       WHERE
       t.foreign_key_id = fk.foreign_key_id AND
       t.foreign_table_id = fk_tab.table_id AND
       t.table_id = pk_tab.table_id AND
       t."event" IN ('C', 'U')) ||
     (SELECT
      (CASE t.referential_action
         WHEN 'C' THEN 'ON DELETE CASCADE'
         WHEN 'D' THEN 'ON DELETE SET DEFAULT'
         WHEN 'N' THEN 'ON DELETE SET NULL'
         WHEN 'R' THEN '' --ON DELETE NO ACTION
      END)
      FROM systrigger t
      WHERE
      t.foreign_key_id = fk.foreign_key_id AND
      t.foreign_table_id = fk_tab.table_id AND
      t.table_id = pk_tab.table_id AND
      t."event" = 'D'))) AS actions

FROM
SYS.SYSFOREIGNKEY AS fk

JOIN SYS.SYSTABLE AS fk_tab ON
fk_tab.table_id = fk.foreign_table_id

JOIN SYS.SYSUSERPERM AS fk_up ON
fk_up.user_id = fk_tab.creator

JOIN SYS.SYSTABLE AS pk_tab ON
pk_tab.table_id = fk.primary_table_id

JOIN SYS.SYSUSERPERM AS pk_up ON
pk_up.user_id = pk_tab.creator

WHERE
fk_tab.table_name = '<table_name>'

ORDER BY
fk.role;
```

sql_definition_foreign_key_2.conf **Templates de origem (*Source*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
SELECT DISTINCT
fkc.primary_column_id, --This column is here to be used at the order by clause
TRIM(fk_col.column_name) AS column_name_origin,
TRIM(pk_col.column_name) AS column_name_destination

FROM
SYS.SYSFKCOL AS fkc

JOIN SYS.SYSCOLUMN AS fk_col ON
fk_col.table_id = fkc.foreign_table_id AND
fk_col.column_id = fkc.foreign_column_id

JOIN SYS.SYSCOLUMN AS pk_col ON
pk_col.column_id = fkc.primary_column_id

WHERE
fkc.foreign_table_id = <foreign_table_id> AND
fkc.foreign_key_id = <foreign_key_id> AND
pk_col.table_id = <primary_table_id>

ORDER BY
fkc.primary_column_id;
```

sql_definition_index.conf **Templates de origem (*Source*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
SELECT
idx.index_name,
CASE idx."unique"
   WHEN 'Y' THEN 'U' --UNIQUE INDEX
   WHEN 'N' THEN 'N' --Non-unique INDEX
END AS index_unique,

TRIM((SELECT
      TRIM(LIST(' ' || column_name))
      FROM SYS.SYSIXCOL JOIN SYS.SYSCOLUMN
      WHERE index_id = idx.index_id AND
            SYSIXCOL.table_id = idx.table_id)) AS index_columns,

CASE WHEN ((SELECT FIRST "order"
```

```
            FROM SYS.SYSIXCOL JOIN SYS.SYSCOLUMN
            WHERE index_id = idx.index_id AND
                  SYSIXCOL.table_id = idx.table_id) = 'A') THEN
  'ASC'
ELSE
  'DESC'
END AS index_order,

TRIM((SELECT
      TRIM(LIST(' ' || column_name || ' ' || (CASE WHEN "order" = 'A' THEN 'ASC' ELSE 'DESC' END)))
      FROM SYS.SYSIXCOL JOIN SYS.SYSCOLUMN
      WHERE index_id = idx.index_id AND
            SYSIXCOL.table_id = idx.table_id)) AS index_columns_order

FROM
SYS.SYSTABLE AS tab

KEY JOIN SYS.SYSFILE AS file

KEY JOIN SYS.SYSINDEX AS idx

JOIN SYS.SYSUSERPERM AS up ON
up.user_id = idx.creator

WHERE
idx."unique" IN ('Y', 'N') AND --UNIQUE INDEX OR Non-unique INDEX
tab.table_name = '<table_name>'

ORDER BY
(CASE idx."unique" WHEN 'Y' THEN 0 WHEN 'N' THEN 1 END),
idx.index_name;
```

sql_definition_table.conf **Templates de origem (*Source*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
SELECT DISTINCT
SYSCOLUMN.column_id, --This column is here to be used at the order by clause
CASE WHEN (SYSTABLE.view_def IS NOT NULL) AND (SYSTABLE.table_type = 'VIEW') THEN
   SYSTABLE.view_def || ';'
ELSE
   NULL
END AS view_def,
SYSCOLUMN.pkey,
SYSCOLUMN.nulls,
SYSCOLUMN.width,
SYSCOLUMN.scale,
SYSCOLUMN.column_name,
CAST(TRIM(SYSCOLUMN."default") AS VARCHAR(254)) AS default_column,
TRIM(SYSCOLUMN."check") AS check_column,
CASE TRIM(SYSDOMAIN.domain_name)
  WHEN 'double' THEN 'double precision'
  WHEN 'float' THEN 'real'
  WHEN 'long varchar' THEN 'text'
  ELSE TRIM(SYSDOMAIN.domain_name)
END AS data_type,
(CASE WHEN SYSCOLUMN."default" = 'autoincrement' THEN 'Y' ELSE 'N' END) AS is_autoincrement,
(CASE WHEN SYSTABLE.table_type = 'VIEW' THEN 'VIEW' ELSE 'TABLE' END) AS table_type

FROM
SYSCOLUMN

JOIN SYSTABLE ON
SYSTABLE.table_id = SYSCOLUMN.table_id

JOIN SYSDOMAIN ON
SYSDOMAIN.domain_id = SYSCOLUMN.domain_id

WHERE
SYSTABLE.table_name = '<table_name>'

ORDER BY
SYSCOLUMN.column_id;
```

sql_definition_unique_key.conf **Templates de origem (*Source*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
SELECT
SUBSTR(idx.index_name, 1, LOCATE(idx.index_name, ' ') - 1) AS constraint_name,

TRIM((SELECT
      TRIM(LIST(' ' || column_name))
      FROM SYS.SYSIXCOL JOIN SYS.SYSCOLUMN
      WHERE index_id = idx.index_id AND
            SYSIXCOL.table_id = idx.table_id)) AS constraint_columns,
```

```
CASE WHEN ((SELECT FIRST "order"
            FROM SYS.SYSIXCOL JOIN SYS.SYSCOLUMN
            WHERE index_id = idx.index_id AND
                  SYSIXCOL.table_id = idx.table_id) = 'A') THEN
       'ASC'
ELSE
       'DESC'
END AS index_order

FROM
SYS.SYSTABLE AS tab

KEY JOIN SYS.SYSFILE AS file

KEY JOIN SYS.SYSINDEX AS idx

JOIN SYS.SYSUSERPERM AS up ON
up.user_id = idx.creator

WHERE
idx."unique" = 'U' AND
tab.table_name = '<table_name>'

ORDER BY
idx.index_name;
```

sql_exists_view.conf **Templates de origem (*Source*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
SELECT
COUNT(*) AS view_exists

FROM
SYSTABLE

WHERE
table_name = '<view_name>' AND
table_type = 'VIEW';
```

sql_format_column_date.conf **Templates de origem (*Source*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
DATEFORMAT(<column_name>, 'yyyy-mm-dd') AS <column_name>
```

sql_format_column_time.conf **Templates de origem (*Source*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
DATEFORMAT(<column_name>, 'hh:mm:ss.sss') AS <column_name>
```

sql_format_column_timestamp.conf **Templates de origem (*Source*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
DATEFORMAT(<column_name>, 'yyyy-mm-dd hh:mm:ss.sss') AS <column_name>
```

sql_retrieve_data.conf **Templates de origem (*Source*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
SELECT <columns_list> FROM <table_name>
```

sql_retrieve_data_2.conf **Templates de origem (*Source*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
WHERE temp_id BETWEEN <start> AND <end>
```

sql_script_post-conversion.conf **Templates de origem (*Source*)** **Banco de dados *Sybase Adaptive Server Anywhere***

sql_script_pre-conversion.conf **Templates de origem (*Source*)** **Banco de dados *Sybase Adaptive Server Anywhere***

sql_table_columns.conf **Templates de origem (*Source*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
SELECT DISTINCT
SYSCOLUMN.column_id, --This column is here to be used at the order by clause
SYSCOLUMN.column_name AS column_name,
CASE TRIM(SYSDOMAIN.domain_name)
  WHEN 'double' THEN 'double precision'
  WHEN 'float' THEN 'real'
  WHEN 'long varchar' THEN 'text'
  ELSE TRIM(SYSDOMAIN.domain_name)
END AS data_type

FROM
SYSCOLUMN

JOIN SYSTABLE ON
SYSTABLE.table_id = SYSCOLUMN.table_id

JOIN SYSDOMAIN ON
```

```
SYSDOMAIN.domain_id = SYSCOLUMN.domain_id

WHERE
SYSTABLE.table_name = '<table_name>'

ORDER BY
SYSCOLUMN.column_id;
```

sql_table_count.conf	Templates de origem (*Source*)	Banco de dados *Sybase Adaptive Server Anywhere*

```
SELECT COUNT(*) AS total FROM <table_name>;
```

sql_table_count.conf	Templates de origem (*Source*)	Banco de dados *Sybase Adaptive Server Anywhere*

```
SELECT
SYSTABLE.table_name,
(CASE WHEN SYSTABLE.table_type = 'VIEW' THEN 1 ELSE 0 END) AS table_type --This column is here to be used at the order by clause

FROM
SYSTABLE

ORDER BY
table_type,
SYSTABLE.table_name;
```

sql_temp_id_create.conf	Templates de origem (*Source*)	Banco de dados *Sybase Adaptive Server Anywhere*

```
--The column must be called id_converter, but the index can have any name
ALTER TABLE <table_name> ADD temp_id BIGINT DEFAULT AUTOINCREMENT;
CREATE INDEX index_temp_id ON <table_name> (temp_id);
```

sql_temp_id_drop.conf	Templates de origem (*Source*)	Banco de dados *Sybase Adaptive Server Anywhere*

```
--The column must be called id_converter, but the index can have any name
DROP INDEX <table_name>.index_temp_id;
ALTER TABLE <table_name> DROP temp_id;
```

default.ini	Templates de destino (*Target*)	Banco de dados *Firebird*

```
[Parameter]
;;DBMS: Firebird 2.1.x

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Conversion of Object Names
;;
;; 0: None
;; 1: Upper
;; 2: Lower
;;
;; Default: 1
;; Type: Integer
Conversion of Object Names=1

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Maximum Length to Object Names
;;
;; Default: 31
;; Type: Integer
Maximum Length to Object Names=31

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Ignore Quotes in Views
;;
;; Default: 0
;; Type: Boolean (1=True/0=False)
Ignore Quotes in Views=0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Replace Data Type. When is necessary replace a data type for
;; another (on table definition), you can enter with this option.
;; If the original data type has the width or scale, the new data
;; type is obligatory has too. If you have more than one, you
;; need separate it with semicolon
;;
;; Default: ""
;; Type: String
;; Usage example:
;; "CHAR:VARCHAR;NUMERIC:DECIMAL;"
Replace Data Type=TEXT:BLOB SUB_TYPE 1 SEGMENT SIZE 80;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Replace Default Values. When is necessary replace a default
;; value for another used on table definition, you can enter with
;; this option. If you have more than one, you need separate it
```

```
;; with semicolon
;;
;; Default: ""
;; Type: String
;; Usage example:
;; "AUTOINCREMENT:;CURRENT DATE:CURRENT_DATE;"
Replace Default Values=AUTOINCREMENT:;CURRENT DATE:CURRENT_DATE;CURRENT TIME:CURRENT_TIME;CURRENT
TIMESTAMP:CURRENT_TIMESTAMP;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Replace Words. Replace Words in the SQL stataments.
;;
;; Type: String
;; Usage example: "TRIM(:dbo.TRIM(;"
Replace Words=

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Default identity keyword. The default identity keyword will be
;; used when column "is_autoincrement" from
;; \Configuration\Source\?\sql_definition_table_1.conf is equal
;; to 'Y'. If something is put here and column "is_autoincrement"
;; is equal to 'Y', the DEFAULT clause of CREATE TABLE statement
;; will be ignored.
;;
;; Type: String
;; Usage example: "IDENTITY" / "DEFAULT AUTOINCREMENT"
Default identity keyword=

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Maximum length for Precision to NUMERIC/DECIMAL data types
;;
;; Default: 18
;; Type: Integer
Maximum length for Precision to NUMERIC/DECIMAL=18

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Set Term. If you need change the term token, you can determine
;; the command to do it. It's usual in Firebird databases
;;
;; Default: "SET TERM"
;; Type: String
Set Term=SET TERM

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; You can put here the prefixes to the system generate the
;; object names
;;
;; Defaults:
;;  Prefix to Primary Key=PK_
;;  Prefix to Foreign Key=FK_
;;  Prefix to Unique Key=UK_
;;  Prefix to Check Constraint=CKC_
;;  Prefix to Sequence=SEQ_
;;  Prefix to Before Insert Trigger=TBI_
;;  Prefix to Index=X_
;;  Prefix to Unique Index=XU_
;; Type: String
Prefix to Primary Key=PK_
Prefix to Foreign Key=FK_
Prefix to Unique Key=UK_
Prefix to Check Constraint=CKC_
Prefix to Sequence=SEQ_
Prefix to Before Insert Trigger=TBI_
Prefix to Index=X_
Prefix to Unique Index=XU_

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; User Scripts. Configurations to User Scripts.
;;
;; Defaults:
;; Types: String
User Script Name #1=
User Script Source #1=
User Script Target #1=
User Script Name #2=
User Script Source #2=
User Script Target #2=
User Script Name #3=
User Script Source #3=
User Script Target #3=
User Script Name #4=
User Script Source #4=
User Script Target #4=
```

```
User Script Name #5=
User Script Source #5=
User Script Target #5=
User Script Name #6=
User Script Source #6=
User Script Target #6=
User Script Name #7=
User Script Source #7=
User Script Target #7=
User Script Name #8=
User Script Source #8=
User Script Target #8=
User Script Name #9=
User Script Source #9=
User Script Target #9=

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Run user script before data conversion. To each table that will
;; have the data converted, this script will be executed before
;; the conversion.
;;
;; Defaults:
;; Types: Integer
Run user Script before data conversion=

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Run user script after data conversion. To each table that will
;; have the data converted, this script will be executed before
;; the conversion.
;;
;; Defaults:
;; Types: Integer
Run user script after data conversion=
```

sql_create_autoincrement.conf **Templates de destino (*Target*)** **Banco de dados *Firebird***

```
CREATE SEQUENCE <sequence_name>;
COMMENT ON SEQUENCE <sequence_name> IS '<table_name>.<column_name>';
EXECUTE PROCEDURE SP_SET_SEQUENCE('<table_name>', '<column_name>', '<sequence_name>');

SET TERM ^;
CREATE TRIGGER <trigger_name> FOR <table_name>
ACTIVE BEFORE INSERT POSITION 0 AS
BEGIN
    IF (NEW.<column_name> IS NULL) THEN
        NEW.<column_name> = NEXT VALUE FOR <sequence_name>;
END^
SET TERM ;^
```

sql_create_foreign_key.conf **Templates de destino (*Target*)** **Banco de dados *Firebird***

```
ALTER TABLE <table_origin_name>
  ADD CONSTRAINT <foreign_key_name> FOREIGN KEY (<columns_list_origin>)
    REFERENCES <table_destination_name> (<columns_list_destination>) <action>
  USING INDEX <index_name>;
```

sql_create_index.conf **Templates de destino (*Target*)** **Banco de dados *Firebird***

```
CREATE <index_unique> <index_order> INDEX <index_name> ON <table_name> (<columns_list>);
```

sql_create_table.conf **Templates de destino (*Target*)** **Banco de dados *Firebird***

```
CREATE TABLE <table_name> (
       <column_name> <data_type> <default> <not_null>,
              CONSTRAINT <check_name> <check_rule>,
              CONSTRAINT <primary_key_name> PRIMARY KEY (<primary_key_columns>) USING INDEX <index_name>
);
```

sql_create_unique_key.conf **Templates de destino (*Target*)** **Banco de dados *Firebird***

```
ALTER TABLE <table_name>
  ADD CONSTRAINT <unique_name> UNIQUE (<columns_list>)
  USING <index_order> INDEX <index_name>;
```

sql_exists_index.conf **Templates de destino (*Target*)** **Banco de dados *Firebird***

```
SELECT COUNT(*) AS index_exists

FROM
RDB$INDEX_SEGMENTS S

JOIN RDB$INDICES I ON
I.RDB$INDEX_NAME = S.RDB$INDEX_NAME

WHERE
```

```
(S.RDB$INDEX_NAME NOT STARTING WITH 'RDB$' AND
 I.RDB$RELATION_NAME = '<table_name>')
```

sql_exists_index_2.conf **Templates de destino (*Target*)** **Banco de dados *Firebird***

```
OR S.RDB$FIELD_NAME = '<column_name>'
```

sql_exists_table.conf **Templates de destino (*Target*)** **Banco de dados *Firebird***

```
SELECT
COUNT(*) AS table_exists

FROM
RDB$RELATIONS R

WHERE
R.RDB$RELATION_NAME = '<table_name>';
```

sql_find_constraints.conf **Templates de destino (*Target*)** **Banco de dados *Firebird***

```
SELECT COUNT(*) AS total FROM RDB$RELATION_CONSTRAINTS WHERE RDB$CONSTRAINT_NAME = '<object_name>';
/*<table_Name>*/
```

sql_find_indices.conf **Templates de destino (*Target*)** **Banco de dados *Firebird***

```
SELECT COUNT(*) AS total FROM RDB$INDICES WHERE RDB$INDEX_NAME = '<object_name>'; /*<table_name>*/
```

sql_find_sequences.conf **Templates de destino (*Target*)** **Banco de dados *Firebird***

```
SELECT COUNT(*) AS total FROM RDB$GENERATORS WHERE RDB$GENERATOR_NAME = '<object_name>'; /*<table_name>*/
```

sql_find_triggers.conf **Templates de destino (*Target*)** **Banco de dados *Firebird***

```
SELECT COUNT(*) AS total FROM RDB$TRIGGERS WHERE RDB$TRIGGER_NAME = '<object_name>'; /*<table_name>*/
```

sql_insert_table.conf **Templates de destino (*Target*)** **Banco de dados *Firebird***

```
INSERT INTO <table_name> (<columns_list>) VALUES (<values_list>);
```

sql_script_post-conversion.conf **Templates de destino (*Target*)** **Banco de dados *Firebird***

```
RECREATE VIEW DUAL (FIELD) AS SELECT NULL AS FIELD FROM RDB$DATABASE;
```

sql_script_pre-conversion.conf **Templates de destino (*Target*)** **Banco de dados *Firebird***

```
SET TERM ^ ;

RECREATE PROCEDURE SP_SET_SEQUENCE (
    table_name VARCHAR(31),
    column_name VARCHAR(31),
    sequence_name VARCHAR(31))
RETURNS (
    sequence_value BIGINT
) AS
DECLARE VARIABLE sql VARCHAR(255);
BEGIN
    sql = 'SELECT (CAST(COALESCE(MAX(' || column_name || '), 0) AS BIGINT)) AS TOTAL FROM ' ||
table_name;
    EXECUTE STATEMENT sql INTO :sequence_value;
    IF (sequence_value < 0) THEN
      sequence_value = 0;

  EXECUTE STATEMENT 'ALTER SEQUENCE ' || sequence_name || ' RESTART WITH  ' || sequence_value;
  SUSPEND;
END^

SET TERM ; ^
```

default.ini **Templates de destino (*Target*)** **Banco de dados *Microsoft SQL Server***

```
[Parameter]
;;DBMS: Microsoft SQL Server 2005

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Conversion of Object Names
;;
;; 0: None
;; 1: Upper
;; 2: Lower
;;
;; Default: 1
;; Type: Integer
```

```
Conversion of Object Names=2

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Maximum Length to Object Names
;;
;; Default: 31
;; Type: Integer
Maximum Length to Object Names=128

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Ignore Quotes in Views
;;
;; Default: 0
;; Type: Boolean (1=True/0=False)
Ignore Quotes in Views=0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Replace Data Type. When is necessary replace a data type for
;; another (on table definition), you can enter with this option.
;; If the original data type has the width or scale, the new data
;; type is obligatory has too. If you have more than one, you
;; need separate it with semicolon
;;
;; Default: ""
;; Type: String
;; Usage example:
;; "CHAR:VARCHAR;NUMERIC:DECIMAL;"
Replace Data Type=DATE:DATETIME;TIME:DATETIME;TIMESTAMP:DATETIME

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Replace Default Values. When is necessary replace a default
;; value for another used on table definition, you can enter with
;; this option. If you have more than one, you need separate it
;; with semicolon
;;
;; Default: ""
;; Type: String
;; Usage example:
;; "AUTOINCREMENT:;CURRENT DATE:CURRENT_DATE;"
Replace Default Values=CURRENT_DATE:CURRENT_TIMESTAMP;CURRENT_TIME:CURRENT_TIMESTAMP;CURRENT DATE:CURRENT
TIMESTAMP;CURRENT_TIME:CURRENT_TIMESTAMP;CURRENT TIMESTAMP:CURRENT_TIMESTAMP;CURRENT TIME:CURRENT_TIMESTAMP

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Replace Words. Replace Words in the SQL stataments.
;;
;; Type: String
;; Usage example: "TRIM(:dbo.TRIM(;"
Replace Words=TRIM(:dbo.TRIM(;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Default identity keyword. The default identity keyword will be
;; used when column "is_autoincrement" from
;; \Configuration\Source\?\sql_definition_table_1.conf is equal
;; to 'Y'. If something is put here and column "is_autoincrement"
;; is equal to 'Y', the DEFAULT clause of CREATE TABLE statement
;; will be ignored.
;;
;; Type: String
;; Usage example: "IDENTITY" / "DEFAULT AUTOINCREMENT"
Default identity keyword=IDENTITY

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Maximum length for Precision to NUMERIC/DECIMAL data types
;;
;; Default: 18
;; Type: Integer
Maximum length for Precision to NUMERIC/DECIMAL=18

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Set Term. If you need change the term token, you can determine
;; the command to do it. It's usual in Firebird databases
;;
;; Default: "SET TERM"
;; Type: String
Set Term=SET TERM

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; You can put here the prefixes to the system generate the
;; object names
;;
;; Defaults:
;;  Prefix to Primary Key=PK_
;;  Prefix to Foreign Key=FK_
```

```
;;  Prefix to Unique Key=UK_
;;  Prefix to Check Constraint=CKC_
;;  Prefix to Sequence=SEQ_
;;  Prefix to Before Insert Trigger=TBI_
;;  Prefix to Index=X_
;;  Prefix to Unique Index=XU_
;; Type: String
Prefix to Primary Key=PK_
Prefix to Foreign Key=FK_
Prefix to Unique Key=UK_
Prefix to Check Constraint=CKC_
Prefix to Sequence=SEQ_
Prefix to Before Insert Trigger=TBI_
Prefix to Index=X_
Prefix to Unique Index=XU_

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; User Scripts. Configurations to User Scripts.
;;
;; Defaults:
;; Types: String
User Script Name #1=Disable Columns with IDENTIFY Clause
User Script Source #1=sql_definition_autoincrement.conf
User Script Target #1=sql_user_script_#1.conf
User Script Name #2=Enable Columns with IDENTIFY Clause
User Script Source #2=sql_definition_autoincrement.conf
User Script Target #2=sql_user_script_#2.conf
User Script Name #3=
User Script Source #3=
User Script Target #3=
User Script Name #4=
User Script Source #4=
User Script Target #4=
User Script Name #5=
User Script Source #5=
User Script Target #5=
User Script Name #6=
User Script Source #6=
User Script Target #6=
User Script Name #7=
User Script Source #7=
User Script Target #7=
User Script Name #8=
User Script Source #8=
User Script Target #8=
User Script Name #9=
User Script Source #9=
User Script Target #9=

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Run user script before data conversion. To each table that will
;; have the data converted, this script will be executed before
;; the conversion.
;;
;; Defaults:
;; Types: Integer
Run user Script before data conversion=1

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Run user script after data conversion. To each table that will
;; have the data converted, this script will be executed before
;; the conversion.
;;
;; Defaults:
;; Types: Integer
Run user script after data conversion=2
```

sql_create_autoincrement.conf **Templates de destino (*Target*)** **Banco de dados *Microsoft SQL Server***

sql_create_foreign_key.conf **Templates de destino (*Target*)** **Banco de dados *Microsoft SQL Server***

```
ALTER TABLE <table_origin_name>
  ADD CONSTRAINT <foreign_key_name> FOREIGN KEY (<columns_list_origin>)
    REFERENCES <table_destination_name> (<columns_list_destination>) <action>;
```

sql_create_index.conf **Templates de destino (*Target*)** **Banco de dados *Microsoft SQL Server***

```
CREATE <index_unique> INDEX <index_name> ON <table_name> (<columns_list_order>);
```

sql_create_table.conf **Templates de destino (*Target*)** **Banco de dados *Microsoft SQL Server***

```
CREATE TABLE <table_name> (
```

```
        <column_name> <data_type> <not_null> <default>,
                <check_rule>,
                CONSTRAINT <primary_key_name> PRIMARY KEY (<primary_key_columns>)
);
```

sql_create_unique_key.conf **Templates de destino (*Target*)** **Banco de dados *Microsoft SQL Server***

```
ALTER TABLE <table_name>
  ADD CONSTRAINT <unique_name> UNIQUE (<columns_list>);
```

sql_exists_index.conf **Templates de destino (*Target*)** **Banco de dados *Microsoft SQL Server***

```
SELECT
COUNT(DISTINCT col.column_id) AS index_exists

FROM
sys.tables AS tab

JOIN sys.columns AS col ON
tab.object_id = col.object_id

JOIN sys.index_columns AS idx ON
idx.object_id = tab.object_id AND
idx.column_id = col.column_id

WHERE
tab.name = '<table_name>'
```

sql_exists_index_2.conf **Templates de destino (*Target*)** **Banco de dados *Microsoft SQL Server***

```
OR col.name = '<column_name>'
```

sql_exists_table.conf **Templates de destino (*Target*)** **Banco de dados *Microsoft SQL Server***

```
SELECT
COUNT(*) AS table_exists

FROM
(SELECT tab.name FROM sys.tables AS tab
 UNION
 SELECT tab.name FROM sys.views AS tab) tab

WHERE
tab.name = '<table_name>';
```

sql_find_constraints.conf **Templates de destino (*Target*)** **Banco de dados *Microsoft SQL Server***

```
SELECT
COUNT(*) AS total

FROM
INFORMATION_SCHEMA.TABLE_CONSTRAINTS

WHERE
constraint_catalog = DB_NAME() AND
constraint_name = '<object_name>'; /*<table_name>*/
```

sql_find_indices.conf **Templates de destino (*Target*)** **Banco de dados *Microsoft SQL Server***

```
SELECT
COUNT(*) AS total

FROM
sys.tables AS tab

JOIN sys.indexes AS idx ON
idx.object_id = tab.object_id

WHERE
tab.name = '<table_name>' AND
idx.name = '<object_name>'
```

sql_find_sequences.conf **Templates de destino (*Target*)** **Banco de dados *Microsoft SQL Server***

sql_find_triggers.conf **Templates de destino (*Target*)** **Banco de dados *Microsoft SQL Server***

sql_insert_table.conf **Templates de destino (*Target*)** **Banco de dados *Microsoft SQL Server***

```
INSERT INTO <table_name> (<columns_list>) VALUES (<values_list>);
```

sql_script_post-conversion.conf **Templates de destino (*Target*)** **Banco de dados *Microsoft SQL Server***

```
CREATE VIEW DUAL (FIELD) AS SELECT NULL AS FIELD;
```

sql_script_pre-conversion.conf **Templates de destino (*Target*)** **Banco de dados *Microsoft SQL Server***

```
CREATE FUNCTION dbo.TRIM(@string VARCHAR(MAX)) RETURNS VARCHAR(MAX)
BEGIN
  RETURN LTRIM(RTRIM(@string))
END;
```

sql_user_script_#1.conf **Templates de destino (*Target*)** **Banco de dados *Microsoft SQL Server***

```
SET IDENTITY_INSERT <table_name> ON;
```

sql_user_script_#2.conf **Templates de destino (*Target*)** **Banco de dados *Microsoft SQL Server***

```
SET IDENTITY_INSERT <table_name> OFF;
```

default.ini **Templates de destino (*Target*)** **Banco de dados *PostgreSQL***

```
[Parameter]
;;DBMS: PostgreSQL 8.3

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Conversion of Object Names
;;
;; 0: None
;; 1: Upper
;; 2: Lower
;;
;; Default: 2
;; Type: Integer
Conversion of Object Names=2

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Maximum Length to Object Names
;;
;; Default: 31
;; Type: Integer
Maximum Length to Object Names=64

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Ignore Quotes in Views
;;
;; Default: 0
;; Type: Boolean (1=True/0=False)
Ignore Quotes in Views=0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Replace Data Type. When is necessary replace a data type for
;; another (on table definition), you can enter with this option.
;; If the original data type has the width or scale, the new data
;; type is obligatory has too. If you have more than one, you
;; need separate it with semicolon
;;
;; Default: ""
;; Type: String
;; Usage example:
;; "CHAR:VARCHAR;NUMERIC:DECIMAL;"
Replace Data Type=CHAR:VARCHAR;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Replace Default Values. When is necessary replace a default
;; value for another used on table definition, you can enter with
;; this option. If you have more than one, you need separate it
;; with semicolon
;;
;; Default: ""
;; Type: String
;; Usage example:
;; "AUTOINCREMENT:;CURRENT DATE:CURRENT_DATE;"
Replace Default Values=AUTOINCREMENT:;CURRENT DATE:CURRENT_DATE;CURRENT TIME:CURRENT_TIME;CURRENT
TIMESTAMP:CURRENT_TIMESTAMP;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Replace Words. Replace Words in the SQL stataments.
;;
;; Type: String
;; Usage example: "TRIM(:dbo.TRIM(;"
Replace Words=

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Default identity keyword. The default identity keyword will be
;; used when column "is_autoincrement" from
```

```
;; \Configuration\Source\?\sql_definition_table_1.conf is equal
;; to 'Y'. If something is put here and column "is_autoincrement"
;; is equal to 'Y', the DEFAULT clause of CREATE TABLE statement
;; will be ignored.
;;
;; Type: String
;; Usage example: "IDENTITY" / "DEFAULT AUTOINCREMENT"
Default identity keyword=

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Maximum length for Precision to NUMERIC/DECIMAL data types
;;
;; Default: 18
;; Type: Integer
Maximum length for Precision to NUMERIC/DECIMAL=18

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Set Term. If you need change the term token, you can determine
;; the command to do it. It's usual in Firebird databases
;;
;; Default: "SET TERM"
;; Type: String
Set Term=SET TERM

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; You can put here the prefixes to the system generate the
;; object names
;;
;; Defaults:
;;  Prefix to Primary Key=PK_
;;  Prefix to Foreign Key=FK_
;;  Prefix to Unique Key=UK_
;;  Prefix to Check Constraint=CKC_
;;  Prefix to Sequence=SEQ_
;;  Prefix to Before Insert Trigger=TBI_
;;  Prefix to Index=X_
;;  Prefix to Unique Index=XU_
;; Type: String
Prefix to Primary Key=PK_
Prefix to Foreign Key=FK_
Prefix to Unique Key=UK_
Prefix to Check Constraint=CKC_
Prefix to Sequence=SEQ_
Prefix to Before Insert Trigger=TBI_
Prefix to Index=X_
Prefix to Unique Index=XU_

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; User Scripts. Configurations to User Scripts.
;;
;; Defaults:
;; Types: String
User Script Name #1=
User Script Source #1=
User Script Target #1=
User Script Name #2=
User Script Source #2=
User Script Target #2=
User Script Name #3=
User Script Source #3=
User Script Target #3=
User Script Name #4=
User Script Source #4=
User Script Target #4=
User Script Name #5=
User Script Source #5=
User Script Target #5=
User Script Name #6=
User Script Source #6=
User Script Target #6=
User Script Name #7=
User Script Source #7=
User Script Target #7=
User Script Name #8=
User Script Source #8=
User Script Target #8=
User Script Name #9=
User Script Source #9=
User Script Target #9=

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Run user script before data conversion. To each table that will
;; have the data converted, this script will be executed before
```

```
;; the conversion.
;;
;; Defaults:
;; Types: Integer
Run user Script before data conversion=

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Run user script after data conversion. To each table that will
;; have the data converted, this script will be executed before
;; the conversion.
;;
;; Defaults:
;; Types: Integer
Run user script after data conversion=
```

sql_create_autoincrement.conf **Templates de destino (*Target*)** **Banco de dados *PostgreSQL***

```
CREATE SEQUENCE <sequence_name>;
COMMENT ON SEQUENCE <sequence_name> IS '<table_name>.<column_name>';
SELECT FN_SET_SEQUENCE('<table_name>', '<column_name>', '<sequence_name>');
ALTER TABLE <table_name> ALTER COLUMN <column_name> SET DEFAULT nextval('<sequence_name>');
```

sql_create_foreign_key.conf **Templates de destino (*Target*)** **Banco de dados *PostgreSQL***

```
ALTER TABLE <table_origin_name>
  ADD CONSTRAINT <foreign_key_name> FOREIGN KEY (<columns_list_origin>)
    REFERENCES <table_destination_name> (<columns_list_destination>) <action>;
```

sql_create_index.conf **Templates de destino (*Target*)** **Banco de dados *PostgreSQL***

```
CREATE <index_unique> INDEX <index_name> ON <table_name> (<columns_list_order>);
```

sql_create_table.conf **Templates de destino (*Target*)** **Banco de dados *PostgreSQL***

```
CREATE TABLE <table_name> (
       <column_name> <data_type> <default> <not_null>,
              CONSTRAINT <check_name> <check_rule>,
              CONSTRAINT <primary_key_name> PRIMARY KEY (<primary_key_columns>)
);
```

sql_create_unique_key.conf **Templates de destino (*Target*)** **Banco de dados *PostgreSQL***

```
ALTER TABLE <table_name>
  ADD CONSTRAINT <unique_name> UNIQUE (<columns_list>);
```

sql_exists_index.conf **Templates de destino (*Target*)** **Banco de dados *PostgreSQL***

```
SELECT
COUNT(*) AS index_exists

FROM
pg_class bc

JOIN pg_index i ON
i.indrelid = bc.oid

JOIN pg_class ic ON
ic.oid = i.indexrelid

JOIN pg_attribute ia ON
ia.attrelid = i.indexrelid

JOIN pg_attribute ta ON
ta.attrelid = i.indrelid AND
ta.attnum = i.indkey[ia.attnum - 1] AND
ta.attrelid = bc.oid

WHERE
(bc.relname NOT LIKE 'pg\_%' AND
 bc.relname NOT LIKE 'sql\_%' AND
 bc.relname = '<table_name>')
```

sql_exists_index_2.conf **Templates de destino (*Target*)** **Banco de dados *PostgreSQL***

```
OR ta.attname = '<column_name>'
```

sql_exists_table.conf **Templates de destino (*Target*)** **Banco de dados *PostgreSQL***

```
SELECT
COUNT(*) AS table_exists

FROM
pg_class bc
```

```
WHERE
bc.relname = '<table_name>';
```

sql_find_constraints.conf **Templates de destino (*Target*)** **Banco de dados *PostgreSQL***

```
SELECT COUNT(*) AS total FROM pg_constraint WHERE conname = '<object_name>'; /*<table_Name>*/
```

sql_find_indices.conf **Templates de destino (*Target*)** **Banco de dados *PostgreSQL***

```
SELECT COUNT(*) AS total FROM pg_index i JOIN pg_class ic ON ic.oid = i.indexrelid WHERE ic.relname =
'<object_name>'; /*<table_name>*/
```

sql_find_sequences.conf **Templates de destino (*Target*)** **Banco de dados *PostgreSQL***

```
SELECT
COUNT(*) AS total

FROM pg_class c

WHERE
c.relkind = 'S' AND
c.relnamespace IN (SELECT oid
                   FROM pg_namespace
                   WHERE nspname NOT LIKE 'pg_%' AND
                         nspname <> 'information_schema')
                         AND
c.relname = '<object_name>';  /*<table_name>*/
```

sql_find_triggers.conf **Templates de destino (*Target*)** **Banco de dados *PostgreSQL***

sql_insert_table.conf **Templates de destino (*Target*)** **Banco de dados *PostgreSQL***

```
INSERT INTO <table_name> (<columns_list>) VALUES (<values_list>);
```

sql_script_post-conversion.conf **Templates de destino (*Target*)** **Banco de dados *PostgreSQL***

```
CREATE VIEW DUAL (FIELD) AS SELECT NULL AS FIELD;
```

sql_script_pre-conversion.conf **Templates de destino (*Target*)** **Banco de dados *PostgreSQL***

```
SET TERM ^;
CREATE OR REPLACE FUNCTION FN_SET_SEQUENCE (IN table_name CHARACTER VARYING, IN column_name CHARACTER
VARYING, IN sequence_name CHARACTER VARYING) RETURNS BIGINT AS $$
DECLARE sql CHARACTER VARYING;
DECLARE sequence_value BIGINT;
BEGIN
  sql := 'SELECT (CAST(COALESCE(MAX(' || column_name || '), 0) AS BIGINT)) AS TOTAL FROM ' || table_name;
  EXECUTE sql INTO sequence_value;

  IF (sequence_value < 1) THEN
    sequence_value := 1;
  END IF;

  EXECUTE 'ALTER SEQUENCE ' || sequence_name || ' RESTART WITH  ' || sequence_value;
  RETURN sequence_value;
END $$ LANGUAGE plpgsql^
```

default.ini **Templates de destino (*Target*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
[Parameter]
;;DBMS: Adaptive Server Anywhere 8.0.x

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Conversion of Object Names
;;
;; 0: None
;; 1: Upper
;; 2: Lower
;;
;; Default: 1
;; Type: Integer
Conversion of Object Names=2

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Maximum Length to Object Names
;;
;; Default: 31
;; Type: Integer
Maximum Length to Object Names=128

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Ignore Quotes in Views
```

```
;;
;; Default: 0
;; Type: Boolean (1=True/0=False)
Ignore Quotes in Views=0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Replace Data Type. When is necessary replace a data type for
;; another (on table definition), you can enter with this option.
;; If the original data type has the width or scale, the new data
;; type is obligatory has too. If you have more than one, you
;; need separate it with semicolon
;;
;; Default: ""
;; Type: String
;; Usage example:
;; "CHAR:VARCHAR;NUMERIC:DECIMAL;"
Replace Data Type=

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Replace Default Values. When is necessary replace a default
;; value for another used on table definition, you can enter with
;; this option. If you have more than one, you need separate it
;; with semicolon
;;
;; Default: ""
;; Type: String
;; Usage example:
;; "AUTOINCREMENT:;CURRENT DATE:CURRENT_DATE;"
Replace Default Values=CURRENT_DATE:CURRENT DATE;CURRENT_TIME:CURRENT TIME;CURRENT_TIMESTAMP:CURRENT
TIMESTAMP;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Replace Words. Replace Words in the SQL stataments.
;;
;; Type: String
;; Usage example: "TRIM(:dbo.TRIM(;"
Replace Words=

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Default identity keyword. The default identity keyword will be
;; used when column "is_autoincrement" from
;; \Configuration\Source\?\sql_definition_table_1.conf is equal
;; to 'Y'. If something is put here and column "is_autoincrement"
;; is equal to 'Y', the DEFAULT clause of CREATE TABLE statement
;; will be ignored.
;;
;; Type: String
;; Usage example: "IDENTITY" / "DEFAULT AUTOINCREMENT"
Default identity keyword=

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Maximum length for Precision to NUMERIC/DECIMAL data types
;;
;; Default: 18
;; Type: Integer
Maximum length for Precision to NUMERIC/DECIMAL=18

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Set Term. If you need change the term token, you can determine
;; the command to do it. It's usual in Firebird databases
;;
;; Default: "SET TERM"
;; Type: String
Set Term=SET TERM

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; You can put here the prefixes to the system generate the
;; object names
;;
;; Defaults:
;;  Prefix to Primary Key=PK_
;;  Prefix to Foreign Key=FK_
;;  Prefix to Unique Key=UK_
;;  Prefix to Check Constraint=CKC_
;;  Prefix to Sequence=SEQ_
;;  Prefix to Before Insert Trigger=TBI_
;;  Prefix to Index=X_
;;  Prefix to Unique Index=XU_
;; Type: String
Prefix to Primary Key=PK_
Prefix to Foreign Key=FK_
Prefix to Unique Key=UK_
Prefix to Check Constraint=CKC_
```

```
Prefix to Sequence=SEQ_
Prefix to Before Insert Trigger=TBI_
Prefix to Index=X_
Prefix to Unique Index=XU_

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; User Scripts. Configurations to User Scripts.
;;
;; Defaults:
;; Types: String
User Script Name #1=
User Script Source #1=
User Script Target #1=
User Script Name #2=
User Script Source #2=
User Script Target #2=
User Script Name #3=
User Script Source #3=
User Script Target #3=
User Script Name #4=
User Script Source #4=
User Script Target #4=
User Script Name #5=
User Script Source #5=
User Script Target #5=
User Script Name #6=
User Script Source #6=
User Script Target #6=
User Script Name #7=
User Script Source #7=
User Script Target #7=
User Script Name #8=
User Script Source #8=
User Script Target #8=
User Script Name #9=
User Script Source #9=
User Script Target #9=

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Run user script before data conversion. To each table that will
;; have the data converted, this script will be executed before
;; the conversion.
;;
;; Defaults:
;; Types: Integer
Run user Script before data conversion=

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Run user script after data conversion. To each table that will
;; have the data converted, this script will be executed before
;; the conversion.
;;
;; Defaults:
;; Types: Integer
Run user script after data conversion=
```

sql_create_autoincrement.conf **Templates de destino (*Target*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
ALTER TABLE <table_name> ALTER <column_name> SET DEFAULT AUTOINCREMENT;
CREATE VARIABLE @aux BIGINT;
SET @aux = (SELECT COALESCE(MAX(<column_name>) + 1, 1) FROM <table_name>);
CALL sa_reset_identity('<table_name>', 'dba', @aux);
DROP VARIABLE @aux;
```

sql_create_foreign_key.conf **Templates de destino (*Target*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
ALTER TABLE <table_origin_name>
  ADD CONSTRAINT <foreign_key_name> FOREIGN KEY (<columns_list_origin>)
    REFERENCES <table_destination_name> (<columns_list_destination>) <action>;
```

sql_create_index.conf **Templates de destino (*Target*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
CREATE <index_unique> INDEX <index_name> ON <table_name> (<columns_list_order>);
```

sql_create_table.conf **Templates de destino (*Target*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
CREATE TABLE <table_name> (
       <column_name> <data_type> <not_null> <default>,
              <check_rule>,
              CONSTRAINT <primary_key_name> PRIMARY KEY (<primary_key_columns>)
);
```

sql_create_unique_key.conf **Templates de destino (*Target*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
ALTER TABLE <table_name>
  ADD CONSTRAINT <unique_name> UNIQUE (<columns_list>);
```

sql_exists_index.conf **Templates de destino (*Target*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
SELECT
COUNT(*) AS index_exists

FROM
SYS.SYSTABLE AS tab

KEY JOIN SYS.SYSINDEX AS idx

KEY JOIN SYS.SYSIXCOL AS ixcol

JOIN SYS.SYSCOLUMN col ON
col.table_id = ixcol.table_id AND
col.column_id = ixcol.column_id

WHERE
tab.table_name = '<table_name>'
```

sql_exists_index_2.conf **Templates de destino (*Target*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
OR col.column_name = '<column_name>'
```

sql_exists_table.conf **Templates de destino (*Target*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
SELECT
COUNT(*) AS table_exists

FROM
SYS.SYSTABLE AS tab

WHERE
UPPER(tab.table_name) = UPPER('<table_name>');
```

sql_find_constraints.conf **Templates de destino (*Target*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
SELECT
COUNT(*) AS total

FROM
SYS.SYSFOREIGNKEY AS fk

JOIN SYS.SYSTABLE AS fk_tab ON
fk_tab.table_id = fk.foreign_table_id

WHERE
fk_tab.table_name = '<table_name>' AND
fk.role = '<object_name>';
```

sql_find_indices.conf **Templates de destino (*Target*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
SELECT
COUNT(*) AS total

FROM
SYS.SYSTABLE AS tab

KEY JOIN SYS.SYSINDEX AS idx

WHERE
tab.table_name = '<table_name>' AND
idx.index_name = '<object_name>';
```

sql_find_sequences.conf **Templates de destino (*Target*)** **Banco de dados *Sybase Adaptive Server Anywhere***

sql_find_triggers.conf **Templates de destino (*Target*)** **Banco de dados *Sybase Adaptive Server Anywhere***

sql_insert_table.conf **Templates de destino (*Target*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
INSERT INTO <table_name> (<columns_list>) VALUES (<values_list>);
```

sql_script_post-conversion.conf **Templates de destino (*Target*)** **Banco de dados *Sybase Adaptive Server Anywhere***

```
CREATE VIEW DUAL (FIELD) AS SELECT NULL AS FIELD;
```

sql_script_pre-conversion.conf **Templates de destino (*Target*)** **Banco de dados *Sybase Adaptive Server Anywhere***

APÊNDICE E – MODELO DOS ARQUIVOS DE LOG DO SISTEMA

commit.log

```
Timestamp: 08/09/2008 10:26:55; Source DBMS: origem; Target DBMS: destino; Table
ID: 1; Line: 619;

Timestamp: 08/09/2008 10:27:40; Source DBMS: origem; Target DBMS: destino; Table
ID: 2; Line: 30000;

Timestamp: 08/09/2008 10:28:19; Source DBMS: origem; Target DBMS: destino; Table
ID: 2; Line: 60000;

Timestamp: 08/09/2008 10:29:04; Source DBMS: origem; Target DBMS: destino; Table
ID: 2; Line: 61034;
```

ddl.log

```
Timestamp: 06/09/2008 14:24:14
Source DBMS: origem
Target DBMS: destino
Table: 1

ALTER TABLE teste
  ADD CONSTRAINT fk_teste FOREIGN KEY (cod_teste)
    REFERENCES teste2 (cod_teste) ON DELETE SET NULL;

General SQL error.
ERROR: insert or update on table "teste" violates foreign key constraint "fk_teste"
Key (cod_teste)=( ) is not present in table "teste2".
```

dml.log

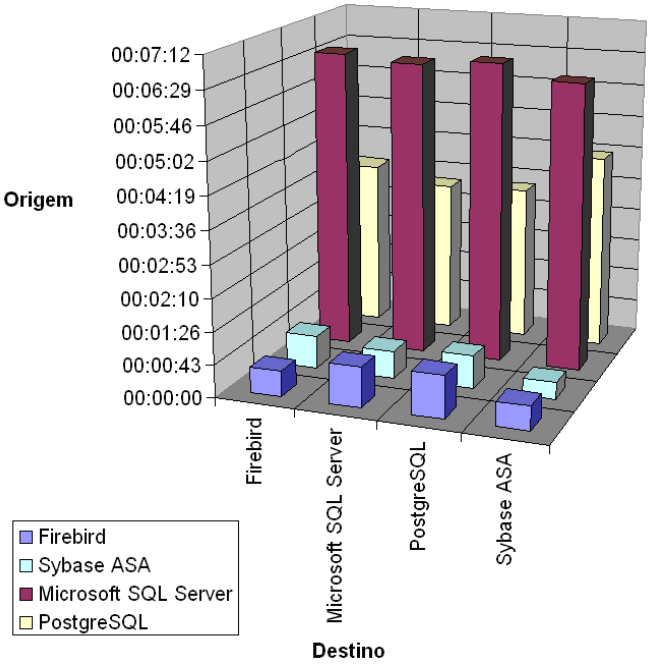
```
Timestamp: 07/09/2008 15:26:03
Source DBMS: origem
Target DBMS: destino
Line: 0

INSERT INTO teste (a, b) VALUES (1, 1);

General SQL error.
[Microsoft][SQL Native Client][SQL Server] Cannot insert explicit value for
identity column in table 'teste' when IDENTITY_INSERT is set to OFF.
[Microsoft][SQL Native Client][SQL Server]The statement has been terminated.
```

APÊNDICE F – ARQUIVOS FONTES DO SISTEMA

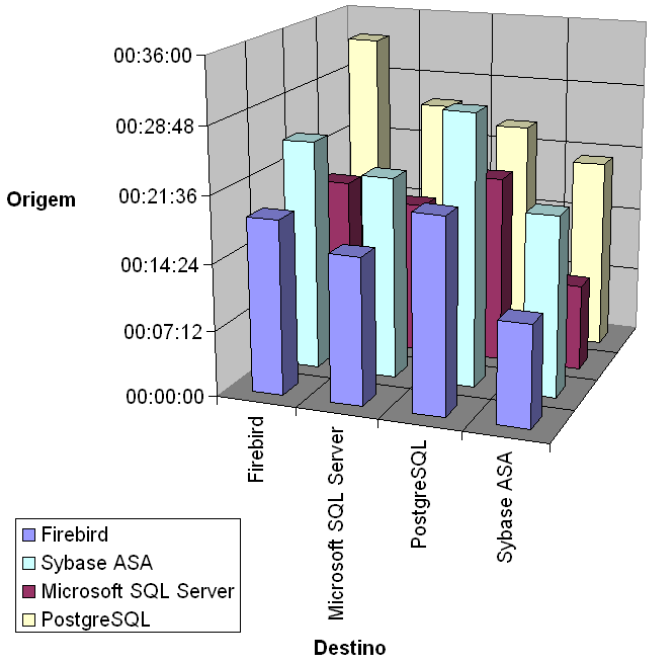
Migrate.cfg

```
Migrate.dof
Migrate.dpr
Migrate.res
UDMSystem.dfm
UDMSystem.pas
UFAbout.dfm
UFAbout.pas
UFMain.dfm
UFMain.pas
UFTableList.dfm
UFTableList.pas
UTConsistency.pas
UTConversionDDL.pas
UTConversionDML.pas
UTLog.pas
UTParameter.pas
UTUtil.pas
```

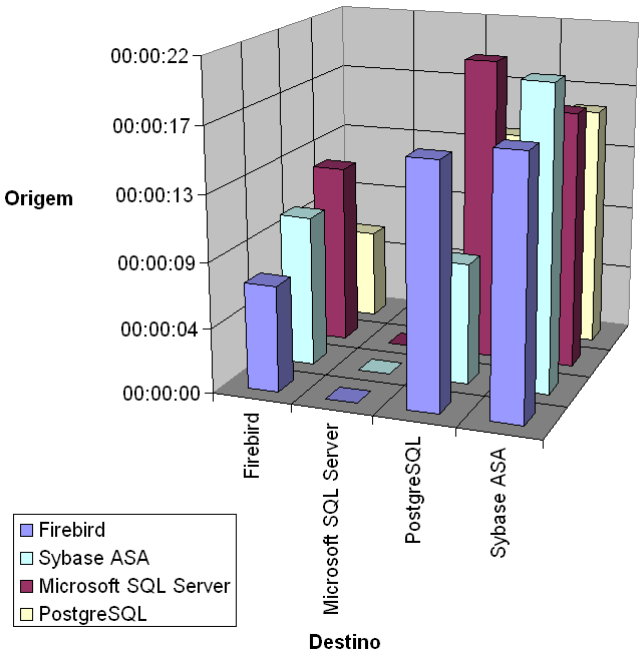
APÊNDICE G – GRÁFICOS INDIVIDUAIS DOS TEMPOS DE CONVERSÃO ENTRE OS BANCOS DE DADOS



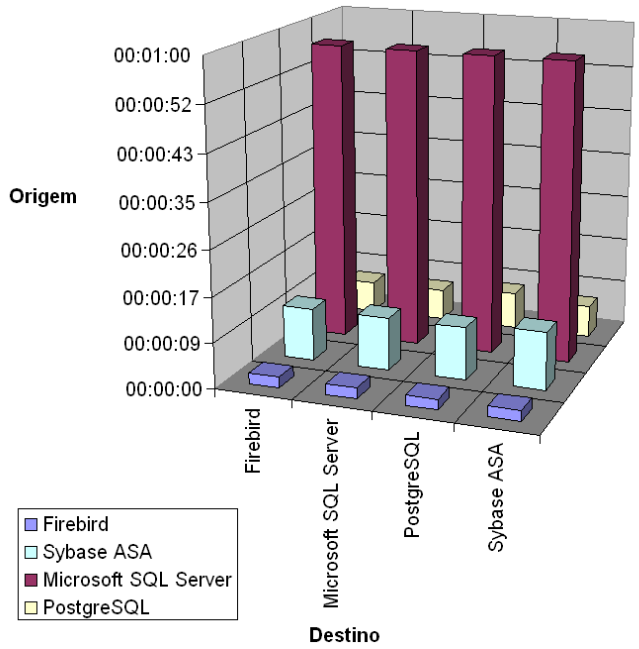
Tempos de conversão de tabelas.



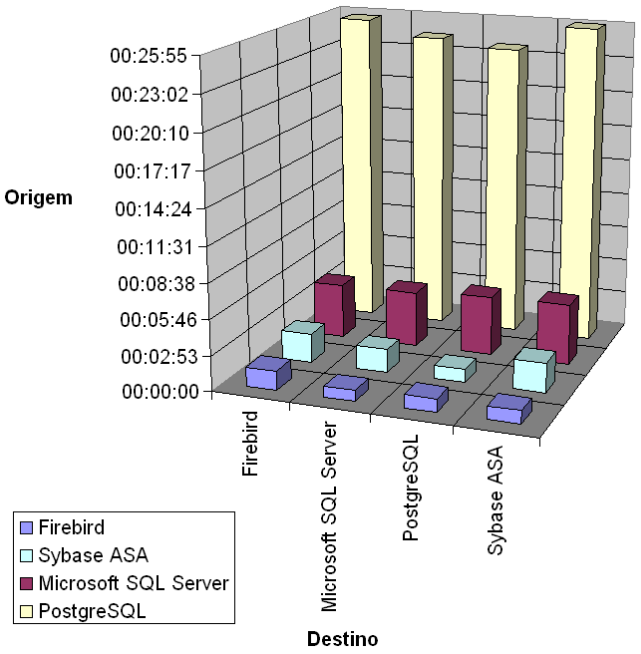
Tempos de conversão de dados.



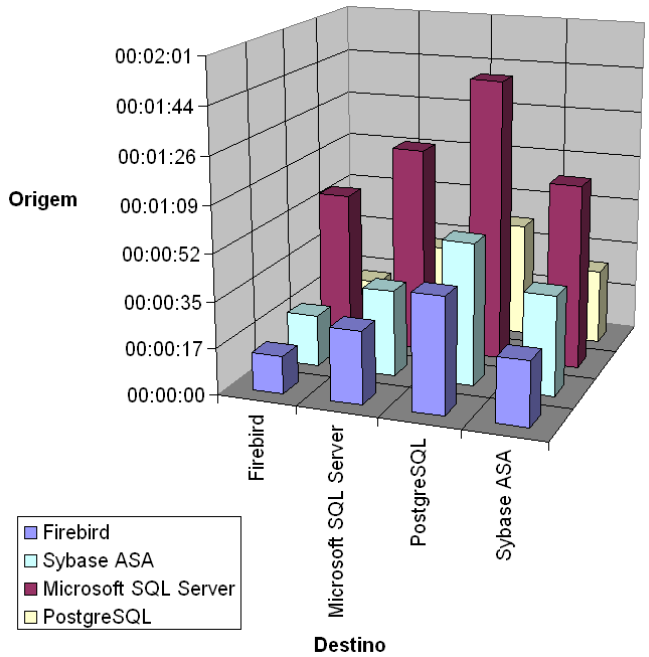
Tempos de conversão de colunas auto-incremento.



Tempos de conversão de chaves únicas.



Tempos de conversão de chaves estrangeiras.



Tempos de conversão de índices.

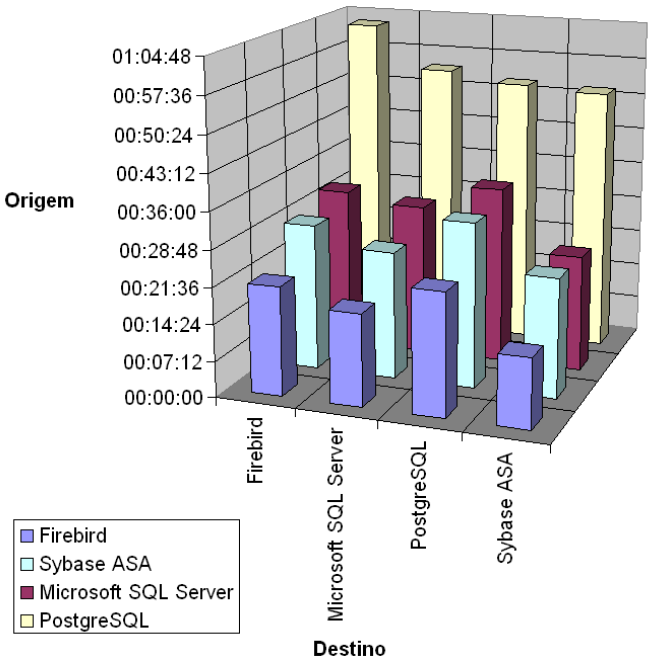


Gráfico de tempos totais de conversão.