EVA/3 Universal Database Converter

Pedra de Rosetta

Os bancos de dados modernos "falam" uma única linguagem, a SQL – porém, ela possui inúmeros "sotaques", e cada fabricante a pronuncia de um jeito. Para acabar com a Babel, que tal um conversor que promete traduzir entre 13 dialetos? Uai, tchê! Danado de bom, óxente!

Por Jens-Christoph Brendel

Quem quiser comprar mandioca no norte do Brasil deve se lembrar de pedir uma *macaxeira*. Em Curitiba um cachorro-quente tem duas *vinas*, e não salsichas. Dependendo da região, a mesma fruta pode ser conhecida como tangerina, *mexerica*, *bergamota* ou *ponkan*. A confusão causada por dialetos é semelhante também na tecnologia. Por exemplo, há um tipo de dado SQL que tem capacidade de armazenar grandes quantidades de texto. Mas, ó dia, ó vida, ó azar, ele é chamado de `Long` nos bancos de dados *Oracle*, `Memo` no *Microsoft Access*, `Text` no *PostgreSQL* e `Longtext` no *MySQL*. Quando eu chuto o balde no fim do dia ninguém sabe porquê…

A confusão dos bancos de dados

O objetivo da SQL (*Structured Query Language* ou linguagem estruturada para consultas), o esperanto dos bancos de dados, era eliminar problemas de comunicação. Quase todos os fabricantes implementaram essa linguagem em seus produtos. Mas o que era para ser motivo de júbilo virou um dantesco pandemônio: há diversas versões da mesma linguagem (SQL-89, SQL-92, SQL-99, SQL-2003) e cada fabricante usou a que bem entendeu. Como se não bastasse, a implementação da linguagem não é completa, indo somente até um certo grau que varia de produto para produto. Além disso (já não temos problemas demais?) cada fabricante mantém seus próprios dialetos.

O padrão SQL foi especialmente modulado para dar uma colher de chá aos fabricantes, uma vez que, depois do lançamento da terceira geração em 1999, nenhum banco de dados existente sequer era compatível com o SQL-92, seu antecessor. Havia três níveis de compatibilidade para o SQL-92, mas nenhum dos produtos do mercado conseguia ser compatível com pelo menos metade de um dos níveis – e o padrão já tinha sete anos! Desde então as regras da SQL mudaram, Agora, quem implementar um pacote de atributos essenciais pode fazer o que bem entender com o resto das funções, implementando-as à sua maneira. Além disso, existem diferenças específicas entre versões e particularidades dos drivers – só no padrão JDBC já existem três gerações e quatro classificações de tipos.

Isso tudo não importa para o usuário enquanto ele trabalhar com apenas um sistema. Mas se ele estiver desenvolvendo para diversas plataformas – por exemplo, quiser fornecer relatórios do banco de dados central Oracle para os notebooks dos representantes com MySQL, ou então quiser consolidar uma grande quantidade de bancos de dados existentes num sistema central – ele encontrará uma barreira lingüística. Necessitará então de um tradutor para bancos de dados.

É exatamente essa a finalidade do recém-lançado conversor de dados *Eva/3 Universal Database Converter* (UDC) da empresa Optadata [1]. Ele trabalha com 13 dialetos SQL, do *DB2* até o *Sybase*, passando pelo MySQL. Segundo a empresa, com um simples toque é possível nomear ou converter dados entre todos esses dialetos. Testamos o produto com

três bancos de dados, o Oracle 10g, o PostgreSQL 8 e o MySQL 4.0.18, para saber o quanto o conversor realmente é útil para o administrador.

Problemas na largada

A instalação em um servidor SUSE LINUX Enterprise (SLES 9) não foi totalmente sem problemas. Inicialmente a versão da máquina virtual Java da IBM fornecida pelo SUSE não pôde ser utilizada porque o instalador não localizava uma determinada biblioteca de criptografia. Tudo funcionou após a mudança para a JDK 1.4.2 da Sun, que teve de ser instalada à parte.

Após uma tela de abertura e o contrato de licença, com o qual você deve concordar, o instalador exibe um sumário as opções pré-definidas, como tipo da instalação e diretórios usados. Entretanto, não há como alterá-las. O botão *Voltar* o leva às telas anteriores.

Aceitando todos os caminhos e ajustes pré-definidos, o programa será instalado e estará pronto para operar em apenas alguns minutos. Após a primeira inicialização o usuário vai precisar dos drivers JDBC necessários para ajustar as conexões com seus bancos de dados. A função de ajuda online indica, se necessário, onde encontrar os drivers na Internet. A janela com os parâmetros dos bancos de dados é auto-explicativa. Feito isso o programa estará pronto para pôr mãos à obra.

Sem filtro

Todos os bancos de dados, como também uma seleção dos objetos correspondentes, são visualizados em uma lista que pode ser aberta na janela principal do programa, o que pode fazer com que se perca a visão geral, especialmente quando um grande número de tabelas e índices em diferentes bancos de dados obriga à rolagem freqüente da tela. Em tal caso seria adequado uma janela para cada banco de dados e a possibilidade de filtrar os objetos mostrados. O filtro seria útil também porque o conversor mostra todas as tabelas como, por exemplo, as excluídas que estão na lixeira do Oracle e que certamente não devem ser convertidas (**figura 1**).

Na lista mencionada, além de tabelas, chaves e índices, também aparecem os *views* (como são chamadas as visualizações de dados no Oracle). Quem por isso chegar à conclusão de que os views também podem ser transferidos entre os bancos de dados estará redondamente enganado. Nem arrastando e soltando com o mouse, nem combinando outras tabelas, a transferência funcionará. Por isso, o motivo dos views serem apresentados continua sem explicação. Na ajuda online as palavras *view* e *visualização* simplesmente não aparecem.

O simples funciona...

Em um primeiro teste o UDC teve que converter uma grande, mas simples, tabela de valores de medições contendo mais de 10.000 registros do MySQL para o Oracle e para o PostgreSQL. Essa operação funcionou sem problemas, rápida e corretamente. Índices e chaves primárias existentes também foram criados novamente no sistema alvo (os *constraints*, porém, serão perdidos). Para isso bastou puxar a tabela da lista do banco de dados fonte para a parte correspondente do banco de dados de destino.

Se o conversor não encontrar nenhum problema, transfere os dados sem qualquer pergunta. Caso contrário, oferece uma caixa de diálogo para a adaptação dos tipos de dados. Mas ele é ladino: inclui apenas as opções que lhe parecem adequadas. Como era de se esperar, isso pode trazer problemas: o administrador pode eliminar o limite de certos tipos de dados e definir o tipo alvo, mas não pode esperar que o conversor saiba como lidar com as diferenças.

Drible inteligente

O tradutor foi muito esperto na conversão de uma tabela do PostgreSQL que continha uma coluna do tipo `serial`. O Oracle não conhece nenhum campo desse tipo que se auto-incrementa. Por isso, no Oracle foi criado um script com um laço (*loop*) e uma seqüência que imitam o comportamento desta função.

Ficou até bom, mas o melhor teria sido informar a inconsistência de dados ao administrador, dando-lhe assim a possibilidade de fazer a manutenção desses novos objetos ou de excluí-los quando necessário. Além disso, como conseqüência acontecia um efeito colateral: a con-

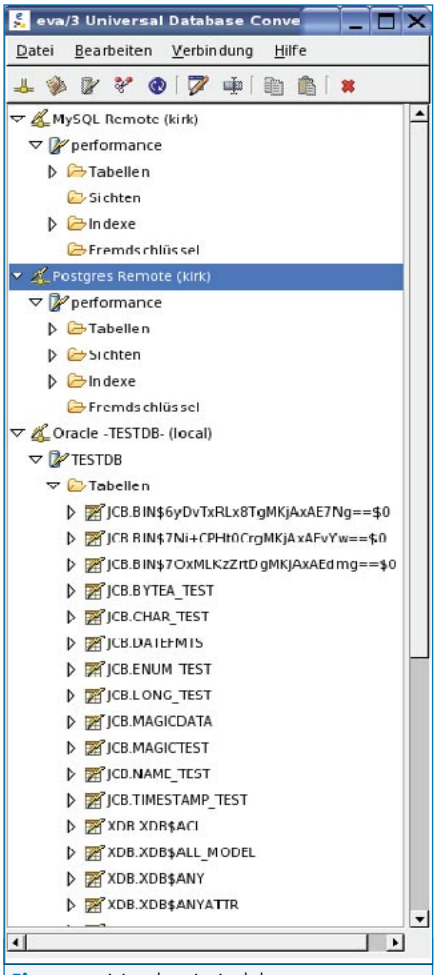

Figura 1: A janela principal do conversor mostra, sem diferenciação, todos os objetos de uma determinada categoria, inclusive objetos que já foram excluídos ou que não podem ser convertidos.

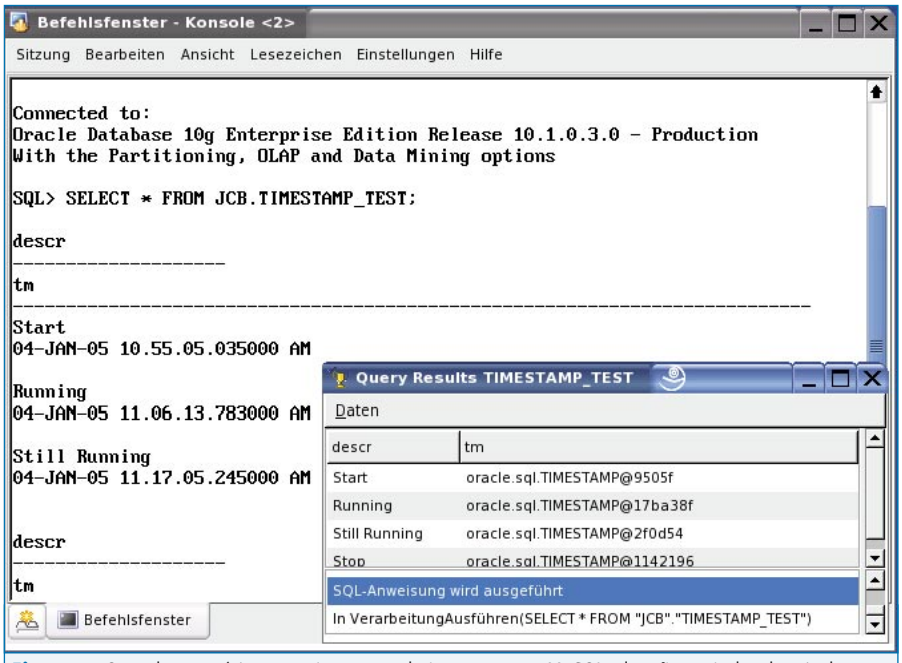

Figura 2: Quando o usuário converte marcas de tempo para o MySQL, elas são cortadas depois do segundo inteiro. Não há nenhum aviso sobre a perda de precisão.

versão múltipla da mesma tabela falhou porque a mensagem de erro avisava que os objetos a serem criados já existiam, mesmo quando haviam sido excluídos. Foi através deste erro que ficou evidente o truque mencionado, que nem deveria estar escondido.

Falsos cognatos

Gradativamente o grau de dificuldade foi aumentando. Em um determinado momento, o tradutor foi enganado e falhou em um problema que normalmente é um caso de lingüística, os falsos cognatos. *Eventually* não é o mesmo que *eventualmente*, como também *actually* não é atualmente, mesmo que pareçam semelhantes. Da mesma forma, o tipo `char` do Oracle não é o mesmo que o tipo `char` do MySQL, que tem capacidade para apenas 255 caracteres.

Ao tentar converter uma tabela com uma coluna `Char(2000)` do Oracle para o MySQL o conversor nada alertará. Em vez disso, continuará recebendo dados até que a seguinte mensagem do MySQL apareça na tela:

```
Too big column length for column
'CHARACTERES' (max=255). Use BLOB instead
```

Tarde demais, portanto. Note que a mensagem é do MySQL e não do conversor. Infelizmente não é possível seguir a sugestão de usar o BLOB (*Binary Large Object*), pois devido à aparente equivalência o UDC não oferece a escolha manual do tipo de dados alvo.

O UDC oferece o tipo `varchar2` do Oracle como correspondente do tipo `text` no PostgreSQL, mesmo que esse tipo também possa estourar. Afinal, o tipo `text` no PostgreSQL – que, além disso, só existe a partir da versão 7.1 – é limitado a 4.000 caracteres. Ao converter uma coluna `text` para o MySQL, o conversor usará o tipo com o mesmo nome, limitado neste caso a 64 kBytes, diminuindo o risco de estouro. Mas nesse caso ele se confunde na sintaxe SQL, de forma que o banco de dados fica sem entender mais nada:

```
CREATE TABLE 'text_test' ('text' text(2
-1) NULL)
```

O comando anterior é muito complicado e, mesmo funcionando no Oracle, provoca um erro de sintaxe no MySQL. Para o banco de dados sueco, bastaria um

```
CREATE TABLE text_test (text text)
```

O comando é até mais simples, mas o conversor não sabe disso.

Tempo perdido

A diferença de precisão provoca um problema semelhante ao da diferença de capacidades de tipos de dados com o mesmo nome. Por exemplo, as marcas de tempo `TIMESTAMP` do MySQL trabalham apenas com segundos inteiros, enquanto o PostgreSQL e o Oracle reconhecem até fragmentos de segundos nos mesmos tipos de dados. Na conversão com o Eva/3 UDC esses fragmentos são simplesmente ignorados. Em alguns casos isso pode até ser aceitável, mas no mínimo deveria haver uma mensagem sobre a perda de informações (**figura 2**).

O conversor geralmente consegue evitar com sucesso outros riscos de conflitos de nome – com diferentes palavras reservadas e objetos já existentes. Quando encontra tabelas com o mesmo nome ele oferece a opção de sobregravar a tabela alvo ou mudar o nome dela. Problemas com palavras reservadas são evitados com um eficiente processo de inserção de comentários.

Pode acontecer, entretanto, do conversor agir além do cumprimento do dever. Houve casos em que o conteúdo de tabelas do Oracle não pôde ser visualizado: na seleção correspondente, o nome do esquema – que fica logo antes do nome da tabela – foi colocado entre aspas duplas e o banco de dados não conseguia mais localizar o objeto.

Identificar falhas nas consultas de tabelas é complicado: a área que indica o estado do banco, na parte inferior da janela de resultados, ordena as mensa-

gens de modo que sempre a mais antiga esteja visível. No resultado é apresentada sempre a primeira mensagem, que já está obsoleta. Para visualizar possíveis mensagens de erro o usuário sempre tem que navegar pelas mensagens subseqüentes. Isso não é prático.

Encontros e desencontros

Vários tipos de dados não podem ser convertidos. Este é o caso das construções especiais próprias como, por exemplo, os tipos geométricos no PostgreSQL. O conversor avisa prontamente que *O tipo de dado indicado não tem suporte no banco de dados*. Para os tipos de contagem especiais do MySQL (`SET`, `ENUM`), no PostgreSQL e Oracle são oferecidos tipos de caracteres. O que é algo bastante razoável; pena que falta novamente o aviso sobre a perda de funcionalidade resultante da conversão.

Todos os tipos de dados definidos pelos usuários continuam sendo solenemente ignorados. Parece também que os contêineres binários não são suportados. Em uma amostra de um BLOB (*Binary Large Object*) do Oracle não foi encontrado um tipo alvo adequado no PostgreSQL. Mesmo com a indicação explícita de `byte` (o tipo correto), o conversor não foi além de mostrar a mensagem *Não encontrada classificação para o tipo oracle.sql.BLOB* .

Resumo

O conversor de bancos de dados Eva/3 UDC transforma uma interessante idéia em software. Entretanto, atualmente não passa de um guia turístico. Se a interface com o usuário fosse aperfeiçoada, os problemas iniciais amenizados e mais soluções fossem encontradas, com a mesma inteligência da emulação do tipo `Serial` no Oracle, ele poderia tornar-se uma ferramenta bastante útil para todos os que precisam converter freqüentemente os conteúdos de seus bancos de dados. No momento o conversor é adequado principalmente para tabelas simples com pequenas cadeias ou valores numéricos, como por exemplo livros de endereços, listas de preços e produtos ou repositórios de dados pessoais.

Por outro lado, dados binários, grandes quantidades de texto, tipos de dados definidos pelo usuário ou tipos próprios de dados são fontes de problemas em potencial. O usuário não deve confiar cegamente no conversor e deve verificar sempre os resultados. Mesmo assim, em muitos casos o conversor poupará boa parte do que seria um cansativo e tedioso trabalho manual.

De um conversor muito bom até uma ferramenta capaz de realizar a conversão completamente automatizada (e perfeita) de bancos de dados ainda há um longo caminho a percorrer. Além de dados de tabelas e índices, espera-se de um software desse calibre que também os usuários, as funções, os direitos, os laços, os procedimentos, os views e tudo mais fosse transferido. Vamos ver o que o futuro nos reserva nesta área. ■

Informações

[1] Página oficial do Eva/3 UDC: www.optadata.com/de/Eva3_udc.htm

[2] Documentação oficial do padrão SQL: webstore.ansi.org

[3] Documentação adicional sobre SQL: www.wiscorp.com/SQLStandards.html

[4] Rascunho quase final, e gratuito, da especificação SQL 2003. Arquivo ZIP, 10.5 MB: www.wiscorp.com/sql/sql_2003_standard.zip

[4] Comparativo entre as diferentes implementações do padrão SQL (documento atualizado constantemente): roels.arvin.dk/db/rdbms/

[5] História da SQL: www.mcjones.org/System_R/SQL_Reunion_95/

[6] Artigo relevante na Wikipedia: en.wikipedia.org/wiki/SQL