COMEÇAR COM

DB2 Express-C

Um livro da comunidade para a comunidade

RAUL CHONG, IAN HAKES, RAV AHUJA

PREFÁCIO POR DR. ARVIND KRISHNA

SEGUNDA EDIÇÃO

Segunda edição (Fevereiro 2008)

Esta edição aplica-se a IBM® DB2® Express-C Versão 9.5 para Linux®, UNIX® e Windows®.

© 2008 Copyright IBM Corporation. Todos os direitos reservados.

Índice

Sobre este livro

Avisos e Marcas Registadas

© Copyright IBM Corporation 2008
Todos os direitos reservados.
IBM Canada
8200 Warden Avenue
Markham, ON
L6G 1C7
Canada

Nem esta nem qualquer outra parte deste documento pode ser copiada ou reproduzida sobre qualquer forma ou por qualquer meio ou traduzido para outro idioma, sem o consentimento prévio dos autores.

A IBM não faz declarações ou garantias no que diz respeito ao conteúdo e isenta-se especificamente de quaisquer garantias implícitas de comercialização ou adequação para qualquer finalidade específica. A IBM não assume nenhuma responsabilidade por quaisquer erros que possam aparecer neste documento. A informação contida neste documento está sujeita a alterações sem aviso prévio. À IBM reserva-se o direito de fazer essas mudanças sem qualquer obrigação de notificar qualquer pessoa ou revisão de tais mudanças. A IBM não faz qualquer compromisso de manter as informações nele contidas actualizadas.

A informação contida neste documento sobre produtos não IBM foram obtidas a partir do(s) fornecedor(es) de tais produtos. A IBM não testou esses produtos e não pode confirmar a precisão dos resultados, compatibilidade ou quaisquer outras reivindicações relacionadas a produtos não IBM. Perguntas sobre as capacidades dos produtos não IBM deverão ser endereçadas ao(s) fornecedor(es) de tais produtos.

A IBM, o logotipo IBM, DB2, DB2 Connect, DB2 Universal Database, i5/OS, pureXML, WebSphere, e z/OS são marcas comerciais ou marcas registadas da International Business Machines Corporation nos Estados Unidos, em outros países, ou ambos.

Java e todas as marcas baseadas em Java são marcas registadas da Sun Microsystems, Inc. nos Estados Unidos, em outros países, ou ambos.

Microsoft e Windows, são marcas registadas da Microsoft Corporation nos Estados Unidos, em outros países, ou ambos.

Linux é uma marca registada de Linus Torvalds nos Estados Unidos, em outros países, ou ambos.

Outras empresas, produtos, serviços ou nomes podem ser marcas comerciais ou marcas de serviço de terceiros.

As referências feitas nesta publicação a produtos ou serviços IBM não significam que a IBM tem a intenção de torná-los disponíveis em todos os países nos quais a IBM opera.

Quem deve ler este livro?

Este livro é destinado a qualquer pessoa que trabalha, ou pretenda trabalhar com bases de dados, tais como administradores de bases de dados (DBAs), programadores, consultores, arquitectos de software, gestores, instrutores e estudantes.

Como está estruturado este livro?

A Parte I – Visão Geral e Configuração, explica o que é a edição DB2 Express-C, introduz a família de produtos e recursos DB2, auxilia a instalação e a criação de bases de dados, e explora as ferramentas disponíveis no DB2.

A Parte II – Aprender DB2: Administração de Base de Dados destina-se a familiarizá-lo com o ambiente DB2, arquitectura, conectividade remota, bases de dados de objectos, movimentação de dados (importação / exportação), segurança, *backup* e recuperação, concurrência e *locking*, manutenção e outras tarefas comuns.

A , abrange procedimentos armazenados (*stored procedures*), funções definidas pelo utilizador (*user defined functions*), *triggers*, SQL/XML, XQuery, desenvolvimento em Java ™, PHP e Ruby.

O apêndice contém informações úteis sobre resolução de problemas.

Exercícios chamados "Quicklabs" são fornecidos na maioria dos capítulos; quaisquer eventuais ficheiros de input necessários para estes exercícios são fornecidos no arquivo zip **`expressc_book_quicklabs_9.5.zip`** que acompanha este livro, ou disponível no Web site IBM DB2 Express-C ®: www.ibm.com/db2/express.

Todo o material usado neste livro também é usado em cursos oferecidos no âmbito do programa "*DB2 on Campus*", e é bastante semelhante aos vídeos disponíveis em www.-channelDB2.com/oncampus. Este livro também pode ajuda-lo a preparar-se para o exame *DB2 on Campus*. Este exame proporciona-lhe uma confirmação de conclusão do programa que reconhece que recebeu 16 horas de formação de DB2. Pode ler mais sobre este programa, no site DB2 Express-C www.ibm.com/db2/express/students.html.

Nota:

Para mais informações sobre o programa *DB2 on Campus* veja este video: http://www.channeldb2.com/video/video/show?id=807741:Video:3902

.

Um livro da comunidade para a comunidade

Este livro foi criado pela equipa DB2 Express-C e distribuído para a comunidade DB2 Express-C sem qualquer custo. Muitos membros da comunidade de todo o mundo traduziram este livro para várias línguas. Se quiser dar *feedback*, contribuir com novo material, melho-

rar material já existente, ou ajudar a traduzir este livro para outra língua, envie um e-mail do plano do seu contributo db2x@ca.ibm.com com o assunto "DB2 Express-C book changes. "

Autores e Contribuidores

As seguintes pessoas forneceram conteúdo e outros contributos significativos para este livro.

Nome do Contribuidor	Empresa	Emprego	Material contribuido	Data
Raul F. Chong	IBM	DB2 on Campus Program Manager	Versão inicial de todos os capítulos do livro para a 1 ª e 2 ª edições	Fev 2008
Ian Hakes	IBM	DB2 Express-C Community Facilitator	Revisão e edição da 1 ª e 2 ª edições do livro	Fev 2008
Rav Ahuja	IBM	DB2 Product Manager	Revisão, actualização, edição, *layout* e formatação de todo o livro	Fev 2008

Agradecimentos

Estamos muito gratos às seguintes pessoas pela sua assistência e desenvolvimento de materiais referenciados neste livro:

- Ted Wasserman, Clara Liu e Paul Yip do laboratório da IBM Toronto que desenvolveram materiais que serviram como base para este livro.
- Don Chamberlin e Cindy Saracco pelos seus artigos no site IBM *developerWorks* sobre XQuery, e Matthias Nicola pelas suas apresentações sobre pureXML™.
- Kevin Czap e Grant Hutchison por desenvolverem material técnico informativo sobre DB2.
- Katherine Boyachok pela realização da capa deste livro.
- Susan Visser pela ajuda na publicação deste livro.

Tradutores

Este livro foi traduzido por vários estudantes portugueses durante a sua participação no programa *DB2 Ambassador* da IBM. Marcelo Sousa, da Universidade do Minho foi o líder deste projecto, responsável pelo planeamento, distribuição e coordenação de tarefas e revisão final.
Eis os nomes dos tradutores e respectivas contribuições:

- **Marcelo Sousa** – capítulos 7, 8, 9, 10, 14, 15 e 16.
- **Ulisses Costa** – capítulos 1, 4, 6, 13, 17 e Apêndice.
- **Tiago Pregueiro** – capítulos 3, 5, 11 e 12
- **Sérgio Vasconcelos** – capítulo 2

O processo de revisão foi efectuado por:

- *Revisão linguística*: **Joana Job**
- *Revisão geral*: **Marcelo Sousa**
- *Revisão técnica*: **Vitor Rodrigues**

Prefácio

A inovação é a pedra angular no progresso da tecnologia. Na IBM, a inovação tem sido parte integrante da evolução dos nossos servidores de dados. Tendo sido pioneiros na década de 1960 e 1970 em técnicas de gestão de dados, temos entregue continuamente tecnologias inovadoras de gestão da informação, reflectida nas milhares de patentes de gestão de dados da autoria de técnicos da IBM. Como resultado, algumas das maiores organizações do mundo de hoje dependem de produtos da IBM como o DB2 para suportar as suas mais exigentes e critícas soluções de gestão dos dados.

No entanto, o DB2 já não é apenas para as grandes empresas. Com o lançamento do DB2 Express-C, a tecnologia DB2 vencedora de vários prémios já está disponível para responder às necessidades das pequenas e médias empresas - e sem nenhum custo obrigatório! Embora existam, por aí, outros servidores de dados gratuitos e open-source, o DB2 Express-C oferece vantagens únicas quando comparado com estas alternativas.

Há muitas inovações tecnológicas presentes no DB2 Express-C. Algumas destas inovações são destinadas a novas funcionalidades mais específicas, algumas à redução dos encargos administrativos, algumas a melhorar o desempenho, e algumas a reduzir o custo de infra-estruturas. Não vamos discutir a maior parte destas aqui, esperando que se sinta tentado a ler o livro - mas vamos descrever uma delas como exemplo.

O DB2 Express-C é construído sobre a tecnologia “Viper”, tornando-se o primeiro servidor híbrido de dados relacionais e de dados XML nos seus formatos nativos. Isto torna o DB2 ideal para a sustentação de uma nova classe de aplicações SOA e Web 2.0 onde existem dados XML em abundância. Ao contrário de outros servidores de dados comerciais, o DB2 Express-C não limita a quantidade de dados que pode armazenar numa base de dados ou o número de bases de dados que pode criar num sistema. E, claro, se precisar de ajuda ou assistência da IBM, a ajuda está a um clique de distância.

Este livro serve como guia para começar a usar o DB2 Express-C. O livro irá ajudá-lo a compreender os conceitos do DB2 e permitir-lhe desenvolver habilidades para administração e desenvolvimento de aplicações DB2. As competências e conhecimentos adquiridos com a ajuda deste livro são muito importantes para as outras edições avançadas do DB2.

Embora o DB2 Express-C não seja open-source, na IBM acreditamos muito no apoio e fomento de iniciativas da comunidade. Estou muito satisfeito por este livro ser desenvolvido pelos membros da comunidade DB2 Express-C e a tornar-se disponível gratuitamente para qualquer pessoa na comunidade. Gostaria tambem de vos encorajar a enriquecer e actualizar este livro com o vosso *know-how*, experiências e ajudar a traduzir este livro para outras línguas, para assim outras pessoas poderem beneficiar dos vossos conhecimentos.

Arvind Krishna
Vice Presidente, Data Servers
Information Management, IBM Software Group

Parte I – Visão Geral e Configuração

1

Capítulo 1 - O que é o DB2 Express-C?

O DB2 Express-C é um membro da família IBM DB2, poderoso software de servidor de dados para gestão tanto de dados relacionais como de dados XML. O DB2 Express-C é uma edição livre, sem limites, e fácil de usar. O "C" no nome DB2 Express-C significa Comunidade. Uma comunidade de utilizadores de DB2 Express-C que juntos se ajudam uns aos outros, tanto on-line como off-line. A comunidade do DB2 Express-C é constituída por todos os tipos de pessoas e empresas que concebem, desenvolvem, implementam ou utilizam soluções de bases de dados, tais como:

- Programadores de aplicações que necessitam de uma base de dados de padrões abertos (*open standards database)* para a construção de software *stand-alone*, cliente-servidor, *web-based*, e aplicações empresariais
- ISVs (Independent Software Vendors), fornecedores de *hardware*, vendedores de infra-estruturas, e outros tipos de fornecedores de soluções que desejam incluir um servidor de dados como parte das suas soluções
- Consultores, administradores de bases de dados, arquitectos das TI que precisam de um servidor de dados robusto para formação, desenvolvimento de qualidades técnicas, avaliação e prototipagem
- Empresas em fases iniciais assim como as pequenas e médias empresas que necessitam de um servidor de dados fiável para as suas aplicações e operações
- Interessados por bases de dados e entusiastas de tecnologia de ponta que querem um servidor de dados fácil de usar para construir aplicações da próxima geração e Web 2.0
- Estudantes, professores, e outros utilizadores académicos que querem um servidor de dados altamente versátil para o ensino, cursos, projectos e pesquisa

O DB2 Express-C partilha a mesma funcionalidade e código base que as outras edições do DB2 pagas para Linux, UNIX e Windows. O DB2 Express-C pode ser executado em sistemas operatibos Linux ou Windows, quer em modo 32-bit ou 64-bit. Está optimizado para sistemas que tenham até 2 processadores e 2GB de memória para a versão gratuita, e até 4 processadores e 4 GB de memória para a versão de 12 meses de licença com suporte.
O DB2 Express-C também inclui pureXML sem qualquer encargo financeiro. pureXML é uma tecnologia exclusiva do DB2 para armazenar e processar nativamente documentos XML.

1.1 Livre para desenvolver, implementar e distribuir… sem limites!

Esta secção resume as principais ideias do DB2 Express-C:

- **Livre para desenvolver**: Se é um programador e necessita de uma base de dados para a sua aplicação, pode usar DB2 Express-C.

- **Livre para implementar**: Se está a trabalhar num ambiente de produção e precisa de uma base de dados para armazenar a sua informação vital, pode usar DB2 Express-C.

- **Livre para distribuir**: Se estiver a desenvolver uma aplicação ou uma ferramenta que requer um servidor de dados embebido, pode usar DB2 Express-C. Apesar de DB2 Express-C ser incorporado na sua aplicação, e distribuído cada vez que vender a sua aplicação, ainda é livre. Você é obrigado a registar-se com a IBM, a fim de redistribuir o DB2 Express-C, porém este registo também é gratuito.

- **Sem limites**: Enquanto outras bases de dados concorrentes oferecem limites fixos de tamanho da base de dados, com o DB2 Express-C, NÃO existem limites. A sua base de dados pode crescer e crescer sem violar o acordo da licença. Também não há nenhum limite em termos de número de conexões por servidor ou utilizadores.

Nota:

Para aprender mais sobre DB2 Express-C e o seu papel no mundo da informação *on-demand* e da Web 2.0, veja este video de apresentação:
http://www.channeldb2.com/video/video/show?id=807741:Video:3922

1.2 Suporte técnico e assistência ao utilizador

Se tem dúvidas técnicas sobre o DB2 Express-C, pode deixar as suas perguntas no fórum do DB2 Express-C. Este forum livre é acompanhado por uma equipa do DB2 Express-C, porém, é a comunidade que fornece a maior parte das respostas de uma forma voluntária. A IBM também oferece aos utilizadores a opção de comprar uma subscrição anual de baixo custo (também conhecida como *12 Months License* e *Fixed Term License* ou *FTL*). Esta subscrição para o DB2 Express-C vem com o apoio técnico 24x7 da IBM e actualizações de *software*. Por um preço anual e renovável de baixo custo (US $2995 por servidor por ano, nos Estados Unidos - pode variar noutros outros países), o utilizador não só pode obter apoio e manutenção de *software* para o seu servidor DB2 Express-C, mas também acesso a duas *features*: HADR (*High Availability Disaster Recovery*) – Recuperação de Desastres de Alta Disponibilidade, e replicação SQL (para replicar dados com outros servidores DB2).
Mais informação sobre a subscrição DB2 Express-C pode ser encontrada em:
http://www-306.ibm.com/software/data/db2/express/support.html

1.3 Servidores DB2

Todas as edições do servidor DB2 contêm os mesmos componentes essenciais; eles são “empacotados” para que os utilizadores possam escolher as funções que precisem ao preço certo. A figura 1.1 ilustra as diferentes edições do produto DB2.

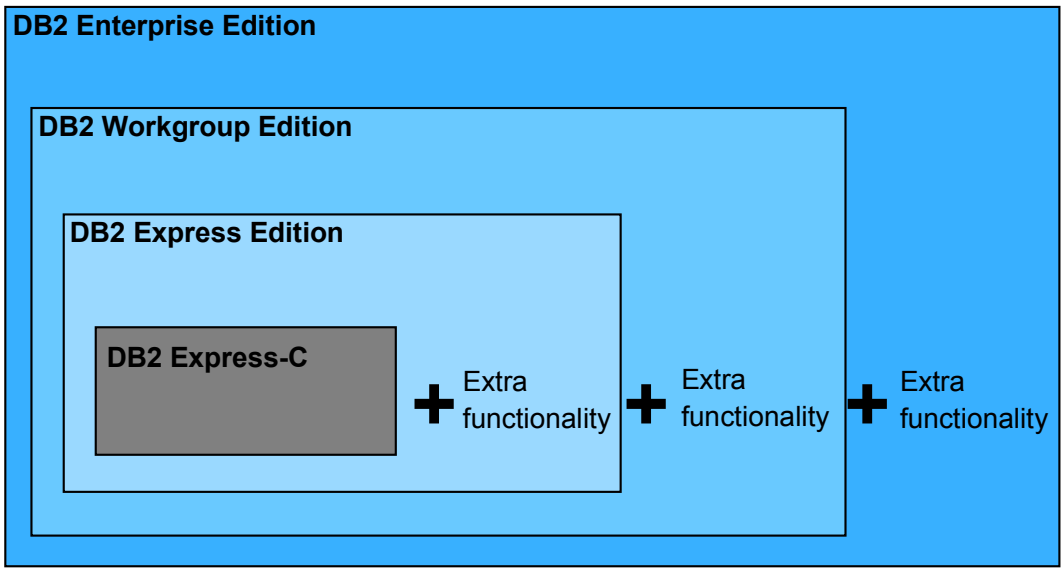


Figura 1.1 – Servidores DB2

Como se pode ver na Figura 1.1, o DB2 Express-C é o mesmo que o DB2 Express mas sem alguns componentes. O DB2 Express-C é gratuito para a comunidade. Como mencionado anteriormente, a assistência técnica está disponível gratuitamente *on-line* através de um fórum, podendo também receber oficialmente suporte técnico DB2 24x7 se comprar a licença dos 12 meses de subscrição.

A Figura 1.1 explica também por que razão é tão fácil actualizar a partir de DB2 Express-C. Se, no futuro, desejar actualizar para qualquer um dos outros servidores DB2, todos os servidores DB2 têm o mesmo núcleo de componentes. Isto também significa que qualquer aplicação desenvolvida para uma edição irá executar, sem qualquer modificação, nas outras edições. E quaisquer conhecimento técnico que adquira numa edição poderá ser aplicada às outras edições.

1.4 Clientes e *drivers* DB2

Através de um cliente DB2 podemos ligar-nos a um servidor DB2; no entanto, um cliente DB2 nem sempre precisa de ser instalado. Por exemplo, uma aplicação JDBC Tipo 4 pode ligar directamente a um servidor DB2, desde que o driver correcto esteja carregado. Os clientes e drivers DB2 existem em vários formatos:

- IBM Data Server Client: mais completo, inclui ferramentas GUI, *drivers.*
- IBM Data Server Runtime Client: um cliente simples com funcionalidades básicas, e inclui *drivers.*
  DB2 Runtime Client Merge Modules para Windows: utilizados principalmente para embeber o *DB2 Runtime Client* em aplicações Windows.
- IBM Data Server Driver para JDBC e SQLJ: permite que aplicações Java se liguem a servidores DB2 sem a instalação de um cliente completo.

- IBM Data Server Driver para ODBC e CLI: permite que aplicações ODBC e CLI se liguem a servidores DB2 sem a instalação de um cliente.
- IBM Data Server Driver para ODBC, CLI e .NET: um *driver* específico para Windows para suportar ambientes .NET com ODBC e CLI

A Figura 1.2 mostra os diferentes clientes DB2 e os *drivers* disponíveis.

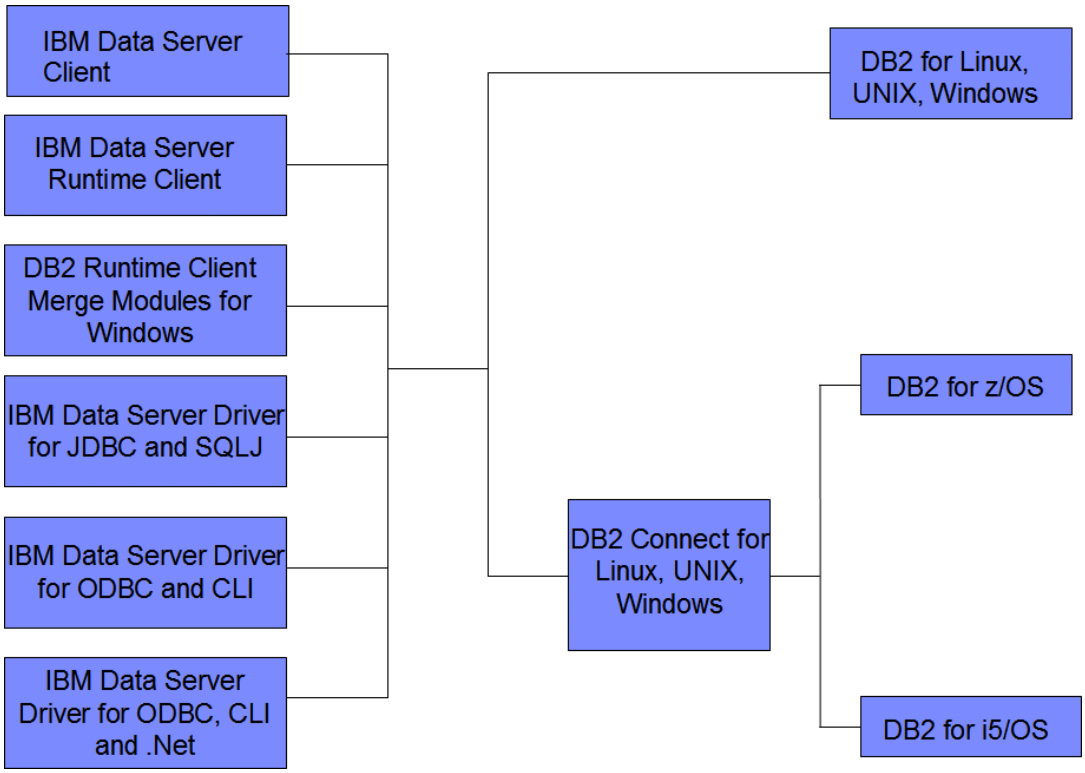


Figura 1.2 – Clientes DB2 e *drivers*

No lado esquerdo da Figura 1.2, são mostrados todos os *drivers* e clientes DB2. Embora todos os clientes incluam os *drivers* necessários, a partir do DB2 9 também fornecemos os *drivers* individualmente. Clientes DB2 e todos os *drivers* são gratuitos e estão disponíveis para download a partir do *website* DB2 Express-C. Os clientes e os *drivers* podem ser usados para estabelecer comunicacão com um servidor DB2 em Linux, UNIX ou Windows. Para se ligar a um servidor DB2 para z / OS® ou DB2 para i5/OS®, terá que passar por um *DB2 Connect Server* (meio da Figura 1.2). Iremos discutir o *software DB2 Connect* no Capítulo 2.

1.5 Liberdade de desenvolvimento de aplicações

O DB2 oferece um ambiente de desenvolvimento de aplicações baseadas em *standards* e é transparente em toda a família DB2. A padronização do SQL em toda a linha de produtos DB2 proporciona um conjunto comum de interfaces de programação de aplicação de acesso a bases de dados.
Além disso, cada produto DB2 oferece pré-compiladores de SQL que permite aos progra-

madores embebir SQL estático e dinâmico em programas portáteis. O DB2 possui ainda um gestor .NET nativo e integração com as ferramentas do Microsoft® Visual Studio.

Linguagens e *standards* que pode usar com DB2 são:

- SQL, XQuery, XPath
- Ruby on Rails (RoR)
- C/C++ (CLI, ODBC e SQL embebido)
- Java (JDBC e SQLJ)
- COBOL
- Python
- PHP
- Perl
- .NET languages
- OLE-DB
- ADO
- Web services
- SQL
- MS Office: Excel, Access, Word

1.6 Versões do DB2 vs edições do DB2

Se é novo no DB2, pode ser um pouco confuso distinguir entre uma versão e uma edição de DB2.

De poucos em poucos anos, a IBM lança uma nova versão de DB2. Uma versão inclui novas funcionalidades e melhorias significativas do produto. Actualmente, as versões de DB2 8 e 9 são oficialmente suportadas pela IBM. Uma versão também pode ter algumas *releases*, o que pode incluir novas funcionalidades, mas geralmente não suficientemente significativas para justificar a necessidade de uma nova versão. Por exemplo, 8.1 e 8.2 são níveis de *releases* para a versão DB2 8. No passado, a IBM tem lançado uma nova versão do DB2 quase todos os anos, no entanto as novas versões são separadas por 2-3 anos de intervalo. A versão mais actual é a V9.5 (anteriormente sob o nome de código de DB2 "Viper 2 ') e tornou-se *Generally Available* (GA) em Outubro de 2007. Cada lançamento pode ter também vários níveis de *Modification*, que normalmente contêm correcções ou correspondem a packs de correcções, e raramente incluem novas funcionalidades. No momento de escrita deste livro o nível do DB2 Express-C*Version, Release, Modification* (V,R,M) é 9.5.0, o que corresponde a um código 9.5 com um fix pack 0, o que significa que é um nível GA.

Por outro lado, as edições são agrupadas num pacote a cada versão. Conforme discutido anteriormente, uma edição é uma embalagem de funções diferentes para um determinado preço e licença. A Versão 9.5 de DB2 (também conhecida como DB2 9.5) tem várias edições, por exemplo, DB2 Express-C 9.5, 9.5 Express DB2, DB2 Workgroup 9.5, e DB2 Enterprise 9.5 (ver Figura 1.1).

1.7 Mudança para outra edição DB2 superior

Tal como a sua base de dados precisa de crescer, pode ser necessário fazer uma actualização para uma edição de DB2 que suporte uma maior configuração de hardware. Se esta situação se verificar, é fácil fazer a actualização para outra edição DB2:

- Se está a actualizar para outra edição de DB2 no mesmo sistema, instale a nova edição de DB2 em cima do DB2 Express-C, e a nova licença correspondente. Os seus dados não serão danificados (mas uma cópia de segurança é sempre recomendável)
- Se está a actualizar o DB2 onde a nova edição vai ser instalada num computador diferente e maior, usando o mesmo sistema operativo, instale a nova edição de DB2 num computador maior, faça *backup* dos seus dados a partir do computador mais pequeno, mova as imagens de *backup* para o computador maior, e restaure a partir da imagem de *backup* a bases de dados que estava no computador maior. Também pode precisar salvar as definições de configuração da instância (dbm cfg) a partir do seu computador mais pequeno, e aplicar esta configuração para o computador maior. Os comandos de *backup* e restauração serão discutidos em detalhe no Capítulo 11 – Cópias de Segurança e Recuperação. O *dbm cfg* será discutido em maior detalhe no Capítulo 5 – Ferramentas DB2.
- Em ambos os casos a sua aplicação não irá necessitar qualquer alteração.

1.8 Manutenção do DB2 Express-C

Conforme discutido anteriormente, existem duas opções para o suporte do DB2 Express-C:

- Comprar uma licença de subscrição por 12 meses. Isto proporciona-lhe suporte técnico da IBM DB2 a tempo inteiro e dá-lhe a capacidade de instalar actualizações de *software* DB2 (também chamados de *fixpacks*).

- Usar o fórum *online* da comunidade DB2 Express-C. Isto é totalmente gratuito, mas vem sem o apoio oficial da IBM. Além disso, no âmbito desta opção, a IBM não se compromete a fornecer novos recursos e correções de *bugs* em datas programadas. O conceito de um *fixpack*, que é discutido no Capítulo 2, não se aplica; em vez disso, actualizações por completo à imagem do DB2 Express-C são disponibilizadas ao longo do tempo. Com a saída de novas versões, pode-se esperar que estejam disponíveis novas imagens do DB2 Express-C mais para os novos lançamentos do que para os mais antigos.

1.9 Software livre relacionado

Todo o software que está disponível para *download* a partir da página de DB2 Express-C (www.ibm.com/db2/express/download.html) é gratuito. Além das imagens para o DB2 Express-C (para Linux e Windows, tanto para arquitecturas de 32 e 64-bit), há outros programas úteis que podem ser obtidos e utilizados gratuitamente:

- IBM Data Studio
- DB2 9 Embedded Application Server
- DB2 9.5 Net Search Extender

Há também ferramentas para iniciantes (*starter toolkits*) adicionais baseadas em DB2 Express-C, disponíveis para *download* a partir do *website* da IBM Alphaworks (www.alphaworks.ibm.com/datamgmt) que podem ser úteis:

- Starter Toolkit for DB2 on Rails
- Web 2.0 Starter Tookit for DB2

1.9.1 IBM Data Studio

IBM Data Studio é uma ferramenta baseada no Eclipse que lhe permite conceber, desenvolver, implementar e gerir os seus dados, bases de dados, aplicações de bases de dados ao longo de todo o ciclo de vida da gestão de dados. O Data Studio substitui a solução DB2 Developer Workbench 9.1.
O IBM Dados Studio ajuda-o a desenvolver funções, procedimentos armazenados, XQuery, Xpath, SQL e inclui um debugger integrado. Além disso, o Data Studio permite-lhe trabalhar com diagramas de modelação física de dados para compreender as relações entre tabelas. Também o pode ajudar a desenvolver e publicar dados como um serviço Web sem programação. Iremos discutir o Data Studio no Capítulo 14 – SQL PL Stored Procedures.

1.9.2 Servidor de aplicações DB2 embebido

O Servidor de aplicações DB2 embebido permite executar aplicações web fornecidas com o DB2 Versão 9.5, sem exigir um servidor de aplicações separado. As aplicações Web oferecidas com o DB2 Versão 9.5 são as seguintes:

- DB2 web tools, para administração web de bases de dados
- DB2WebServices, uma aplicação que automatiza a implantação de serviços web .NET da Microsoft Visual Studio para o Servidor de aplicações DB2 embebido.

1.9.3 DB2 9.5 Net Search Extender

Com o DB2 9,5 Net Search Extender, pode executar rápida e detalhadamente pesquisas em documentos de texto, incluindo quaisquer documentos XML armazenados nativamente no DB2 9.5.

1.9.4 *Starter Toolkit* para DB2 on Rails

O *Starter Toolkit* para DB2 on Rails é um conjunto conveniente de produtos e de tecnologias que permite a rápida criação de um ambiente DB2 para construir aplicações Web usando tecnologia Ruby on Rails. Todo o software necessário está incluído: DB2 Express-C; driver DB2 para Ruby; DB2 adaptador para Rails; juntamente com tutoriais, exemplos, e outros materiais pedagógicos. Iremos discutir Ruby on Rails no Capítulo 17 – Desenvolvimento em Java, PHP e Ruby.

1.9.5 Web 2.0 *Starter Toolkit* para DB2

Web 2.0 *Starter Toolkit* para DB2 é uma maneira fácil para começar a usar o DB2, PHP, e Dojo. Este ajuda-o a implantar o software necessário, indica-lhe *links* para tutoriais, e inclui aplicações de demonstração. Duas das aplicações de demonstração são o Painel de Controlo Atom Feed, que gera Atom feeds a partir de tabelas DB2, e os Web Services Control Panel, que cria serviços web REST em torno das suas tabelas DB2. Ambos dependem de Dojo significativo para o Ajax e capacidades como widget.

2

Capítulo 2 – Funcionalidades e produtos relacionados

Este capítulo descreve as funcionalidades oferecidas pela aquisição de uma subscrição anual do DB2 Express-C. Descreve também outras funcionalidades ***não*** disponíveis com a edição DB2 Express-C, mas pertencentes a outras edições do DB2, em alguns casos, através de uma taxa adicional.

As funcionalidades disponíveis na versão grátis do DB2 Express-C são:

- Funcionalidade Base do DB2
- Control Center, Data Studio e outras ferramentas de gestão
- pureXML
- Utilização de recursos até 2 Gb e 2 processadores
- Disponivel em Linux, Windows, e Solaris (x86)

As funcionalidades não disponiveis na versão grátis mas incluídas na licença de 12 Meses do DB2 Express-C são:

- *Fix packs*
- *High Availability Disaster Recovery* (HADR)
- Replicação de dados
- Utilização de recursos até 4Gb e 4 cores de processamento (em 2 *sockets)*

A seguinte tabela lista as várias funcionalidades do DB2 e a versão em que estas estão incluidas. Funcionalidades que podem ser compradas separadamente aparecem identificadas com o nome para a respectiva versão de DB2

Funcionalidade	**DB2 Express-C com licensa**	**DB2 Express Edition**	**DB2 Workgroup Server Edition**	**DB2 Enterprise Server Edition**
Homogeneous SQL replication	Sim	Sim	Sim	Sim
Net Search Extender	Sim	Sim	Sim	Sim
Spatial Extender	Sim	Sim	Sim	Sim
pureXML™ technology	Sim	pureXML Feature	pureXML Feature	pureXML Feature
High Availability Disaster Recovery (HADR)	Sim	High Availability Feature	Sim	Sim
Tivoli System Automation	Sim	High Availability Feature	Sim	Sim
Advanced Copy Services	Sim	High Availability Feature	Sim	Sim
Online reorganization	Sim	High Availability Feature	Sim	Sim

Homogeneous Federation	Não	Homogeneous Federation Feature	Homogeneous Federation Feature	Homogeneous Federation Feature
MQT	Não	Não	Query Optimization Feature	Sim
MDC	Não	Não	Query Optimization Feature	Sim
Query parallelism	Não	Não	Query Optimization Feature	Sim
Connection Concentrator	Não	Não	Não	Sim
Table partitioning	Não	Não	Não	Sim
DB2 Governor	Não	Não	Não	Sim
Compression: row level	Não	Não	Não	Storage Optimization Feature
Compression: backup	Não	Não	Não	Storage Optimization Feature
Label-bases access control (LBAC)	Não	Não	Não	Advanced Access Control Feature
Geodetic Extender	Não	Não	Não	Geodetic Data Management Feature
Query Patroller	Não	Não	Não	Performance Optimization Feature
Performance Expert	Não	Não	Não	Performance Optimization Feature
Homogeneous Q replication	Não	Não	Não	Performance Optimization Feature
Database partitioning	Não	Não	Não	Não

Funcionalidades disponiveis noutras versões de DB2 são:

Funcionalidades pagas no DB2 Express Edition:

- pureXML
- Alta disponibilidade
- Federação Homogénea

Funcionalidades incluidas gratuitamente na edição DB2 Workgroup Edition:

- Alta Disponibilidade
- Disponivel em AIX, Solaris, HP-UX, à parte de Linux e Windows.

Funcionalidades pagas no DB2 Workgroup Edition:

- pureXML
- Funcionalidade de Optimização de Consultas (MQT, MDC e paralelismo de Consultas)
- Federação Homogénea

Funcionalidades gratuitas no DB2 Enterprise Edition:

- Particionamento de Tabelas
- MQT – materialização de consultas em forma de Tabelas
- MDC - Cluster Multi-dimensional
- Recuperação de Desastres de Alta Disponibilidade
- Concentrador de conexões

Funcionalidades pagas no DB2 Enterprise Edition:

- pureXML
- Optimização de armazenamento (inclui compressão)
- Controlo de acesso avançado (LBAC)
- Optimização de performance (Workload Management, Performance Expert, Query Patroller)
- Suporte para dados geodésicos
- Federação Homogénea

Productos pagos relacionados com DB2:

- DB2 Connect
- DB2 Wharehouse Editions
- Websphere Federation Server
- Websphere Replication Server

2.1 Funcionalidades incluídas na subscrição do DB2 Express-C

Esta secção esboça os DB2 *Fix packs*, o HADR e a replicação SQL.

2.1.1 *Fix packs*

Um *fix pack* DB2 é um conjunto de correcções de código aplicadas a um produto DB2, de forma a corrigir os demais problemas encontrados após o lançamento do produto. Com uma licença de subscrição, os *fix packs* podem ser obtidos e instalados gratuitamente. *Fix packs* são fornecidos tipicamente a cada três meses.

Para descarregar o último fix pack, consulte o *site* de suporte técnico do DB2, em http://www.ibm.com/software/data/db2/support/db2_9/

2.1.2 Recuperação de Desastres de Alta Disponibilidade

A Recuperação de Desastres de Alta Disponibilidade (*High Availability Disaster Recovery* - HADR) é uma característica de fiabilidade de base de dados que oferece uma elevada disponibilidade e recuperação de danos para perdas totais ou parciais de um servidor. Um ambiente HADR consiste geralmente em dois servidores, um primário e um secundário (que podem encontrar-se em localizações geográficas diferentes). O servidor principal é aquele onde a base de dados fonte é armazenada e acedida por aplicações cliente. Á me-

dida que as transacções são processadas no servidor principal, o histórico da base de dados é automaticamente encaminhado para o servidor secundário através da rede. O servidor secundário tem uma cópia clone da base de dados do servidor principal, usualmente criada copiando a base de dados primária e substituindo-a no sistema secundário. Quando o histórico da base de dados primária é recebido, ele é repetido e aplicado à base de dados secundária. Através da cópia contínua do histórico, a base de dados secundária mantém uma réplica sincronizada da base de dados primária, réplica esta que pode substituir a principal caso ela falhe.

Uma solução completa de DB2 que contém o recurso HADR oferece-lhe:

1. Capacidade relâmpago de ligação alternativa a outro servidor em caso de falhas (*failover*), com completa transparência para aplicações cliente.
2. Atomicidade de transacção integral para prevenir perda de dados
3. A capacidade para actualizar sistemas ou aplicações sem que haja uma interrupção visível do serviço
4. Ligação alternativa a outro servidor em caso de falhas por parte do servidor remoto, proporcionando uma completa recuperação dos danos locais que tenham atingido o servidor de dados.
5. Fácil gestão com o apoio das ferramentas gráficas do DB2
6. Tudo isto sem um impacto significante na eficiência total do sistema

Nota:

Para ver uma demonstração sobre o funcionamento do HADR, visite o *site*:
http://www-306.ibm.com/software/data/db2/express/demo.html

2.1.3 Replicação de Dados

Esta funcionalidade permite a replicação de dados entre um servidor fonte, cujas alterações são anotadas, e um servidor alvo onde estas alterações são aplicadas.
A Figura 2.1 ilustra a forma pela qual a replicação é efectuada.

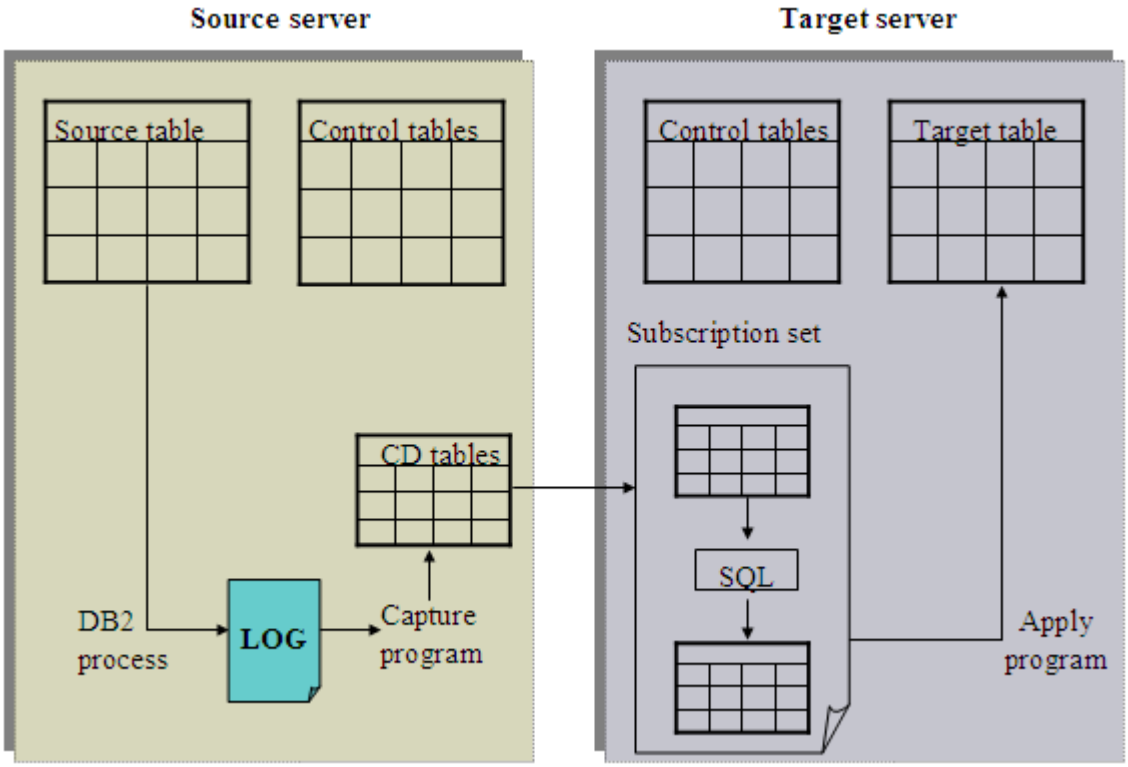


Figura 2.1 –Replicação SQL

Na Figura 2.1 encontram-se dois servidores, um servidor fonte e um servidor alvo. No servidor fonte, um programa de captura anota as alterações efectuadas à base de dados. No servidor alvo, um programa de actualização aplica as alterações à base de dados réplica. A replicação é útil para várias situações que requerem dados replicados, incluindo alívio de capacidade, abastecimento de armazéns de dados para suporte à decisão (data warehousing), e análise do histórico de alterações. Através da utilização da funcionalidade de replicação SQL é possível replicar dados entre servidores DB2 Express-C e outros servidores DB2, estando esses noutros sistemas Linux, UNIX, z/OS, e i5/OS.

2.2 Funcionalidades não disponíveis com o DB2 Express-C

Esta secção descreve algumas das funcionalidades disponíveis noutras edições do DB2, no entanto indisponíveis no DB2 Express-C.

2.2.1 Particionamento de Base de Dados

A funcionalidade de particionamento de base de dados (DPF – *Database Partitioning Feature*) encontra-se disponível apenas no DB2 Enterprise Edition através de um custo adicional. Permite que as bases de dados sejam difundidas através de múltiplas partições que podem residir em vários computadores. Esta funcionalidade de particionamento de base de dados é baseada numa arquitectura *shared-nothing*. Cada computador, à medida que vai sendo adicionado ao grupo de partição, acrescenta poder adicional de processamento com o seu próprio CPU e memória. O DPF é particularmente útil em ambientes de servido-

res de dados muito alargados como *data warehouses* onde consultas de sistemas de suporte à decisão são executados.

2.2.2 Concentrador de conexões

O concentrador de conexões (*Connection Concentrator*) é uma funcionalidade que permite dar suporte a um grande número de utilizadores conectados simultaneamente. Anteriormente, qualquer conexão de base de dados requeria um agente de base de dados. O concentrador de conexões introduz o conceito de um “agente lógico”, permitindo que um agente trate de várias conexões. Estes agentes serão discutidos com maior detalhe no Capítulo 6, Arquitectura DB2.

2.2.3 Extensão Geodésica

O *Geodetic Extender* é uma funcionalidade opcional (paga à parte) do DB2 Enterprise Edition. Esta extensão traz vantagens para decisões de negócios e aplicações electrónicas governamentais que requerem que a análise de localização geográfica seja feita de modo mais fácil. O *Geodetic Extender* pode construir um globo virtual a qualquer escala. A maioria da informação de localização é recolhida usando sistemas como o GPS, e pode ser representada através de coordenadas latitude/longitude (geocódigo). Dados para negócio, tais como moradas, podem ser convertidas para um geocódigo pelo *Geodetic Extender* do DB2, e as aplicações empresariais funcionam melhor se mantiverem os dados desta forma, deixando as projecções de mapas (globo para mapa plano) onde pertencem: na camada de apresentação, para visualizar e imprimir mapas.

2.2.4 *Workload Management* (WLM)

Gere a carga de trabalho – *workload* - *através* da base de dados baseada nas prioridades do utilizador e da aplicação combinadas com a disponibilidade de recursos e *workload thresholds* – limites impostos para cada carga. Permite regular o *workload* e consultas da base de dados para que as consultas mais importantes e mais prioritárias possam executar instantâneamente e previne que consultas mais exigentes monopolizem os recursos, garantindo que o sistema funciona da forma mais eficiente. O WLM é novo no DB2 9.5 e fornece recursos mais poderosos do que as ferramentas *Query Patroller* e o *DB2 Governor* disponiveis com versões mais antigas do DB2.

2.3 Produtos não-gratuitos relacionados com DB2

2.3.1 *DB2 Connect*

O *DB2 Connect* é um programa pago que permite que um cliente DB2 para Linux, UNIX ou Windows se ligue a um servidor DB2 para z/OS ou i5/OS como é mostrado na Figura 2.2. O *DB2 Connect* não é necessário quando a ligação ocorre no sentido contrário, isto é, quando se liga um cliente DB2 para z/OS ou i5/OS a um servidor DB2 para Linux, UNIX ou Windows. O DB2 está disponível nas duas edições principais, dependendo das suas necessidades de ligação: *DB2 Connect Personal Edition* e *DB2 Connect Enterprise Edition*.

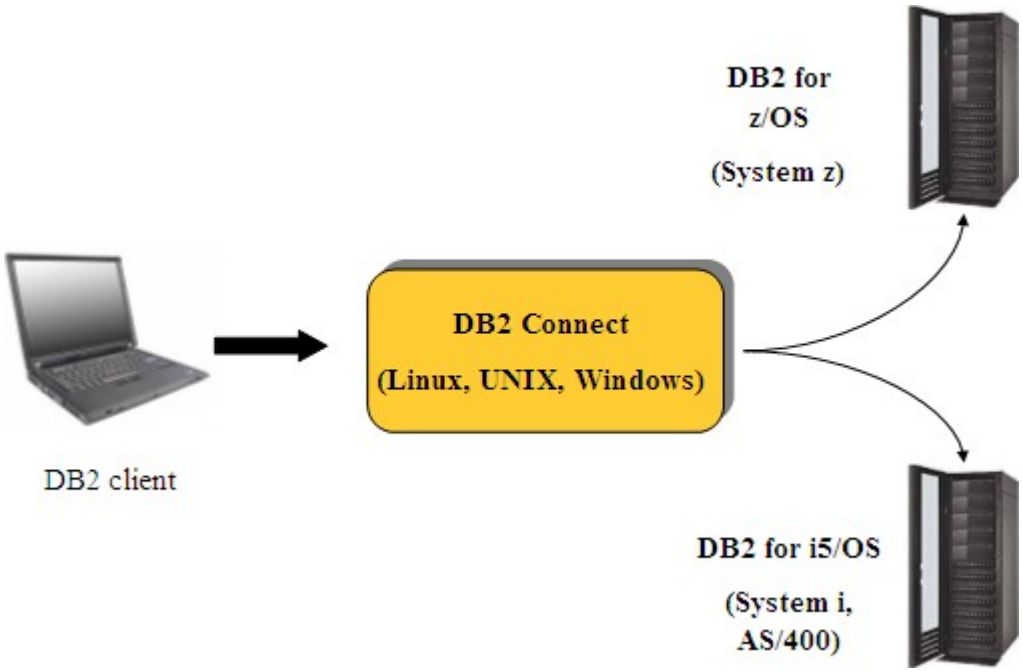


Figura 2.2 – DB2 Connect

2.3.2 *WebSphere Federation Server*

Anteriormente conhecido como *WebSphere Information Integrator*, o *WebSphere Federation Server* permite a federação de bases de dados, o que significa que é possível correr consultas de bases de dados que funcionem com objectos de diferentes sistemas de bases de dados relacionais. Por exemplo, se adquirir o *WebSphere Federation Server*, poderá executar a seguinte consulta:

```
SELECT  *
FROM      Oracle.Table1  A
             DB2.Table2  B
       SQLServer.Table3  C
WHERE
            A.col1 < 100
        and B.col5 = 1000
        and C.col2 = 'Test'
```

A Figura 2.3 ilustra a utilização do *WebSphere Federation Server*.

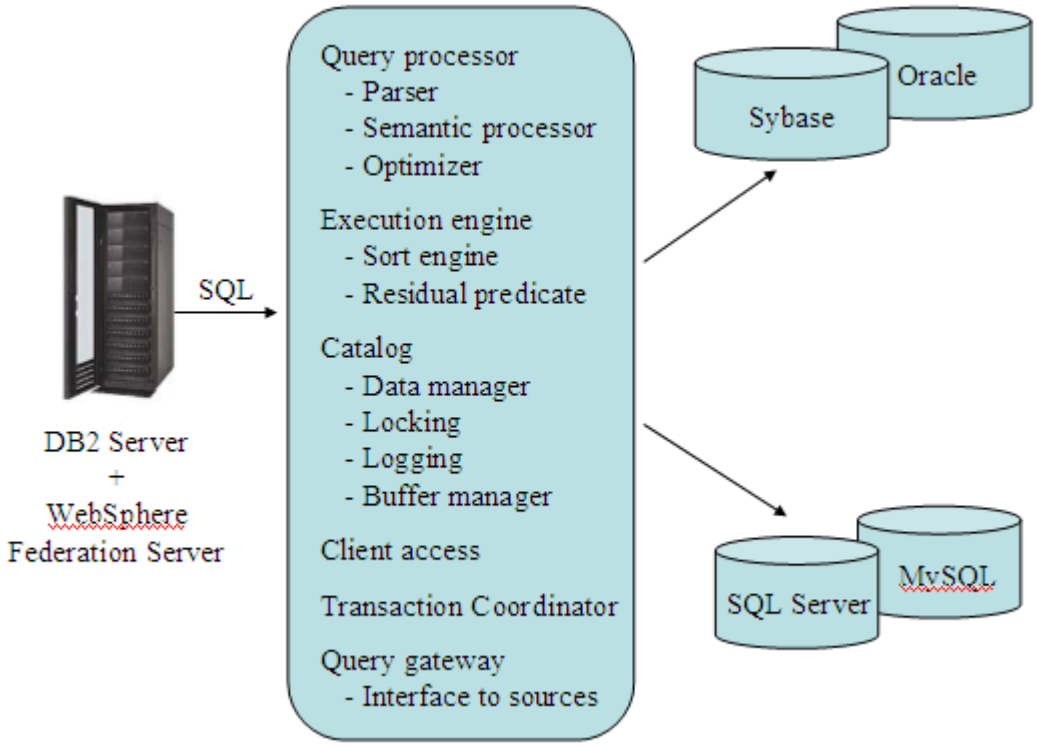


Figura 2.3 – *WebSphere Federation Server*

Para os sistemas de gestão de bases de dados relacionais que fazem parte da família IBM, o suporte de federação é incluído no DB2 Express-C. Isto significa que o produto *WebSphere Federation Server* não é necessário quando, por exemplo, deseja correr uma consulta entre duas bases de dados distintas, ou entre uma base de dados DB2 e uma base de dados Informix® (Informix faz parte da família IBM)

2.3.3 *WebSphere Replication Server*

Anteriormente conhecido como *WebSphere Information Integrator* (para suporte de replicação), o *WebSphere Replication Server* permite a replicação SQL de históricos de base de dados quando estão envolvidos servidores de dados não IBM. Inclui ainda uma funcionalidade conhecida como *Q-Replication* para replicar dados utilizando filas de mensagens.

3

Capítulo 3 – Instalação do DB2

Para instalar o DB2 Express-C Edition, em Linux ou Windows, assegure-se que o seu sistema satisfaz os pré-requisitos de instalação.

3.1 Pré-requisitos de instalação

Com respeito às versões dos sistemas operativos e nível de requisitos, o DB2 Express-C está disponível para Linux, Solaris (x64), e Windows 2003, 2000, XP, e Vista. As arquitecturas suportadas são 32-bit, 64-bit e PowerPC (Linux). Se precisa de correr DB2 noutra plataforma (como UNIX), deve comprar uma outra versão e servidor de dados descritos anteriormente neste livro. Os requisitos dos sistemas operativos e das edições DB2 estão descritas neste documento: http://www.ibm.com/software/data/db2/udb/sysreqs.html

Em termos de recursos de *hardware*, o DB2 Express-C pode ser instalado em sistemas com qualquer número de *cores* de CPU e memória, no entanto, apenas utilizará até 2 cores e 2 GB de memória nas versões gratuitas, e até 4 cores e 4 GB de memória para as versões com licença de 12 meses. Os sistemas podem ser físicos ou virtuais, criados por partições ou software de virtualização (máquinas virtuais). Pode, ser preferir, correr em sistemas mais pequenos, por exemplo, um processador e 1 GB de memória.

Para mais informações sobre requisitos hardware para DB2 Express-C, visite: http://www-306.ibm.com/software/data/db2/express/getstarted.html

3.2 Permissões para instalação

Para instalar DB2 Express-C em Linux ou Windows, deve possuir uma conta de utilizador com permissões suficientes.

Para **Linux**, necessita de ser root (super-utilizador) para instalar DB2 Express-C. Pode instalar DB2 Express-C com outro tipo de utilizador que não root, no entanto, está limitado nas funcionalidades das quais pode tirar partido. Por exemplo, sem root, não pode criar instâncias além da instância padrão da instalação.

Para **Windows**, o utilizador deve pertencer ao grupo de Administradores da máquina onde instala. Alternativamente, um não-Administrador pode ser utilizado desde que, antes, um Administrador configure os privilégios dos outros utilizadores para poder correr instalações.

Para contas de domínio Windows, para verificar os User IDs no DB2 Server, o User ID da instalação deve pertencer ao grupo de Administradores de Domínio, no domínio onde as contas serão criadas. Deve também usar o Sistema Local de contas para correr a instalação.

A conta de utilizador deve, além disso, deve possuir permissões para “Aceder a este computador através da rede”.

Nota:

Veja o vídeo de apresentação sobre instalação de DB2 Express-C neste endereço:
http://www.channeldb2.com/video/video/show?id=807741:Video:4442

3.3 Assistente de Instalação

Apesar de haver várias formas de instalar o DB2 Express-C, o método mais simples é usar o Assistente gráfico de instalação do DB2. Depois de descarregar e descompactar a imagem DB2 Express-C, pode executar o assistente de instalação:

- Windows: execute o setup.exe na directoria EXP/image.
- Linux: execute o comando db2setup na directoria exp/disk1.

A instalação é simples se seguir as instruções do assistente. Na maioria dos casos, as configurações padrão são suficientes, por isso tudo o que tem que fazer é aceitar a licença, clicar em "Next" várias vezes, e "Finish" para terminar. Após alguns minutos, a instalação estará completa e o DB2 pronto a ser utilizado.

A Figura 3.1 mostra o Assistente de instalação do DB2. Clique em "Install a Product" e depois escolha "Install New" para instalar uma nova cópia do DB2 Express-C no seu sistema.

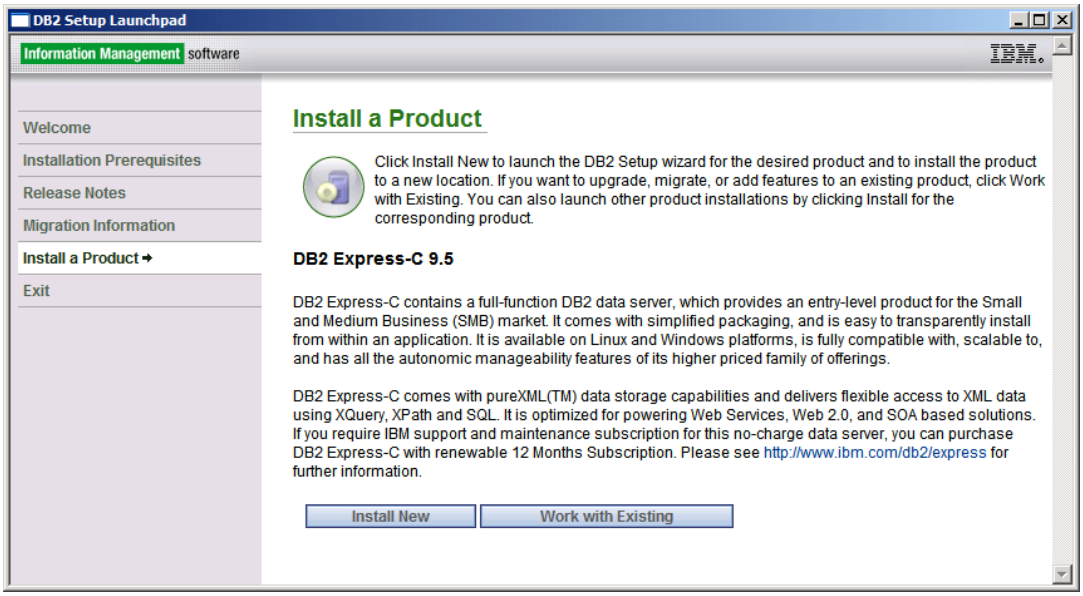


Figura 3.1 – Assistente de Instalação DB2.

Depois de aceitar a licença, uma instalação "Typical" (padrão) é suficiente, como mostra a figura 3.2.

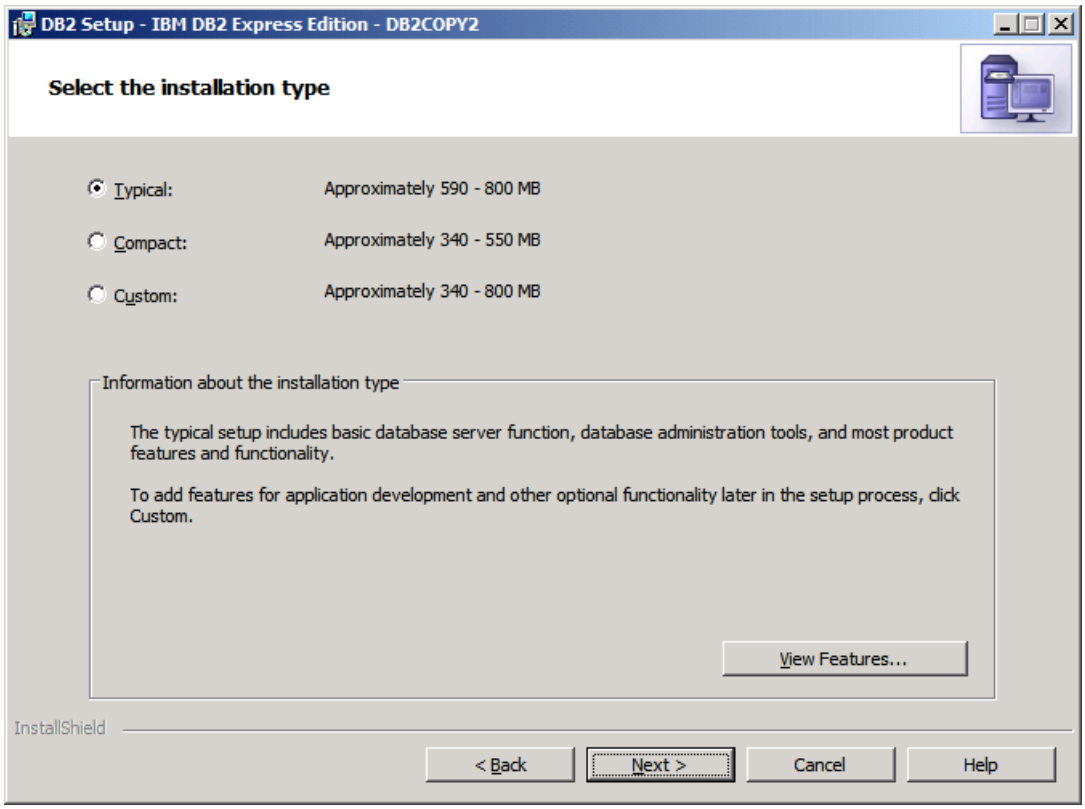


Figura 3.2 – Tipos de instalação

Na Figura 3.3, pode escolher entre instalar o produto, criar um ficheiro de resposta, ou ambos. Os ficheiros de resposta serão discutidos na secção 3.4 Instalação Silenciosa. Escolhendo a opção (Install IBM DB2 Express Edition on this computer and save my settings in a response file) é suficiente.

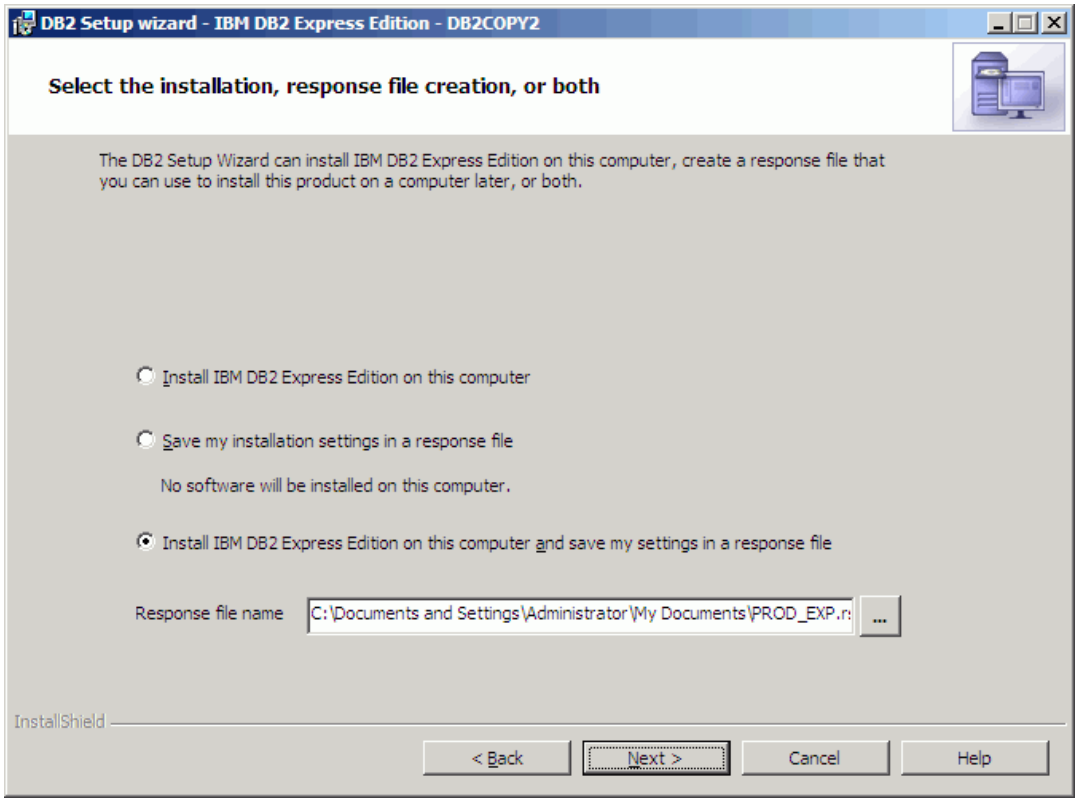


Figura 3.3 – Seleccionar **a Instalação**

Escolha os valores por omissão nos próximos ecrãs. Quando chegar ao ecrã mostrado na Figura 3.4., pode inserir um utilizador existente que será utilizado para trabalhar com a instância e outros serviços. Este utilizador deve fazer parte do grupo de Administradores Locais do Windows. Se o utilizador não pertencer a este grupo, será criado como Administrador Local. Pode deixar em branco, se o User ID não pertencer a um domínio. O User ID por defeito para Windows é db2admin. No caso de Linux, é db2inst1.

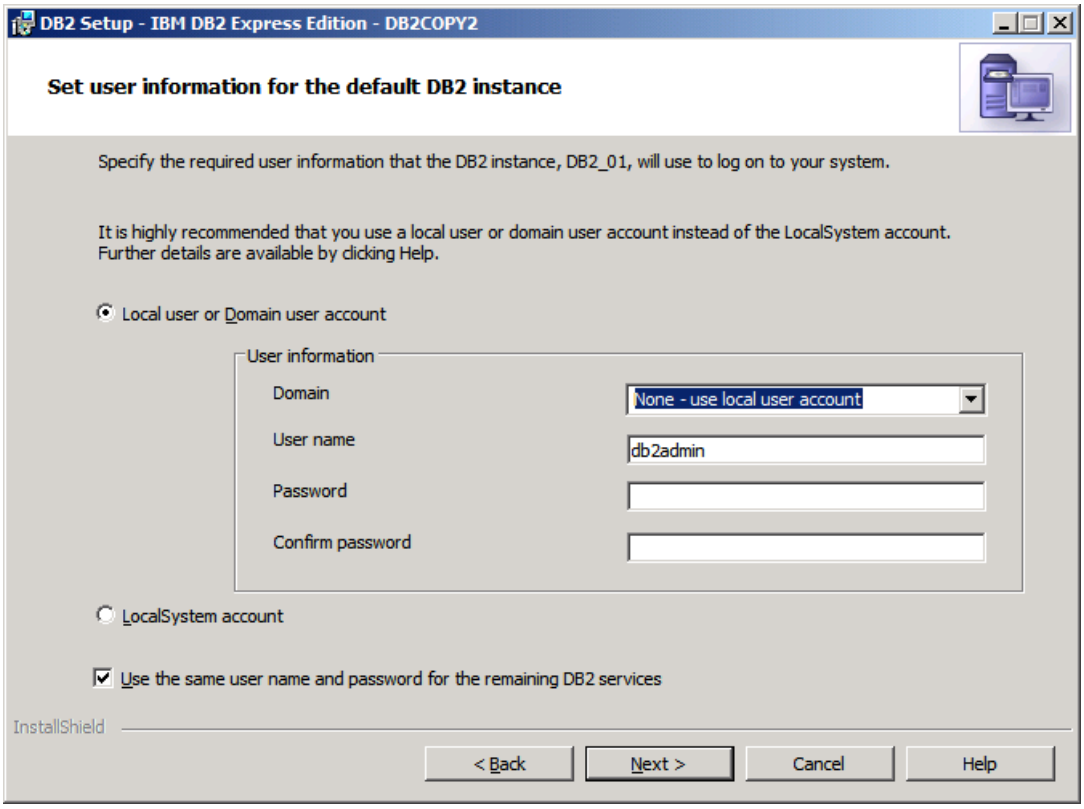


Figura 3.4 – Especificar informações do utilizador para a instância padrão do DB2

Finalmente, na Figura 3.5., o assistente de instalação mostra um resumo do que será instalado, e das configurações fornecidas nos passos anteriores. Quando clicar em “Finish”, a instalação irá iniciar-se, e os ficheiros serão copiados para o seu sistema.

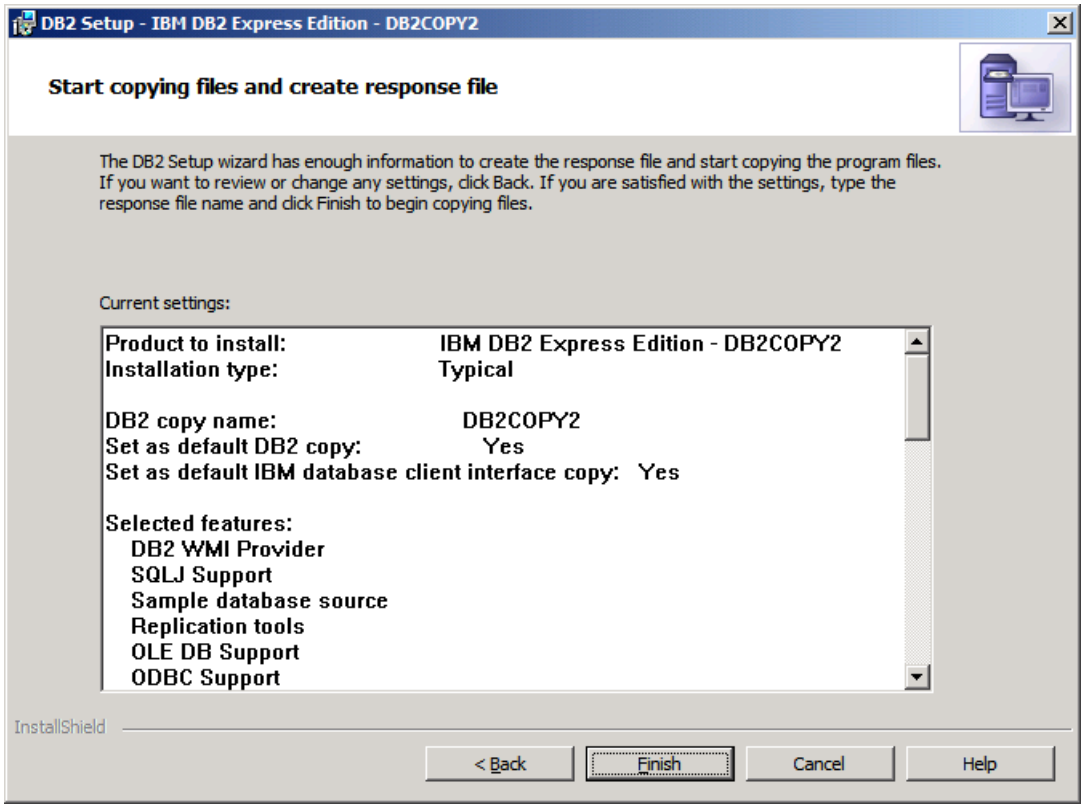


Figura 3.5 – Resumo do que será instalado

3.4 Instalação Silenciosa

Poderão existir situações em que precisará de instalar o cliente DB2 em vários computadores; ou em que precise de inserir o servidor de dados DB2 na sua aplicação, e gostaria de o instalar como parte do processo de instalação da sua aplicação. Nestas situações a instalação silenciosa é a solução para instalar o DB2.

O DB2 permite instalações silenciosas através de ficheiros de resposta que contêm informações sobre a instalação. Exemplo de um ficheiro de resposta:

```
PROD=UDB_EXPRESS_EDITION
LIC_AGREEMENT=ACCEPT
FILE=C:\Program Files\IBM\SQLLIB\
INSTALL_TYPE=TYPICAL

LANG=EN
```

```
INSTANCE=DB2
DB2.NAME=DB2
DEFAULT_INSTANCE=DB2
DB2.SVCENAME=db2c_DB2
DB2.DB2COMM=TCPIP
...
```

Existem várias maneiras de gerar ficheiros de resposta:

1. Instale o DB2 Express-C num computador utilizando o assistente de instalação. Uma das primeiras opções do assistente, permite-lhe seleccionar uma checkbox que lhe permite guardar os dados de instalação para um ficheiro de resposta. Este ficheiro será guardado na directoria que indicou, e tratando-se de um ficheiro de texto pode ser alterado. Ver Figura 3.3.

1. Edite o ficheiro de resposta que vem com a instalação do DB2. Este ficheiro (com extensão .rsp) está localizado na directoria *db2*/*platform*/samples/

    1. No Windows, pode também usar o comando de geração de ficheiros de resposta:

        db2rspgn –d <directoria>

Depois, para instalar o DB2 usando o ficheiro de resposta, no Windows use:

```
setup -u <response filename>
```

E em Linux:

```
db2setup -r <response filename>
```

QuickLab #1: Instalar DB2 Express-C e criar uma base de dados SAMPLE

Objectivo

Antes de começar a explorar as funcionalidades e ferramentas que vêm com o DB2 Express-C, deve primeiro instalá-lo no seu sistema. Neste QuickLab iremos executar uma instalação básica do DB2 no Windows. O mesmo assistente de instalação está disponível para Linux pelo que os passos a seguir são semelhantes nessa plataforma.

Procedimentos

- **Obter imagens de disco do DB2 Express-C**. Descarregue a imagem apropriada ou encomende o Discovery Kit DVD no site (ibm.com/db2/express). Descompacte todos os ficheiros numa directoria à sua escolha.

- **Localize os ficheiros**. Navegue pela directoria que contém os ficheiros descompactados para a instalação.

- **Corra o ficheiro de instalação**. Corra a instalação do DB2 fazendo duplo-clique sobre o ficheiro `setup.exe`. Em Linux, corra o comando `db2setup`, como root. A partir do Assistente, clique *Install Product*.

- **Corra o assistente de instalação**. O assistente de instalação garante que todos os pré-requisitos do sistema são cumpridos, e reconhece a existência de alguma instalação DB2 no sistema. Clique no botão *Next* para prosseguir com a instalação.

- **Leia a licença**. Leia e aceite (seleccione "*I Accept...*") e clique *Next* para continuar

- **Escolha o tipo de instalação.** Para este exercício, seleccione *Typical (por defeito)*. A opção *Compact* executa uma instalação básica, enquanto que a *Custom* permite configurar as funcionalidades que deseja instalar. Clique *Next* para continuar.

- **Seleccione a directoria da instalação.** Este ecrã permite-lhe configurar a unidade e directoria onde o DB2 será instalado. Assegure-se que existe espaço suficiente para a instalação. Utilize a directoria por defeito:

  Drive: C:
  Directory: C:\Program Files\IBM\SQLLIB

  Clique *Next* para continuar.

- **Escolher a informação do utilizador.** Uma vez o DB2 instalado, alguns processos DB2 correm como serviços do sistema. Este serviços precisam de uma conta

do sistema operativo para correr. Em Windows é recomendado utilizar a conta **db2admin**. Se a conta do utilizador não existir, o DB2 cria-a no sistema operativo. Pode também especificar se quiser usar uma conta existente, mas que tenha permissões de administrador. Recomendamos que use as configurações sugeridas por defeito. Assegure-se que especifica uma password para a conta. Em Linux use o User ID **db2inst1** para *instance owner*, **db2fenc1** para *fenced user* e **dasusr1** para administrador do servidor DB2. Clique *Next* para continuar.

- **Configure a instância DB2**. A instância DB2 pode ser entendida como um contentor de base de dados. Uma instância deve existir para que uma base de dados possa ser criada nesta. Durante a instalação em Windows, uma instância chamada DB2 será automaticamente criada. Em Linux, será a instância *db2inst1*. Falaremos de instâncias mais adiante neste livro.

  Por omissão, a instância DB2 está configurada para "escutar" conexões TCP/IP na porta 50000. Tanto o protocolo como a porta podem ser alterados clicando nos botões *Protocols* e *Startup,* respectivamente. Recomendamos que se usem os protocolos por defeito neste exemplo. Clique *Next* para continuar.

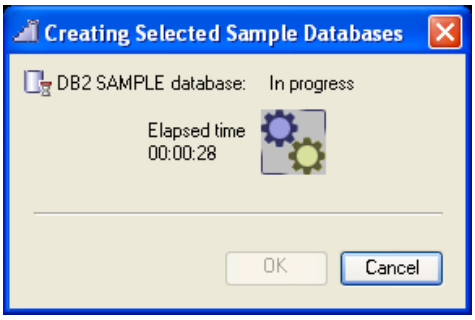
- **Iniciar instalação.** Reveja as opções de instalação previamente seleccionadas. Clique *Install* para começar a cópia de ficheiros para o local de instalação. O DB2 irá também executar alguns processo iniciais de configuração.

- **Primeiros passos**. Depois da instalação estar completa, outro utilitário, chamado *First Steps*, arrancará. Este utilitário pode também ser executado à posteriori com o comando `db2fs`.

- A base de dados SAMPLE é uma base de dados que pode usar para efeitos de teste. É criada automaticamente, logo após a instalação do DB2. Verifique que a base de dados existe abrindo o Centro de Controlo DB2 (*Control Center)*. Para abrir esta ferramenta, a partir do menu Iniciar do Windos, vá a *Menu Iniciar -> Programas -> IBM DB2 -> DB2COPY1 (Default) -> General Administration Tools ->Control Center.* Pode também executar o *Contron Center* através do comando `db2cc`.

- Se a base de dados é exibida no *Control Center*, pode passar para o passo 16. Se não aparecer clique em *Refresh* no menu *View* para assegurar que está a ver a informação mais actualizada. Se ainda assim a base de dados SAMPLE não aparecer, pode não ter sido criada. Pode criá-la manualmente a partir no utilitário First Steps. Escolha a tab que diz "Database Creation", e depois siga o assistente para criar a base de dados SAMPLE. Assegure-se que selecciona a opção *XML and SQL objects and data*, e clique OK. Esta última opção irá criar uma base de dados UNICODE, que era necessária na versão 9 para suportar pureXML, mas já não é necessária na versão 9.5.

- O seguinte ecrã é exibido enquanto a base de dados é criada. (Este procedimento pode demorar algum tempo). Quando a criação da base de dados estiver completa, clique OK e feche o utilitário First Steps.



- Retroceda ao Centro de Controlo e verifique que uma base de dados chamada SAMPLE aparece agora no painel com a árvore de objectos. Pode actualizar para ver novas alterações. Para isso, clique em *Refresh* no menu *View* do Centro de Controlo.

- **Reinicie o computador.** Apesar de este passo não vir mencionado na documentação oficial de instalação DB2, recomendamos que reinicie o seu sistema (se possível, pelo menos no Windows) para assegurar que todos os processos se iniciam com sucesso e para limpar recuros de memória que possam não ter sido correctamente limpos. Isto é OPCIONAL.

4

Capítulo 4 – Ambiente do DB2

Neste capítulo iremos falar sobre o ambiente do DB2. Na Figura 4.1 podemos ver um resumo sobre o DB2, e a elipse vermelha mostra a área que iremos focar neste capítulo. O lado esquerdo da figura mostra os diferentes comandos DB2, os comandos SQL, SQL/XML, e XQuery que podem ser criados para interagir com um servidor de dados DB2. A meio da figura pode-se ver os nomes das diferentes ferramentas utilizadas para interagir com um servidor de dados DB2. O lado direito da figura mostra o ambiente básico do DB2 que consiste numa instância, uma base de dados, e os respectivos arquivos de configuração.

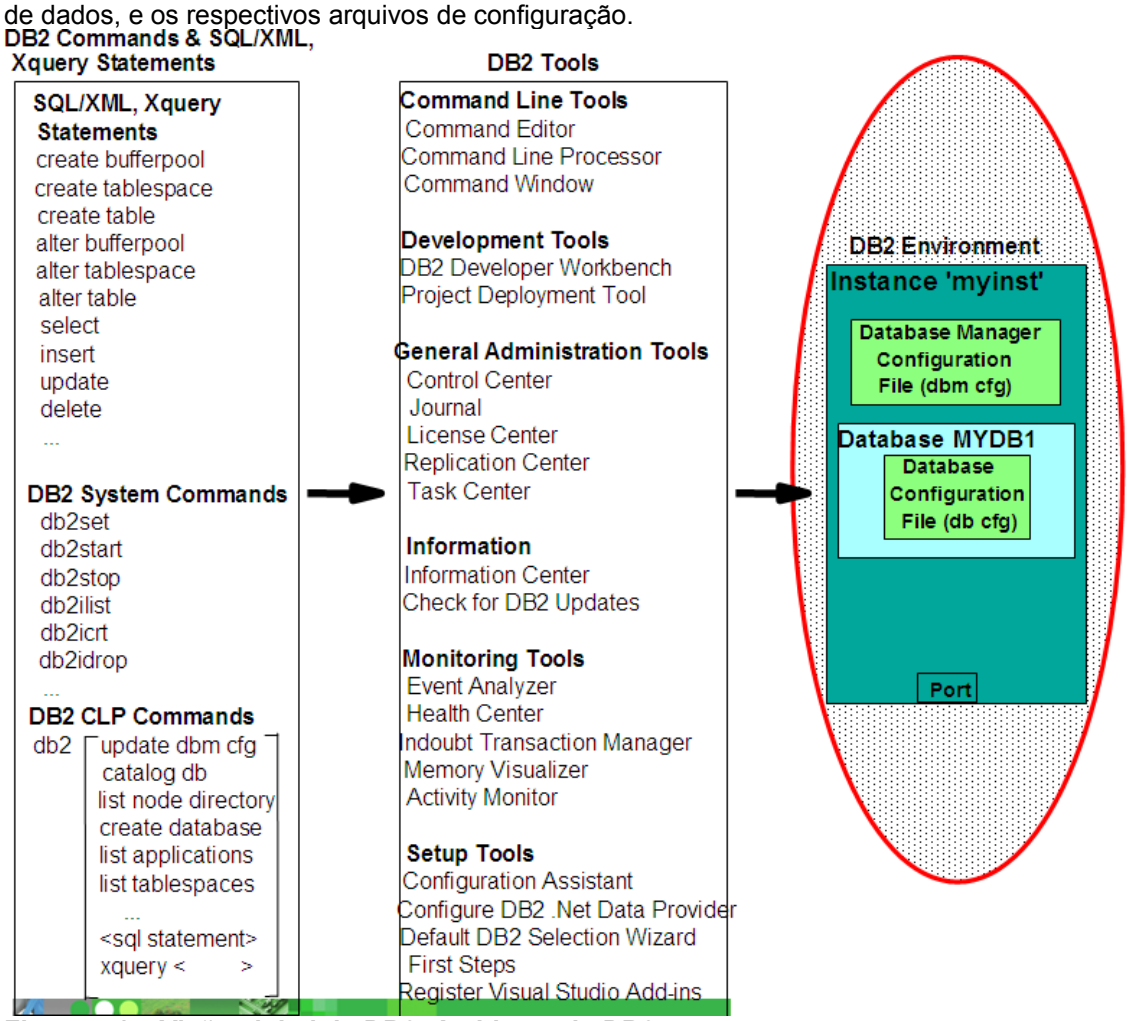


Figura 4.1 – Visão global do DB2: Ambiente do DB2

Nota:

Vídeos de apresentações sobre o ambiente de DB2 nos seguintes links:
http://www.channeldb2.com/video/video/show?id=807741:Video:4029
http://www.channeldb2.com/video/video/show?id=807741:Video:4042

Para descrever o ambiente do DB2, vamos descrever, passo a passo, cada elemento que o compõe. A Figura 4.2 mostra uma representação de um servidor de dados DB2 depois de instalar o DB2 Express-C 9.5.

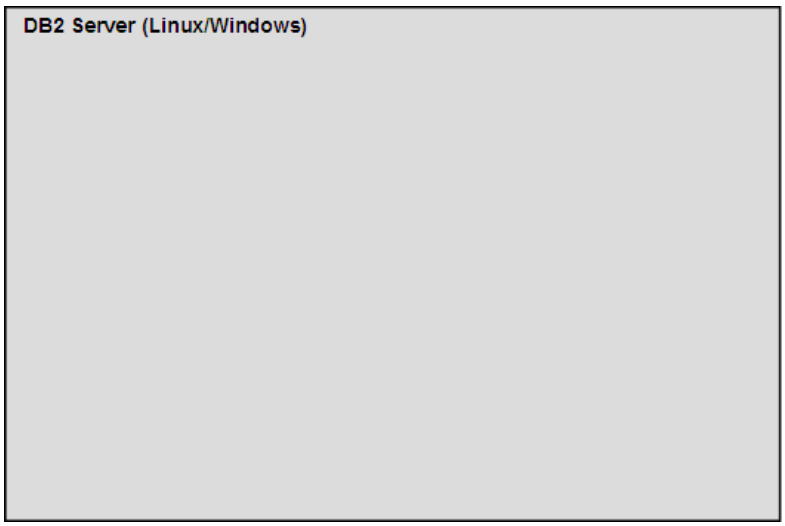


Figura 4.2 – Representação de um servidor de dados DB2 depois de instalar o DB2 Express-C 9.5

Como parte da instalação no Windows, é criada uma instância padrão chamada "DB2" ("db2inst1" no Linux). Esta é representada pela caixa verde na Figura 4.3. Uma instância é simplesmente um ambiente independente onde as aplicações podem ser executadas e onde as bases de dados podem ser criadas. Podemos criar várias instâncias num servidor de dados, e usá-las para diferentes fins. Por exemplo, uma instância pode ser usada para armazenar dados para uso de produção, outra pode ser usada para ambiente de testes, e outra para ambiente de desenvolvimento. Todas estas instâncias são independentes, isto é, realizar operações numa instância não irá afectar as outras instâncias.

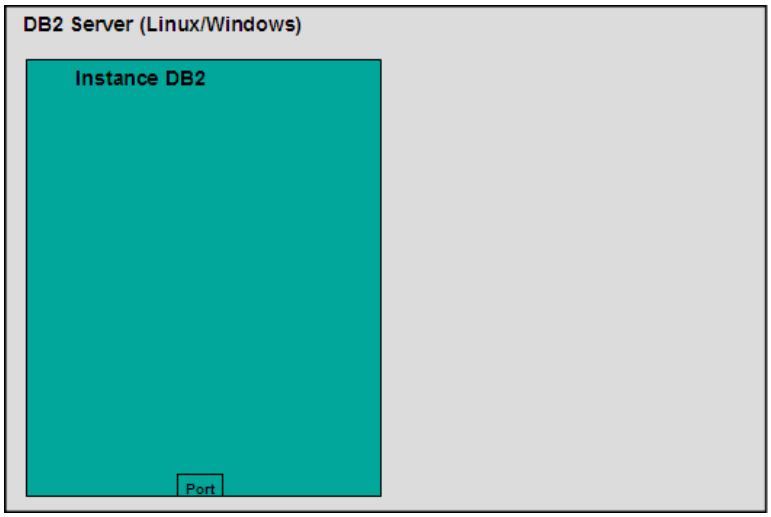


Figura 4.3 – Instância padrão do DB2 criada

Para criar uma nova instância DB2, utiliza-se o comando `db2icrt <nome da instância>`, onde `<nome da instância>` é substituído por um qualquer nome de 8 caracteres. Por exemplo, para criar a instância *myinst*, usa-se o seguinte comando: `db2icrt myinst`.

A figura 4.4 mostra a nova instância chamada `myinst` como uma caixa verde separada.

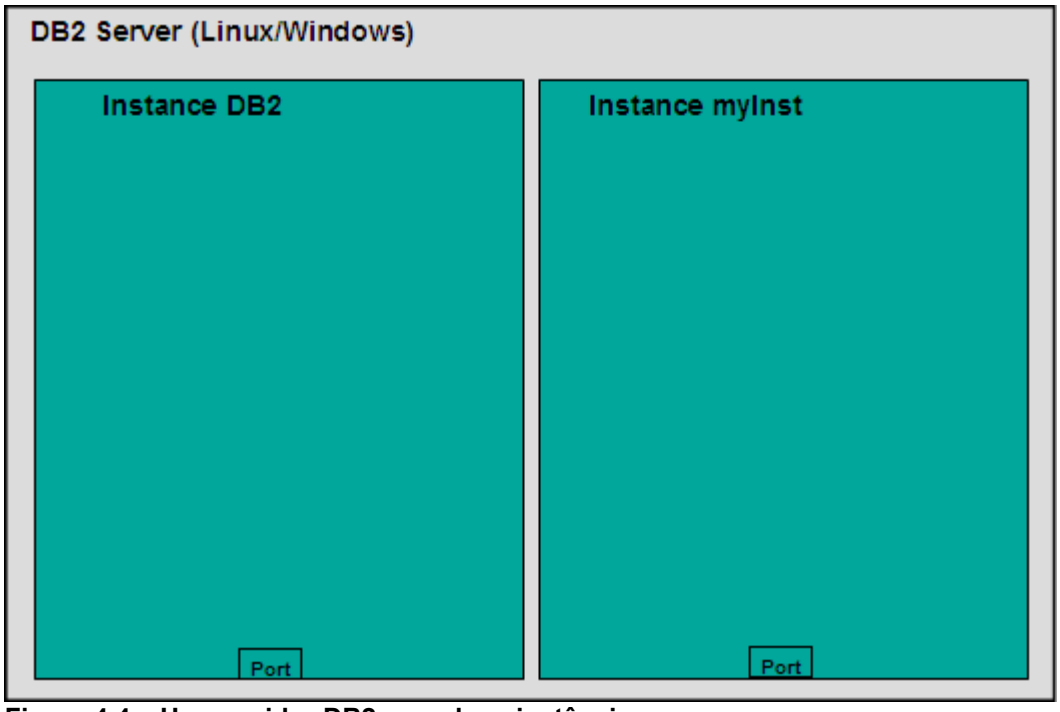


Figura 4.4 – Um servidor DB2 com duas instâncias

De notar que cada instância tem um porta única. Isto ajuda a distinguir as duas instâncias quando nos queremos ligar, através de um cliente remoto, a uma base de dados que está numa das instâncias. Se usarmos o *DB2 Command Window (*consola de comandos DB2), podemos activar qualquer instância DB2 usando este comando do sistema operativo Windows:

```
set db2instance=myinst
```

Neste exemplo, se queremos criar uma base de dados a partir do *DB2 Command Window*, esta será criada na instância `myinst`.

Para listar as instâncias, executa-se este comando:

```
db2ilist
```

Em Linux, uma instância deve coincidir com um utilizador do sistema operativo; portanto, para alternar entre as instâncias pode-se simplesmente mudar de utilizador (com o comando `su`).

A Tabela 4.1 mostra alguns comandos úteis para instâncias.

Command	Description
db2start	Inicia a instância actual
db2stop	Pára a instância actual
db2icrt	Cria uma nova instância
db2idrop	Elimina uma instância
db2ilist	Lista as instâncias que o sistema tem
db2 get instance	Lista as instâncias activas actuais

Tabela 4.1 – Comandos DB2 úteis para instâncias.

Alguns dos comandos acima descritos podem também ser feitos através do *Control Center*. Por exemplo, no *Control Center*, se expandirmos a pasta *Instâncias* (*Instances*) e clicarmos com o botão direito do rato na instância pretendida podemos “iniciar” (*Start*) que é equivalente a executar o comando `db2start` na consola DB2. O mesmo se aplica à acção “parar” (*Stop*), que é equivalente a executar o comando `db2stop` como mostra na Figura 4.5.

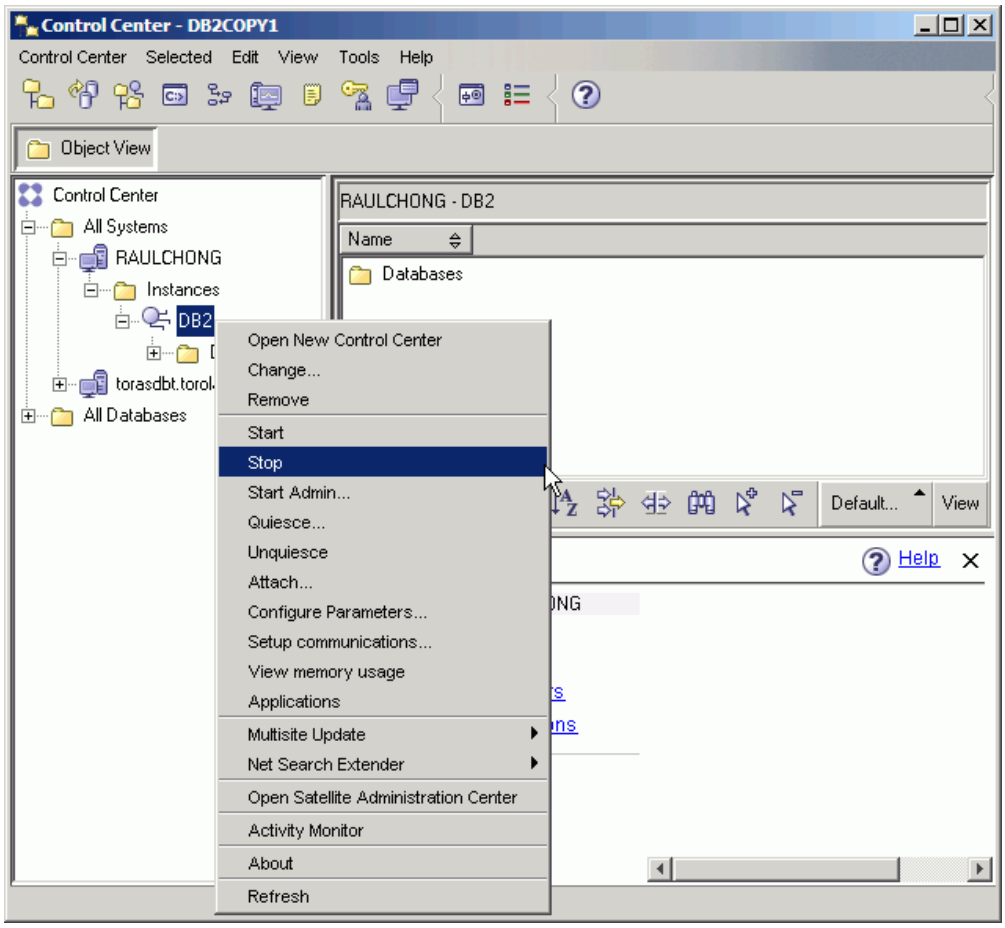


Figura 4.5 – Executar comandos para instâncias a partir do *Control Center*

Para criar uma base de dados na instância activa, executa-se este comando no *DB2 Command Window*:

```
db2 create database mydb1
```

Para listar todas as bases de dados criadas, executa-se este comando:

```
db2 list db directory
```

Muitas bases de dados podem ser criadas dentro de uma qualquer instância. Uma base de dados é uma colecção de objectos, tais como tabelas, vistas (*views*), índices (*indexes*) entre outros. As bases de dados são unidades independentes e, portanto, não partilham objectos com outras bases de dados. A Figura 4.6 mostra uma representação de uma base de dados "MYDB1" criada dentro da instância "DB2".

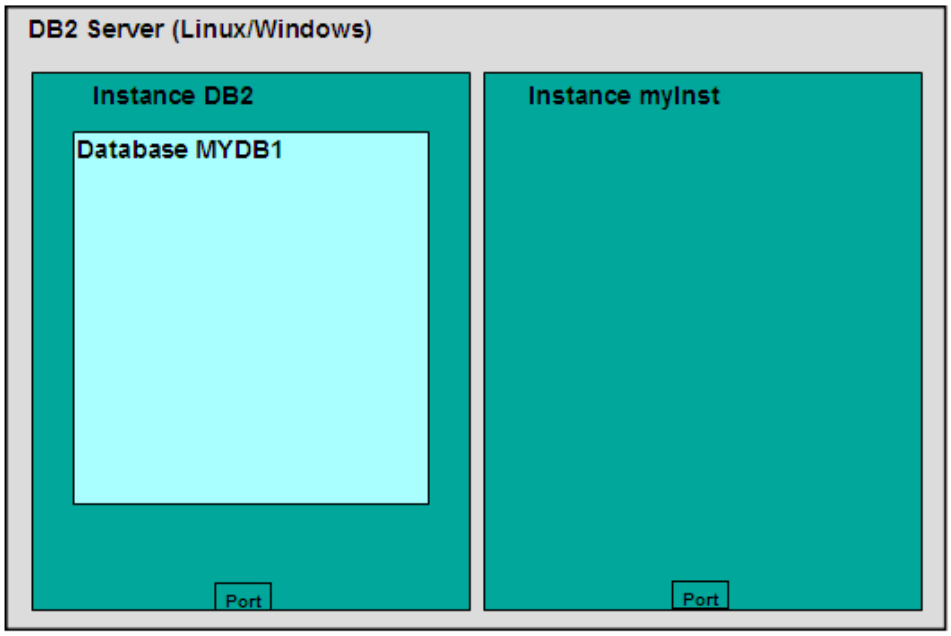


Figura 4.6 – Base de dados "MYDB1" criada dentro da instância "DB2"

A Tabela 4.2 mostra alguns comandos que podem ser usados nas bases de dados.

Command/SQL statement	Description
`db2 create database`	Cria uma nova base de dados
`db2 drop database`	Elimina uma base de dados
`db2 connect to <database_name>`	Liga-se a uma base de dados
`db2 create table/create view/create index`	Comando SQL para criar, respectivamente, tabelas, *views* e *indexes*

Tabela 4.2 – Comandos SQL para bases de dados

Se quisermos criar uma outra base de dados com o mesmo nome (MYDB1), mas na instância "myinst", os seguintes comandos devem ser executados no *DB2 Command Window*:

```
db2 list db directory
set db2instance=myinst
db2 create database mydb1
set db2instance=db2
```

A Figura 4.7 mostra a nova base de dados "MYDB1" criada na instância "myinst".

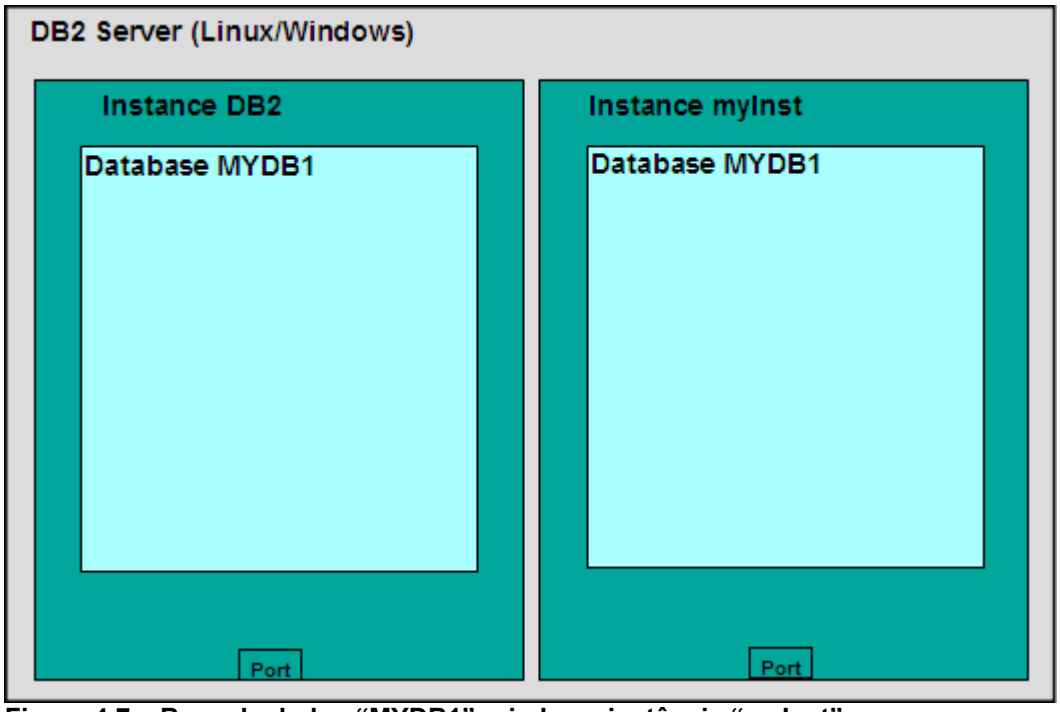


Figura 4.7 – Base de dados "MYDB1" criada na instância "myInst"

Quando é criada uma base de dados vários objectos são criados por defeito: *Table spaces*, tabelas, um *buffer pool* e ficheiros de *log*. O tempo de execução do comando `create database` requer alguns minutos para criar todos os objectos padrão de uma base de dados. A Figura 4.8 mostra três *table spaces* criadas por defeito. Falaremos das *Table spaces* mais detalhadamente no Capítulo 6 – Arquitectura do DB2; por agora, pensemos em *Table spaces* como uma camada lógica entre as tabelas lógicas, e os recursos físicos, tais como discos e memória.

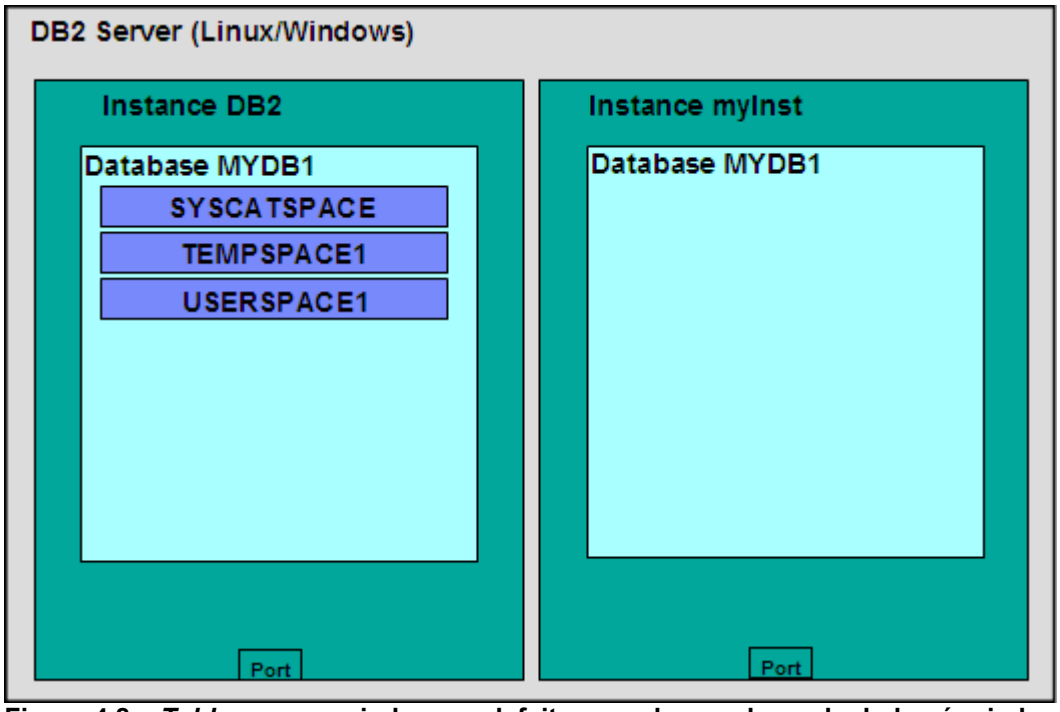


Figura 4.8 – *Table spaces* criadas por defeito quando uma base de dados é criada

A *Table space* SYSCATSPACE contém as tabelas Catálogo (*Catalog tables)*. O *Catalog* é também conhecido noutros sistemas de gestão de dados relacionais como o dicionário de dados. Basicamente, o sistema contém informações que não devem ser alteradas ou suprimidas, caso contrário o banco de dados não irá funcionar correctamente. O *Table space* TEMPSPACE1 é usado pelo DB2 quando este precisa de espaço adicional para executar algumas operações, por exemplo ordenações. O *Table space* USERSPACE1 é normalmente usado para armazenar tabelas de dados de utilizadores, se não for especificado um *Table space* ao criar uma tabela.

Também podemos criar os nossos próprios *Table spaces* usando o comando CREATE TABLESPACE. Na Figura 4.9 o *Table space* MYTBLS1 é criado dentro da base de dados na instância MYDB1. Quando criamos um *Table space*, especificamos os discos para utilização e a memória (*buffer pool*). Portanto, se tivermos uma *"hot" table*, isto é, uma tabela que é usada com muita frequência, podemos atribuir os discos mais rápidos e a maior quantidade de memória através da atribuição de um *Table space* com essas características.

Na Figura 4.9, estão representados outros dois objectos criados por defeito: Um *buffer pool* chamado IBMDEFAULTBP, e os ficheiros de *log*.

Um *buffer pool* é basicamente um bloco de memória *cache* utilizada pela base de dados. Pode-se criar um ou mais *buffer pool*, mas deverá haver sempre um *buffer pool* com um tamanho de página que corresponda ao tamanho de página do *Table space* existente. Páginas e tamanho de página serão discutidas com mais detalhe no Capítulo 6 – Arquitectura do DB2.

Os ficheiros de *log* são usados para recuperação. Quando se trabalha numa base de dados, não é só a informação que está armazenada no disco; os ficheiros de *log* armazenam todas as operações executadas sobre os dados. Pense nos *logs* como ficheiros temporários que são gravados automaticamente. Os *logs* serão discutidos com maior detalhe no Capítulo 11 – Cópias de Segurança e Recuperação.

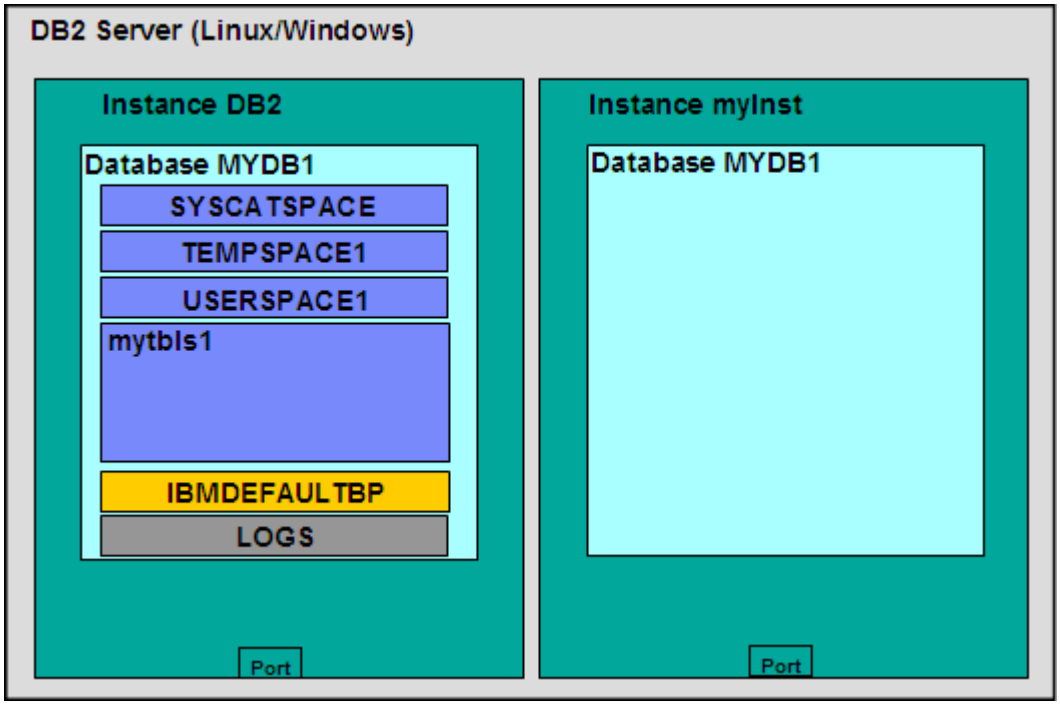


Figura 4.9 – *Buffer pool* e *logs* criados por omissão

Anteriormente, vimos que instâncias são independentes do ambiente e, portanto, uma base de dados com o mesmo nome poderia ser criada em várias instâncias. Assim como as instâncias, as bases de dados são unidades independentes; portanto, objectos numa base de dados não têm qualquer relação com objectos que estão noutra base de dados. A Figura 4.10 mostra o *Table space* "mytbls1" dentro da base de dados MYDB1 e da base de dados SAMPLE da mesma instância DB2. Isso é possível porque as bases de dados são unidades independentes. Note-se que a Figura 4.10 não mostra, para a base de dados SAMPLE, os outros objectos padrão devido à restrição de espaço para a figura.

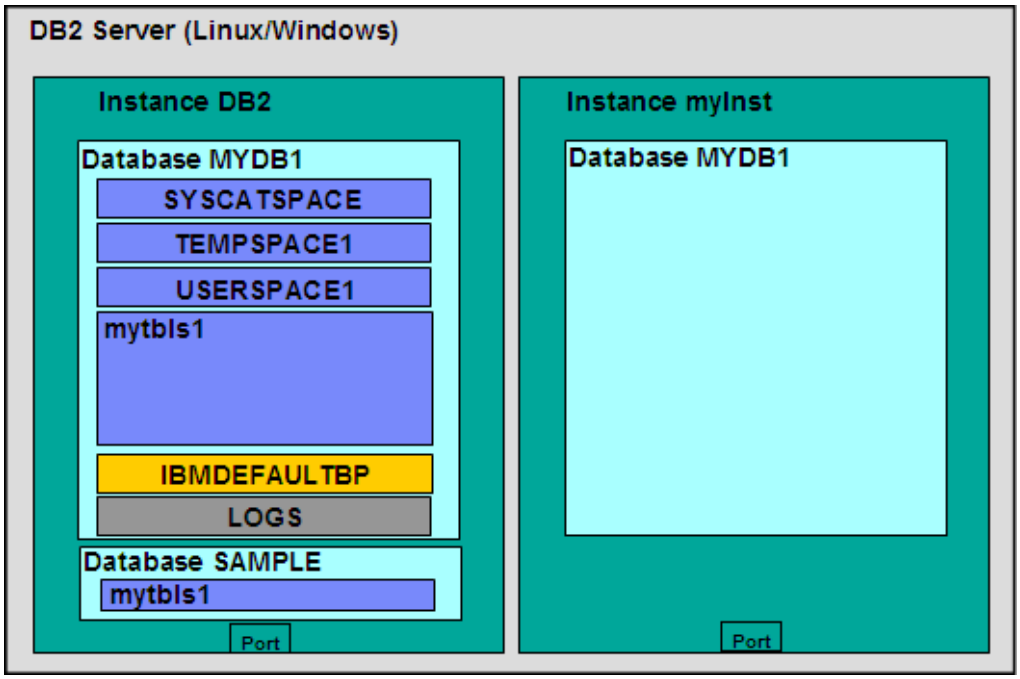


Figura 4.10 – *Table spaces* com o mesmo nome em bases de dados diferentes.

Depois de termos criado um *Table space*, podemos criar objectos dentro do *Table space*, como por exemplo tabelas, *views* e *indexes* (Figura 4.11).

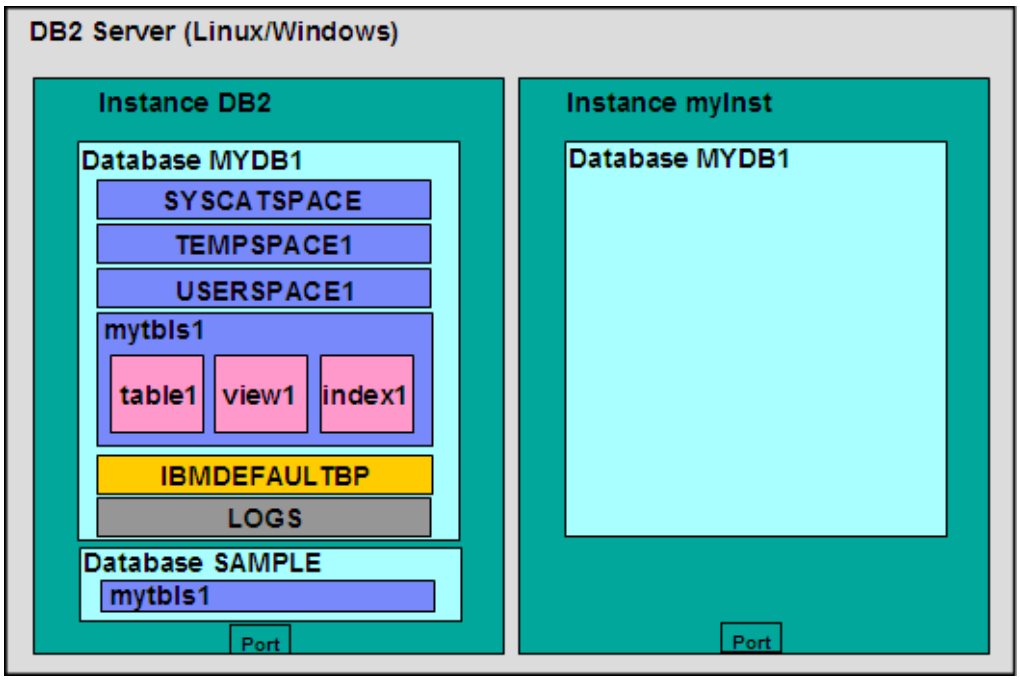


Figura 4.11 – Tabelas, *views*, *indexes* criados dentro de uma *Table space*

Quicklab #2 – Criar uma nova base de dados

Objectivo
Neste Quicklab, irá criar uma nova base de dados utilizando o *wizard* Create Database do Control Center.

Procedimento

1. Através do painel do Control Center, clique com o botão direito na pasta *All Databases*, seleccione o item *Create Database*, e escolha o item *With Automatic Maintenance*. Irá aparecer o Create Database Wizard.

2. Especifique o nome e a localização da base de dados na *Name page* do wizard.
   Utilize os seguintes valores:
   Database Name: EXPRESS
   Default Drive (Windows): C:
   Default Path: (Linux): /home/db2inst1

   Alias: Campo default para EXPRESS se deixado em branco
   Comment: Opcional e pode ser deixado em branco
   Clique no botão *Next* para continuar para a próxima página do *wizard.*

3. Em *Specify where to store your data page*, não faça alterações, e clique em *Next.*

4. Em *Select your maintenance strategy page*, deixe o valor por omissão (*Yes, I can specify an offline ...), e clique Next.*

5. Especifique a janela de tempo de manutenção offline na *Timing page* do *wizard*. Especifique duas ou mais horas por semana para o DB2 pode realizar a manutenção automatica para preservar a saúde da sua base de dados. Por agora, configure a janela para começar à 1AM todas as Segundas a Quintas por um período de 6 horas. Clique no botão *Next*.

6. Configure notificação na página *Mail Server* do *wizard.*O DB2 pode automaticamente mandar um email ou um *page* se algum problema for detectado. Se quiser configura isso, indique um servidor SMTP válido para o DB2 usar para enviar emails. Para este lab não temos um servidor SMTP, por isso deixe este campo em branco, e clique em *Next*.

7. Reveja as opções seleccionadas na página *Summary* do *wizard.* Clique no botão *Finish* para começar o processo de criação da base de dados. A cri-

ação da base de dados normalmente demora alguns minutos, e durante esse tempo um indicador de progresso é exibido.

4.1 Configuração do DB2

Os parâmetros do DB2 podem ser configurados usando o *Configuration Advisor Tool*. Para aceder ao *Configuration Advisor* através do *Control Center*, clique numa base de dados e escolha *"Configuration Advisor"*. Com base em algumas respostas a perguntas sobre os recursos do sistema e da carga de trabalho (*workload*) do mesmo, o *Configuration Advisor* irá fornecer uma lista de parâmetros DB2 que devem ser alterados com os valores sugeridos para cada um. Se quiser mais informações sobre a configuração do DB2, continue a ler; senão, use o *Configuration Advisor* e está pronto para trabalhar com DB2!

Um servidor DB2 pode ser configurado em quatro níveis diferentes:

- Variáveis de ambiente
- Ficheiro de configuração do gestor de base de dados (dbm cfg)
- Ficheiro de configuração da base de dados (db cfg)
- Perfil de registo DB2

Note na Figura 4.12. onde cada uma das caixas está. Por exemplo, as variáveis de ambiente são fixadas ao nível do sistema operativo do servidor, enquanto que os parâmetros do ficheiro de configuração do gestor de bases de dados são fixados ao nível da instância. Os parâmetros de configuração da base de dados são definidos no nível da base de dados, e o perfil de registo do DB2 é definido ao nível do sistema operativo ou de instância.

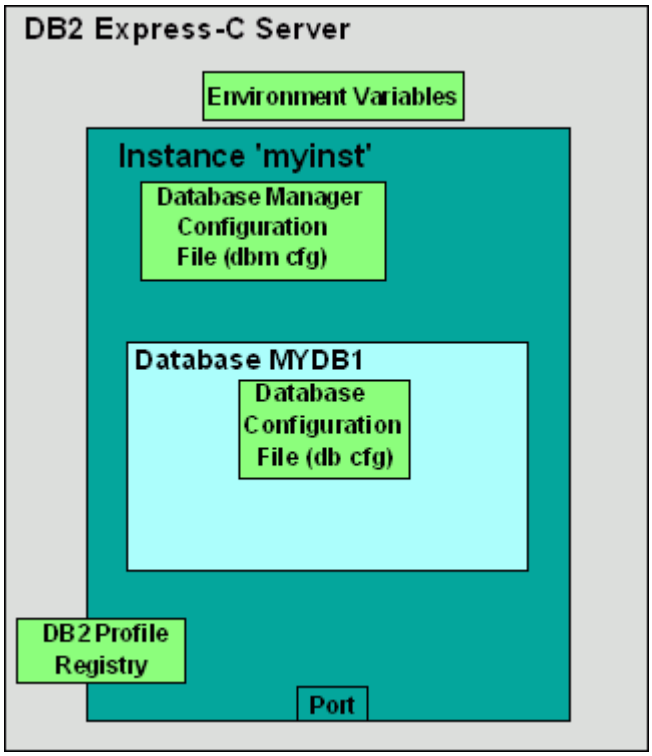


Figura 4.12 – Configuração do DB2

4.1.1 Variáveis de ambiente

Variáveis de ambiente são variáveis definidas ao nível do sistema operativo. Uma das principais variáveis de ambiente é a variável DB2INSTANCE. Esta variável indica a instância activa onde se está a trabalhar, e onde os seus comandos DB2 serão aplicados. Por exemplo, para definir, no Windows, a instância "myinst" como activa, pode-se executar este comando do sistema operativo: `set db2instance=myinst`

4.1.2 Ficheiro de configuração do gestor da base de dados (dbm cfg)

O ficheiro de configuração do gestor de base de dados *(Database manager configuration file)* (dbm cfg) inclui parâmetros que afectam a instância e todas as bases de dados contidas nela. O ficheiro de configuração do gestor de base de dados pode ser visualizado ou modificado utilizando a linha de comandos ou através do DB2 *Control Center*.

Para trabalhar com o DBM CFG a partir do *Control Center*, seleccione a instância a partir da pasta das instâncias do *Control Center*, clique com o botão direito do rato para ver o menu e seleccione Configurar Parâmetros *(Configure Parameters)*. Figura 4.13:

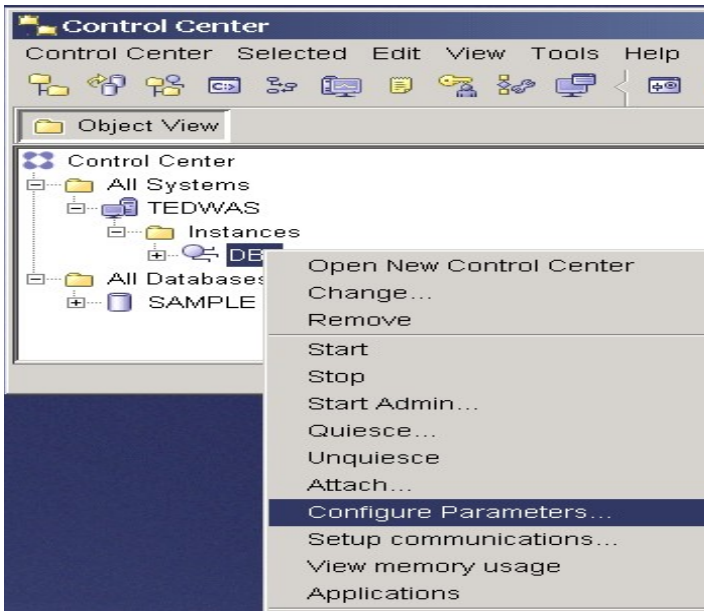


Figura 4.13 – Configurar o *dbm cfg* a partir do *Control Center*.

Depois de escolher Configurar Parâmetros, irá aparecer a imagem da Figura 4.14, com a lista de parâmetros *dbm cfg*.

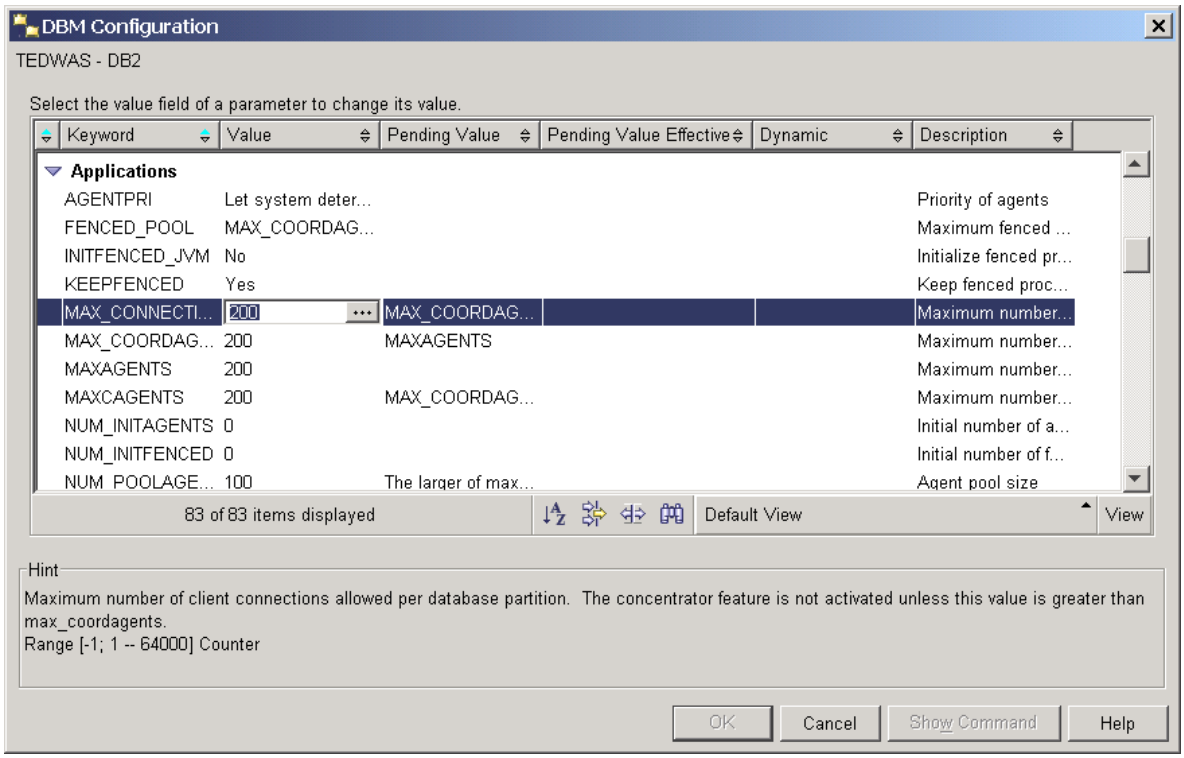


Figura 4.14 – Lista de parâmetros do dbm cfg

Muitos parâmetros são dinâmicos, o que significa que as suas mudanças têm efeito imediato. No entanto, alterações em alguns parâmetros podem exigir parar e iniciar a instância. A partir da linha de comandos, isto pode ser feito usando os comandos `db2stop` e `db2start`.

Antes de uma instância poder ser interrompida, todas as aplicações devem desconectar-se. Se desejar forçar a instância a parar, pode-se usar o comando `db2stop force`.

Uma instância pode também ser parada através do *Control Center* clicando na instância desejada e seleccionando Parar *(Stop)* ou Iniciar *(Start)*.

A Tabela 4.3 mostra alguns comandos úteis para gerir a *dbm cfg* a partir da linha de comandos.

Commando	**Descrição**
`db2 get dbm cfg`	Dá informação sobre o *dbm cfg*
`db2 update dbm cfg using <parameter_name> <value>`	Actualiza o valor de um parâmetro *dbm cfg*

Tabela 4.3 – Comandos para manipular o *dbm cfg*

4.1.3 Ficheiro de configuração da base de dados (db cfg)

O Ficheiro de configuração da base de dados *(database configuration file)* (db cfg) inclui parâmetros que afectam uma determinada base de dados. O ficheiro de configuração da base de dados pode ser visualizado ou alterado através da linha de comando ou através do DB2 *Control Center*.

Para trabalhar com o DB CFG do *Control Center*, seleccione a base de dados desejada da pasta das bases de dados do *Control Center*, clique no botão direito do rato para ver o menu e seleccione Configurar Parâmetros. Isto é ilustrado na Figura 4.15.

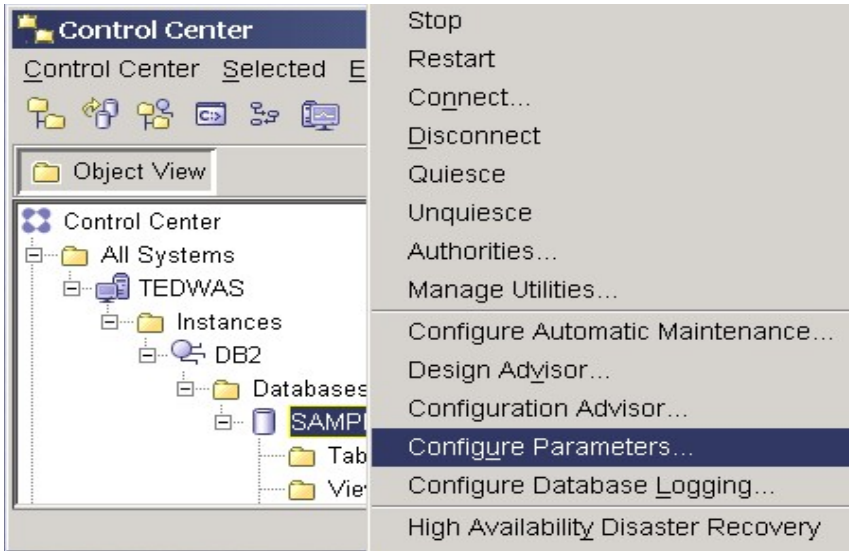


Figura 4.15 – Configurar o *db cfg* a partir do *Control Center*.

Depois de escolher Configurar Parâmetros, a imagem da Figura 4.16 será exibida com a lista de parâmetros db cfg.

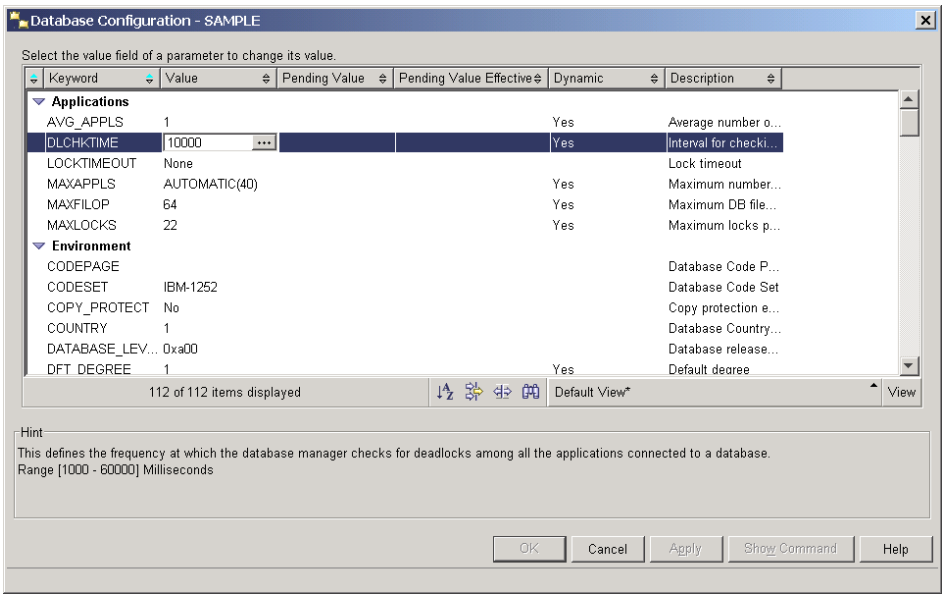


Figura 4.16 – O *db cfg*

A Tabela 4.4 mostra alguns comandos úteis para gerir o *db cfg* a partir da linha de comandos.

Comando	Descrição
`get db cfg for <nome_base_dados>`	Dá informação sobre o *dbm cfg* de uma determinada base de dados
`update db cfg for <nome_base_dados> using <nome_parametro> <value>`	Actualiza o valor de um parâmetro *db cfg*

Tabela 4.4 – Comandos para manipular o *db cfg*

4.1.4 Perfil de registo DB2

As variáveis do perfil de registo DB2 incluem parâmetros que podem ser específicos da plataforma e podem ser definidos a nível global (afectando todas as instâncias), ou a nível de instância (afectando uma instância específica).

A Tabela 4.5 mostra alguns comandos úteis para manipular o perfil de registo DB2

Commando	Descrição
`db2set -all`	Lista todas as variáveis de perfis de registo DB2 que estão actualmente definidas

`db2set -lr`	Lista todas as variáveis de perfis de registo DB2
`db2set <parametro>=<valor>`	Atribui um determinado valor a um parâmetro

Tabela 4.5 – Comandos para manipular os perfis de registo DB2

A tabela 4.6 mostra algumas variáveis de registo DB2 mais usadas

Variável de registo	Descrição
DB2COMM	Especifica quais os gestores de comunicação que estão iniciados quando o gestor da base de dados é iniciado.
DB2_EXTSECURITY	No Windows, impede o acesso não autorizado ao DB2, bloqueando o sistema de ficheiros
DB2_COPY_NAME	Guarda o nome da cópia DB2 actualmente a ser usada. Para mudar para uma outra cópia instalada, execute o comando pasta_instalaçaoDB2\bin\db2envars.bat. Esta variável não pode ser usada para este fim.

Tabela 4.6 – Variáveis de perfil de registo DB2 mais usadas

Por exemplo, para permitir a comunicação usando TCPIP, defina a variável registo DB2-COMM para TCPIP como é mostrado a seguir:

```
db2set db2comm=tcpip
```

4.2 O Servidor de Administração DB2 (DB2 Administration Server)

O *DB2 Administration Server* (DAS) é um processo *daemon* que corre no servidor DB2 para permitir que clientes remotos administrem graficamente o servidor de DB2. Existe apenas uma DAS por cada computador físico como mostra a Figura 4.16.

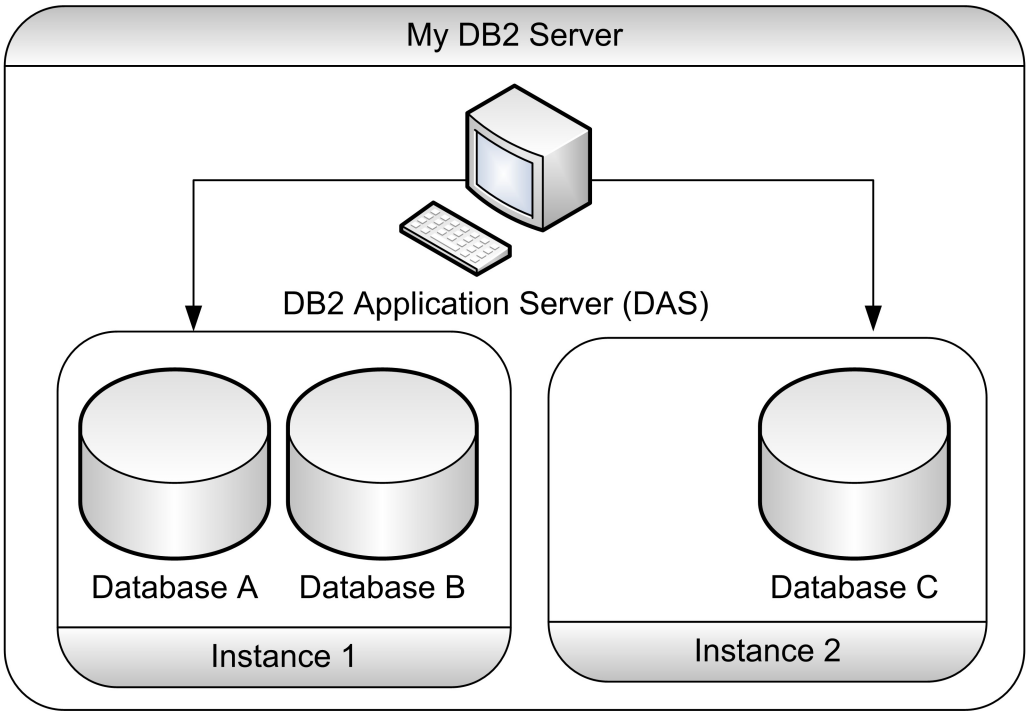


Figure 4.16 – O Servidor de Administração DB2 (DAS)

Quicklab #3 – Trabalhar com instâncias, bases de dados e configuração

Objectivo
Neste Quicklab, irá criar uma nova instância, base de dados, e mudar os parâmetros de configuração num servidor DB2 em Windows. Pode faze-lo através do Control Center ou do Command Windows. Neste quicklab iremos fornecer instruções para o fazer no Command Window.

Procedimento

1. Através do Command Window, cria uma nova instância chamada *newinst*
   ```
   db2icrt newinst
   ```

2. Na nova instância *newinst*, crie a base de dados *newdb* com os valores default
   ```
   set db2instance=newinst
   db2start
   db2 create database newdb
   ```

3. Liste todas as instância no seu servidor
   ```
   db2ilist
   ```

4. Mude para a instância DB2 e assegure-se que está realmente nessa instância
   ```
   set db2instance=db2
   db2 get instance
   ```

5. Mude o parâmetro dbm cfg FEDERATED para o YES de NO e verifique que a alteração ocorreu.
   ```
   db2 update dbm cfg using FEDERATED YES
   db2 force applications all
   db2 terminate
   db2stop
   db2start
   db2 get dbm cfg
   ```

6. Ligue-se à base SAMPLE com o ID/psw que usou para fazer login no sistema operativo
   db2 connect to sample user <userID> using <psw>

7. Reveja quantas aplicações estão a correr na instância corrente
   db2 list applications show detail

8. Abra outra DB2 Command Window e liga-se novamente à base de dados SAMPLE sem especificar o ID/psw. Reveja quantas aplicações tem agora.
   db2 connect to sample
   db2 list applications

9. Force uma das DB2 Command Windows
   db2 force application (<application handle obtained from the db2 list applications command for application name “db2bp.exe”>)

10. Elimine a instância *newinst*
    db2idrop newinst

11. Elimine e recrie o DAS, e arranque-o.
    db2admin stop
    db2admin drop
    db2admin create
    db2admin start

12. Atribua a variável DB2COMM do DB2 Registry a tcpip e npipe na sua instância
    db2set db2comm=tcpip,npipe
    db2stop
    db2start

13. Não atribua a variável de registro DB2COMM
    db2set db2comm=
    db2stop
    db2start

14. Verique o valor actual do db cdf parânetro LOGSECOND , e altera-o para o valor de 5 e verifique o novo valor
    db2 connect to sample
    db2 get db cfg
    db2 update db cfg using LOGSECOND 5
    db2 get db cfg

5

Capítulo 5 – Ferramentas DB2

Neste capítulo, descrevemos algumas das ferramentas que pode usar com o DB2. A elipse vermelha na Figura 5.1 mostra a área de foco deste capítulo.

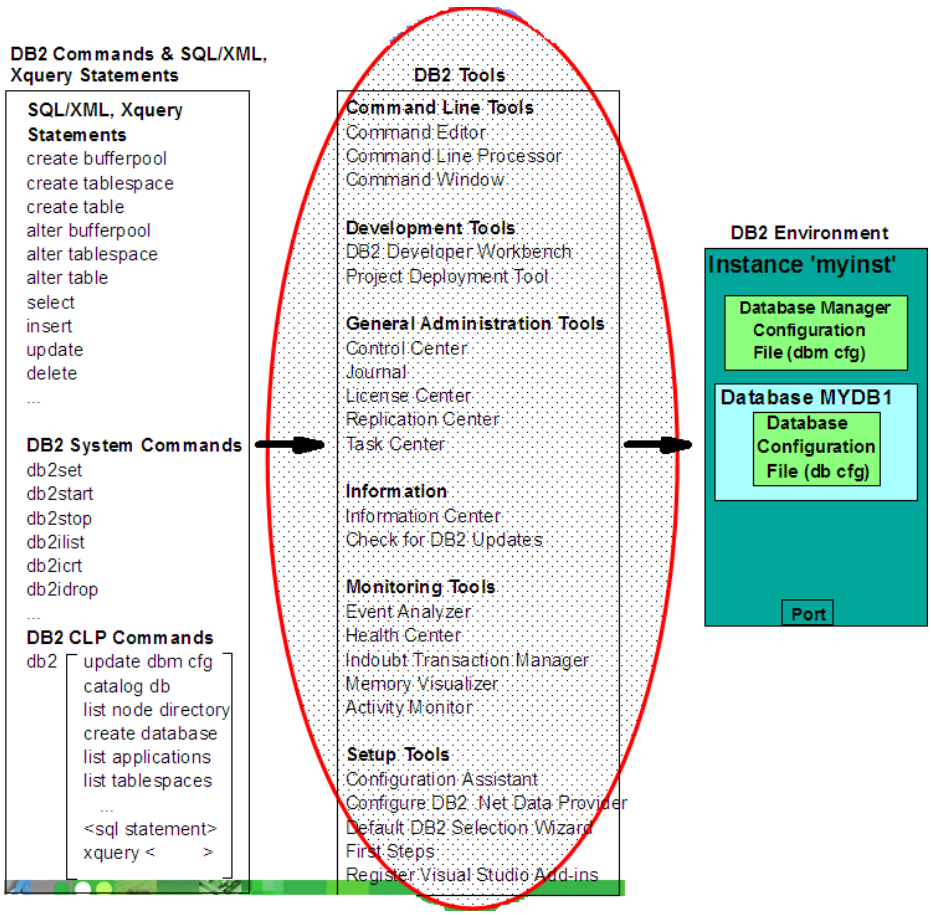


Figura 5.1 – "The Big Picture": As ferramentas DB2

Nota:

Veja os vídeos de apresentação das ferramentas de DB2 e DB2 *scripting* nestes links:
http://www.channeldb2.com/video/video/show?id=807741:Video:4202
http://www.channeldb2.com/video/video/show?id=807741:Video:4182

A Figura 5.2 lista todas as ferramentas DB2 disponíveis a partir do menu IBM DB2. A maioria destas ferramentas é igual para Linux e Windows.

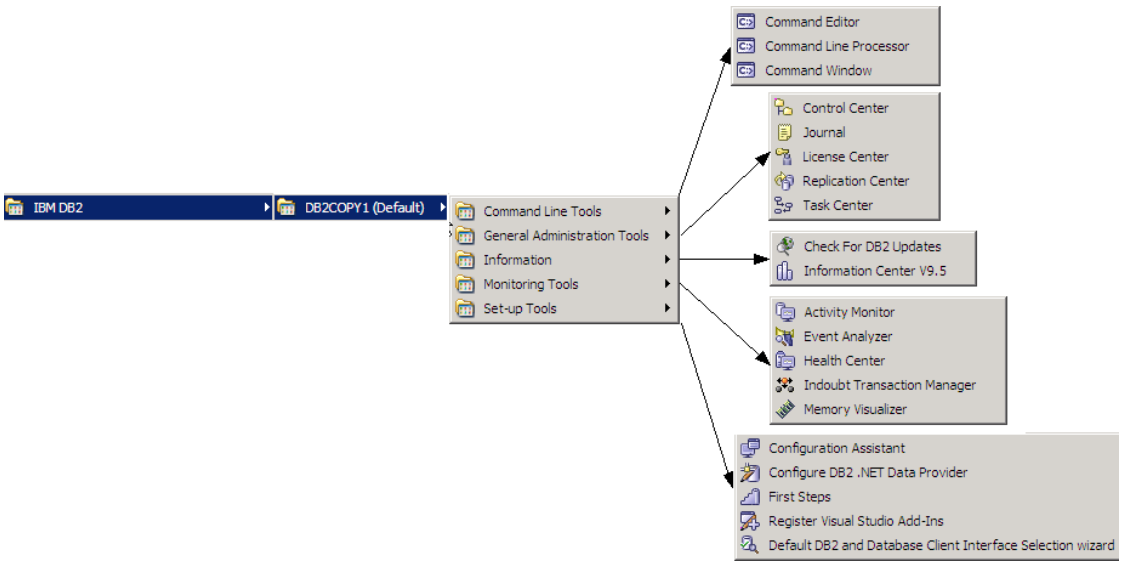


Figura 5.2 – Ferramentas DB2 a partir do menu de início IBM DB2

A Tabela 5.1 expõe uma lista de comandos de atalho que podem ser usados para iniciar as ferramentas mais populares quer em Linux quer em Windows.

Nome da Ferramenta	Comando
Command Editor	db2ce
Command Line processor	db2
Command Window (Only on Windows platforms)	db2cmd
Control Center	db2cc
Task Center	db2tc

Health Center	db2hc
Configuration Assistant	db2ca
First Steps	db2fs

Tabela 5.1 – Comandos de atalho para algumas ferramentas DB2

5.1 *Control Center*

A ferramenta principal do DB2 para a administração de bases de dados é o *Control Center* (Centro de Controlo), ilustrado na Figura 5.3.

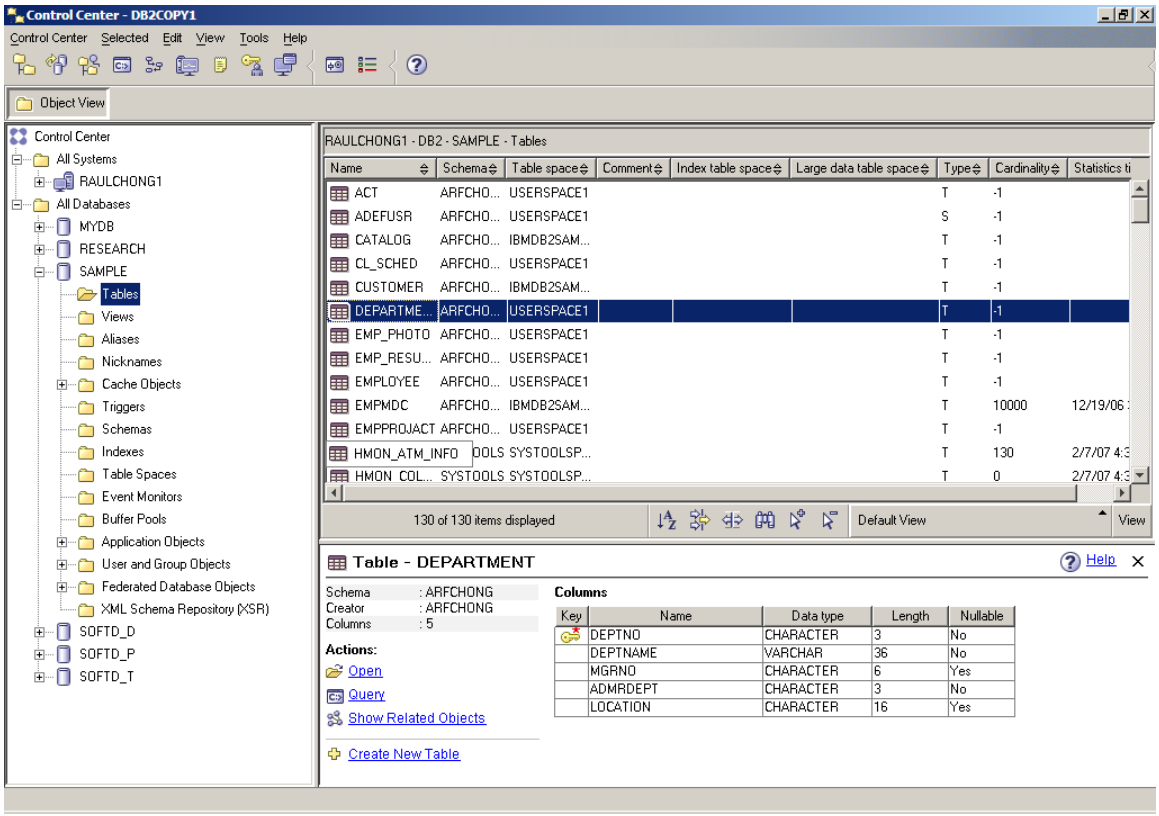


Figura 5.3 – *DB2 Control Center*

O *Control Center* é uma ferramenta de administração centralizada que permite:

- Ver todos os sistemas, instâncias, bases de dados, e objectos de bases de dados;
- Criar, modificar e gerir bases de dados e respectivos objectos;
- Executar outras ferramentas gráficas do DB2.

O painel da esquerda disponibiliza uma hierarquia visual dos objectos das bases de dados no seu sistema(s), fornecendo uma "directoria" para Tabelas, *Views*, etc. Quando se faz duplo-clique com o rato numa das directorias (por exemplo, a directoria Tabelas, como mostra a Figura 5.3), o painel da direita irá listar todos os objectos relacionados, neste caso, todas as tabelas associadas com a base de dados SAMPLE. Se seleccionar uma dada tabela no painel da direita (topo), o painel inferior da direita irá mostrar informação específica sobre essa tabela.

Clicando com o botão direito do rato nas diferentes directorias/objectos da árvore de objectos, teremos menus aplicáveis às respectivas directorias/objectos. Por exemplo, clicar com o botão direito numa instância e escolher "*Configure Parameters*" (configuração de parâmetros) irá permitir a visualização e alteração do ficheiro de configuração do gestor da base de dados. O ambiente DB2 e a configuração de parâmetros são discutidos com mais detalhe no Capítulo 4 – Ambiente do DB2.

A primeira vez que executar o *Control Center*, ser-lhe-á pedido para escolher que tipo de visualização deseja utilizar. A escolha desta visualização determina que tipos de opções e de objectos da base de dados estarão expostos. A Figura 5.4 mostra este passo de configuração.

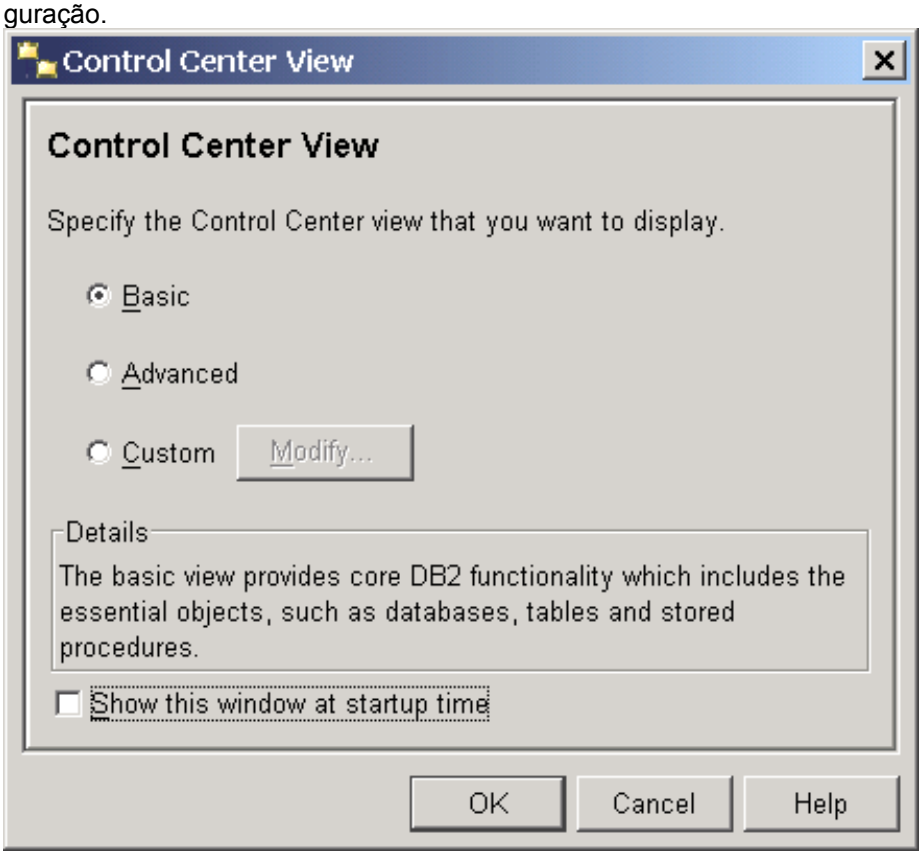
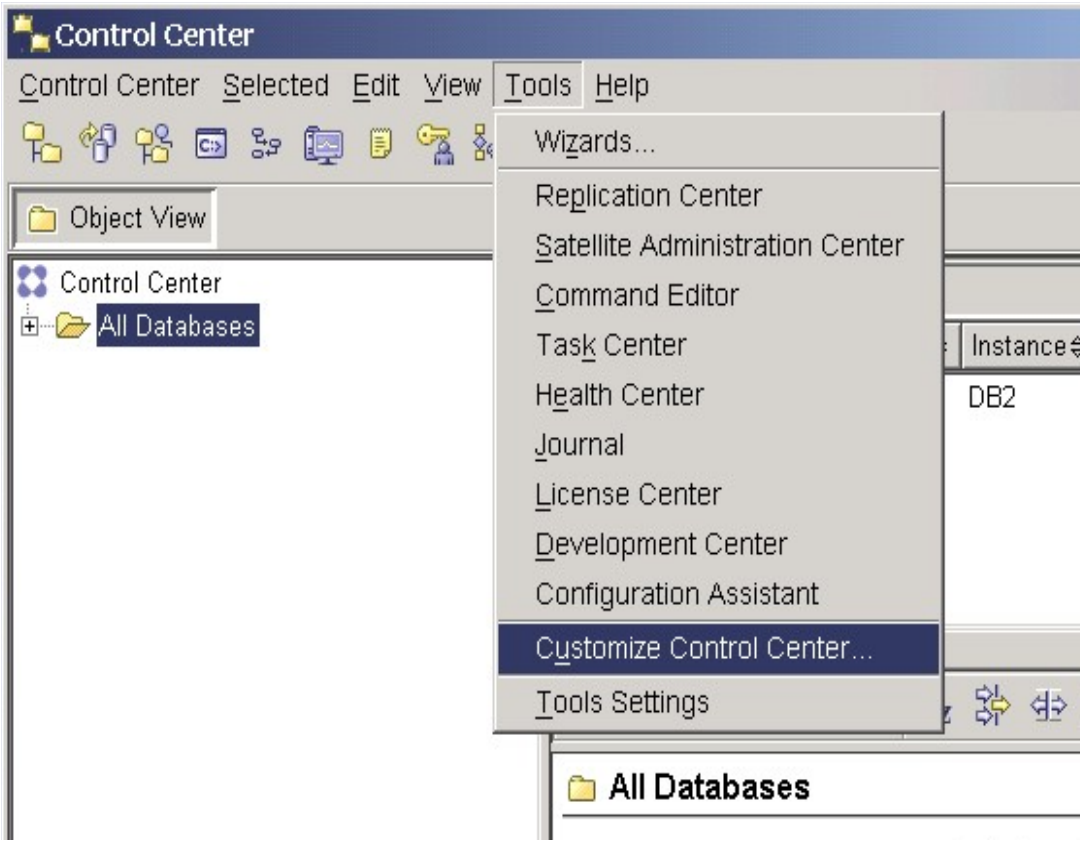


Figura 5.4 – Janela de configuração de visualizações do *Control Center* do DB2

A visualização *Basic* abrange as funcionalidades básicas do DB2.
A visualização *Advanced* mostra mais opções e mais funcionalidades.

A visualização *Custom* permite a configuração das funcionalidades, objectos e opções que deseja ver.

Para reinvocar a janela de configuração de visualizações do Centro de Controlo, seleccione a opção "*Customize Control Center*" no menu de ferramentas como mostra a Figura 5.5.



F**igura 5.5 – Configurar o *Control Center***

Executar o *Control Center*
Existem muitas maneiras de iniciar o *Control Center*:

7. Navegando a partir do menu Iniciar do Windows
8. Executando o comando *db2cc* na linha de comandos
9. Clicando no ícone  do *Control Center* na barra de ferramentas de qualquer outra ferramenta gráfica do DB2
10. A partir da barra "*system tray*" no ícone DB2 como mostra a Figura 5.6 (Clicar com o botão direito no ícone verde do DB2 e seleccionar a opção *DB2 Control Center*)

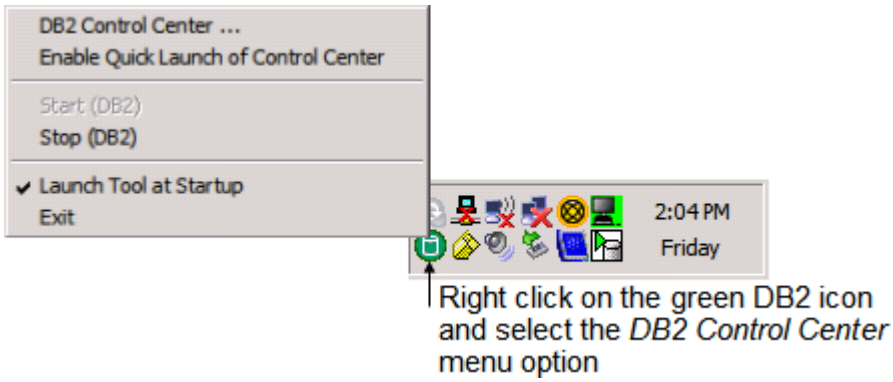


Figura 5.6 – Iniciar o *Control Center* a partir da barra "*system tray*" do Windows

5.2 *Command Editor*

Através do *DB2 Command Editor* (Editor de Comandos), pode executar comandos DB2, consultas SQL e XQuery, analisar o plano de execução de uma consulta, e ver ou actualizar resultados de consultas.

A Figura 5.7 mostra o *Command Editor* com a descrição dos seus elementos.

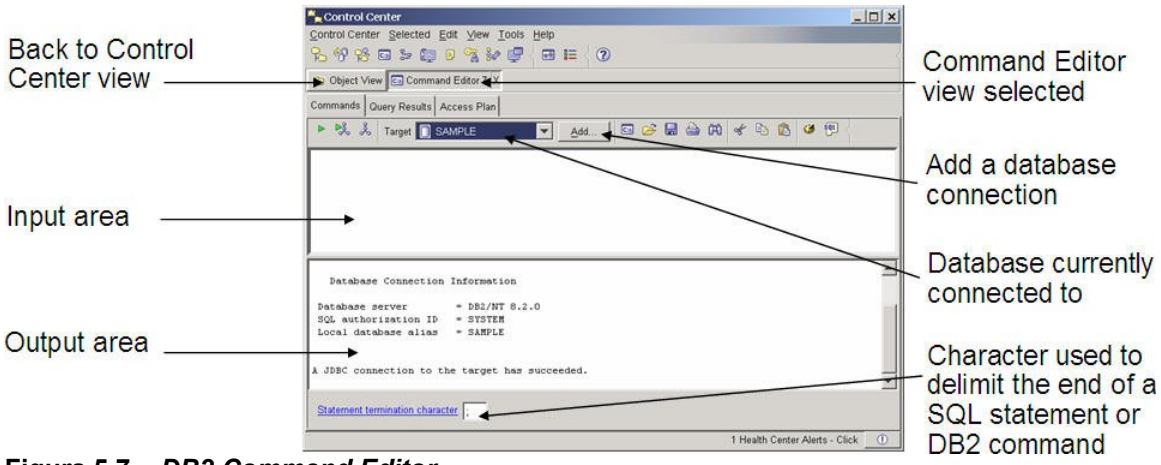


Figura 5.7 – *DB2 Command Editor*

Na área de *input*, pode inserir múltiplos comandos desde que se coloque em cada comando o carácter de terminação. Se carregar no botão Executar (ver Figura 5.8), os comandos serão executados um após o outro. A ligação à base de dados deve existir de maneira a executar os comandos SQL, no entanto, um desses comandos pode ser o comando de ligação.

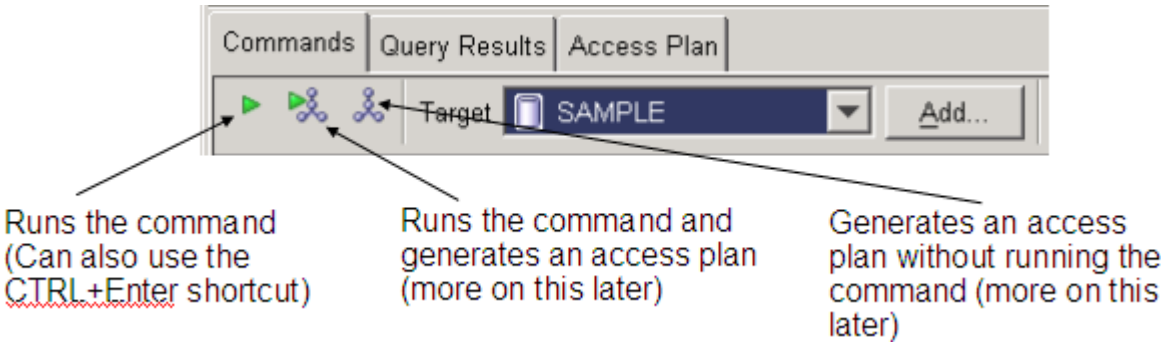


Figura 5.8 – *Command Editor* – Separador de Comandos

Executar o *Command Editor*
Pode inciar o *Command Editor* de várias maneiras:

2. A partir do menu Iniciar do Windows:
Iniciar -> Programas -> IBM DB2 -> DB2COPY1 (Default) -> Command Line Tools -> Command Editor
3. A partir da linha de comandos, escrevendo `db2ce`
4. A partir do menu Ferramentas (*tools*) no *Control Center*
5. Embebido no *Control Center*
    1. Clicando com o botão direito no ícone da base de dados SAMPLE na árvore de objectos do *Control Center* e seleccionando o item de menu *Query*
    2. Cada vez que um objecto utilizado frequentemente em *queries* (bases de dados, tabelas, etc.) é seleccionado, pode lançar o *Command Editor* clicando no atalho *Query* no painel *Object Detail*.
6. A partir do *Control Center*, clicando no ícone do *Command Editor*  na barra de ferramentas conforme a Figura 5.9

Figura 5.9 – Ícone do *Command Editor* no *Control Center*

Adicionar uma ligação à base de dados
Para adicionar uma ligação a uma base de dados, clique no botão *Add* (Ver Figura 5.7). Uma janela tal como mostra a Figura 5.10 irá aparecer.

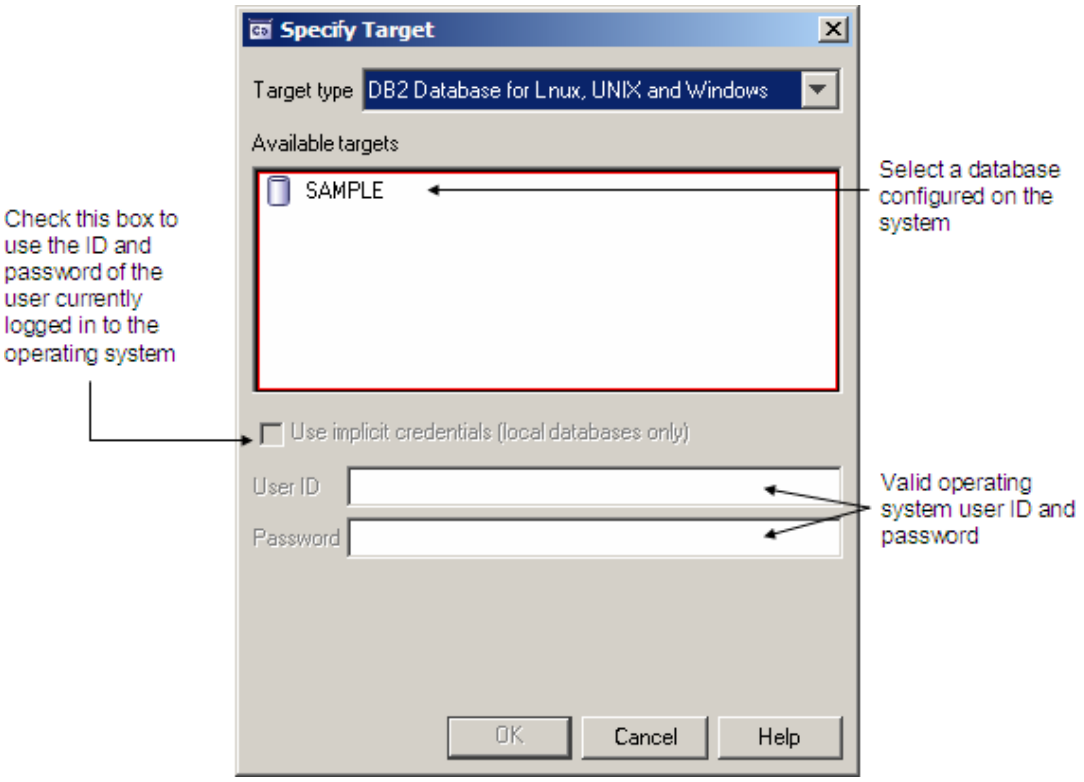


Figura 5.10 – Adicionar uma ligação à base de dados

5.3 Assistente SQL

Se não está familiarizado com a linguagem SQL e gostaria de usar um assistente para gerar código SQL, o Assistente SQL está disponível a partir do *Command Editor* para o ajudar. Como mostra a Figura 5.11, podemos inicia-lo a partir do *Command Editor* clicando no último ícone da barra com o símbolo SQL (como mostra a figura)

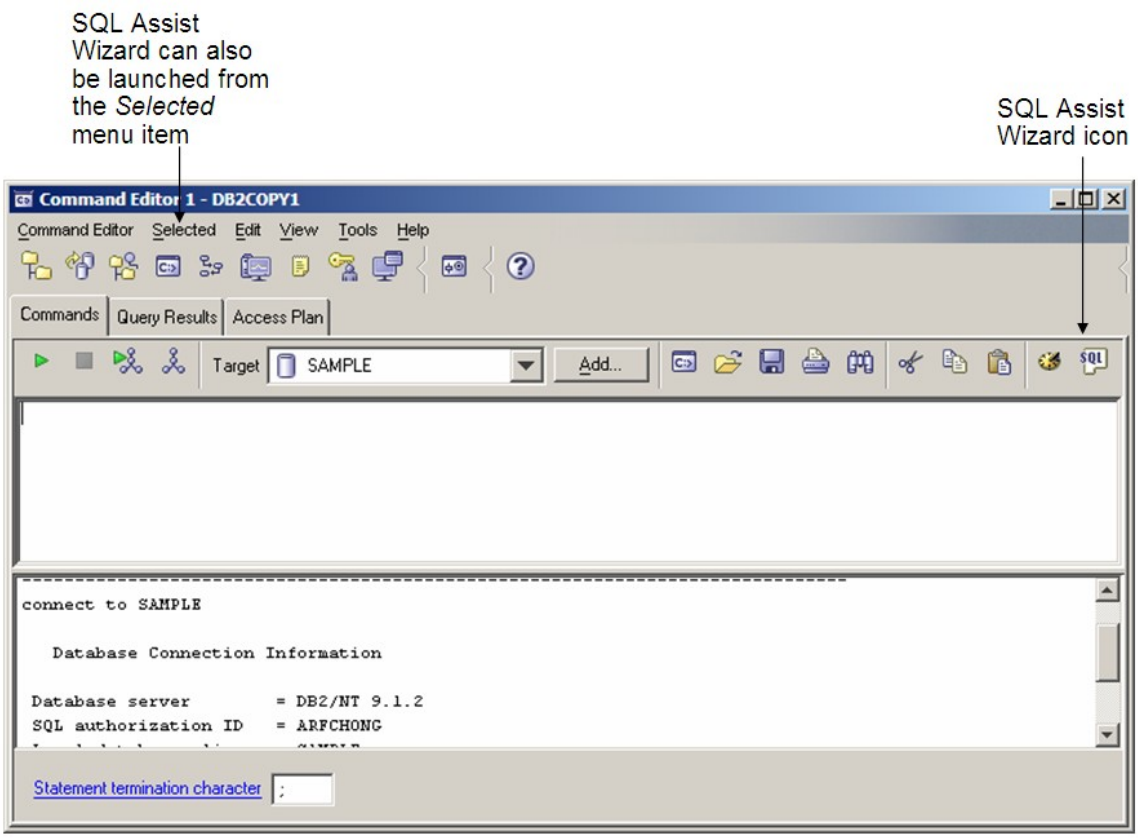


Figura 5.11 – Invocar o Assistente SQL

A Figura 5.12 mostra o Assistente SQL. É muito simples de utilizar. Primeiro, escolhe o tipo de comando a executar (SELECT, INSERT, UPDATE, DELETE). Dependendo da sua escolha, diferentes opções aparecerão. Em baixo pode ver que o comando é construído à medida que selecciona as diferentes opções no Assistente.

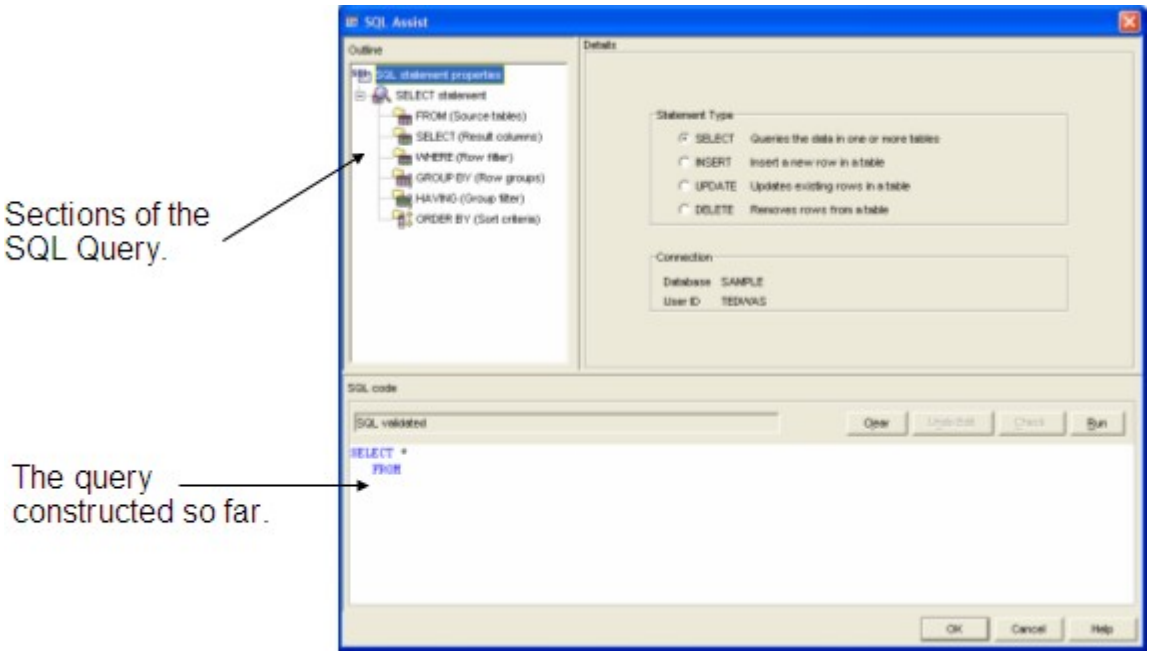


Figura 5.12 – Assistente SQL

5.4 Botão *Show SQL*

A maioria das ferramentas gráficas em DB2 permitem que reveja o comando actual gerado como resultado da utilização do Assistente. Para ver clique em *Show SQL* na ferramenta que está a utilizar, como mostra a Figura 5.14

Figura 5.13 – O botão *Show SQL*

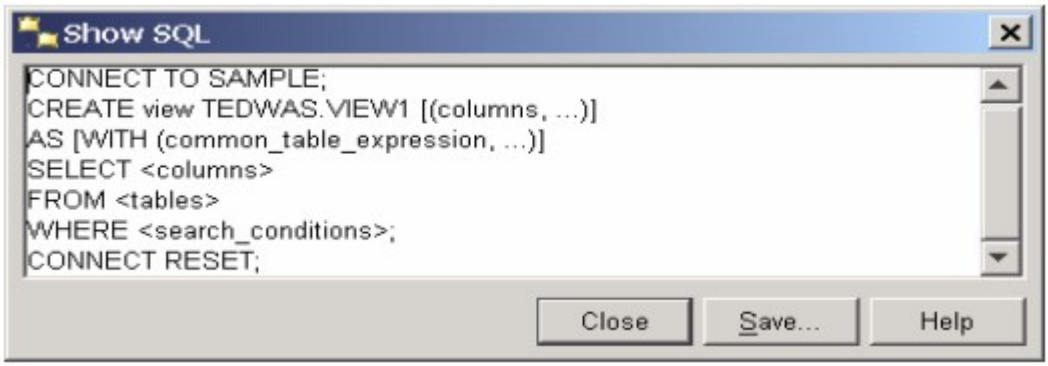


Figura 5.14 – O output do botão *Show SQL*

A capacidade de permitir rever a consulta SQL é muito útil para aprender a sintaxe SQL, e para guardar esta num ficheiro para uso futuro. Também pode construir scripts reutilizando estes comandos gerados.

Quicklab #4: Preencher a base de dados EXPRESS utilizando scripts

Objectivo

Neste *Quicklab*, irá preencher a base de dados EXPRESS através do *Command Editor* utilizando dois scripts adicionais.

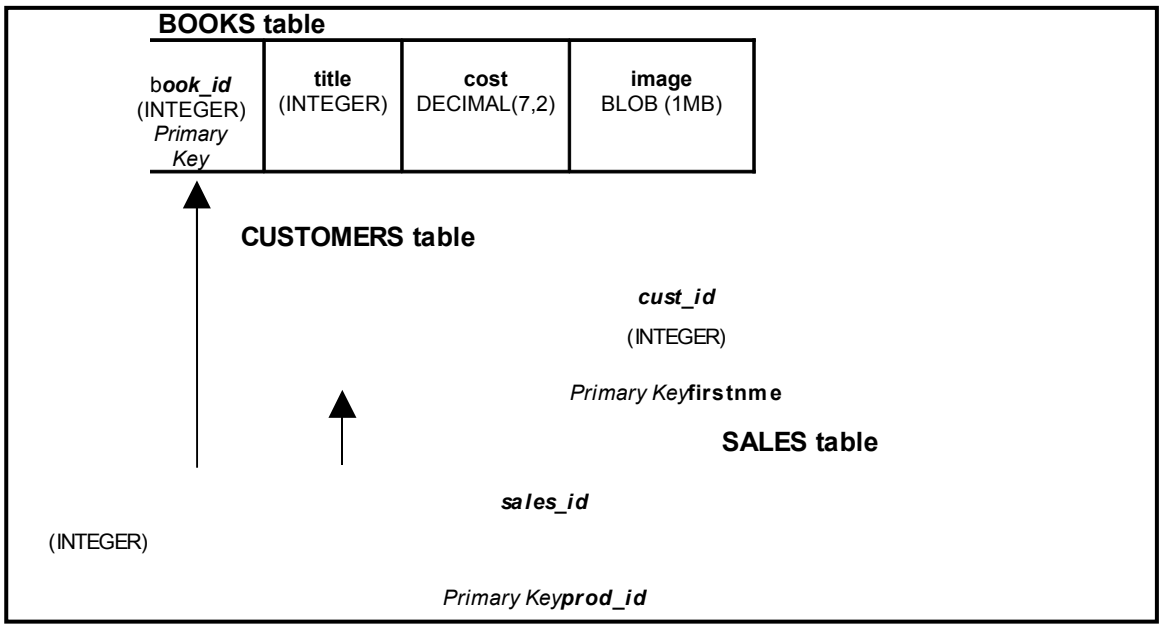
Procedimentos

- No *Quicklab* #2 criou a base de dados EXPRESS; agora precisará de preenchê-la com algumas tabelas e alguns dados. Para seu conforto, dois scripts, chamados `quicklab4.db2` e `quicklab4.dat` foram criados para fazê-lo por si. O script `quicklab4.db2` contém os comandos utilizados na criação das tabelas logo tem que ser corrido primeiro. O script `quicklab4.dat` contém os comandos para inserção de dados nas tabelas. Ambos os scripts podem ser encontrados no ficheiro .zip *quicklabs* que acompanha este livro. Para correr estes scripts, abra o *Command Editor*. Assegure-se que a nova base de dados que criar é seleccionada na barra de ferramentas na lista *drop-down*. Se a nova base de dados não aparecer na lista, adicione uma conexão com o botão *Add*.

- Clique no menu *Selected → Open* no *Command Editor* e navegue até à directoria onde se encontram armazenados os scripts. Seleccione o ficheiro *quicklab4.db2* e clique no botão *OK*. Os conteúdos do ficheiro deverão aparecer agora na área *input* do *Command Editor*. Clique no botão *Run* para executar o script. Verifique se não existem erros encontrados após a execução do script.

- Repita o passo (2) para o ficheiro `quicklab4.dat`.

A nova base de dados que criou é para uma livraria *online* muito simples. A tabela BOOKS contém toda a informação sobre os livros que existem na loja. A tabela COSTUMERS contém informação acerca dos clientes da loja. Finalmente, a tabela SALES contém os dados das vendas. Quando um cliente compra um livro, é criado um registo na tabela SALES. O diagrama em baixo mostra as relações entre as tabelas.


BOOKS table
book_id
(INTEGER)
Primary
Key
title
(INTEGER)
cost
DECIMAL(7,2)
image
BLOB (1MB)
CUSTOMERS table
cust_id
(INTEGER)
Primary Keyfirstnme
SALES table
sales_id
(INTEGER)
Primary Keyprod_id

5.5 Scripting

É sempre útil ser capaz de criar ficheiros de script para executar vários comandos DB2 ou consultas SQL repetidamente. Por exemplo, um administrador de bases de dados (DBA) pode querer correr um determinado script todos os dias que lhe devolva o número de entradas em determinadas tabelas.

Existem duas formas de scripting:

- Scripts SQL
- Scripts do Sistema Operativo (shell).

5.5.1 Scripts SQL

Os scripts SQL incluem consultas (*queries*) e comandos. Estes scripts são relativamente simples de compreender e são independentes da plataforma. No entanto não suportam variáveis ou parâmetros.

Por exemplo, os seguintes comandos estão guardados no ficheiro script1.db2

```
CONNECT TO EXPRESS;
CREATE TABLE user1.mytable
          ( col1 INTEGER NOT NULL,
            col2 VARCHAR(40),
            col3 DECIMAL(9,2));
SELECT * FROM user1.mytable FETCH
FIRST 10 ROWS ONLY;
COMMIT;
```

Ficheiro script1.db2

Neste script, todos os comandos são comandos SQL, e cada comando está delimitado pelo carácter de terminação, neste caso um ponto e vírgula. O nome do ficheiro não necessita de ter a extensão .db2, pois qualquer uma pode ser usada.

Executar Scripts SQL
Um script SQL pode ser executado quer a partir do *Command Editor* quer pela *DB2 Command Window* em Windows, ou pela shell de Linux. Para executar o script anterior a partir da *Command Window* ou da shell de Linux, pode usar o seguinte comando:

```
db2 -t -v -f script1.db2 -z script1.log
```

ou:

```
db2 –tvf script1.db2 –z script1.log
```

Neste comando:

-t indica que as *queries* usam o carácter de terminação predefinido (ponto e vírgula)
-v indica o modo *verbose*; fazendo com seja mostrada a execução do comando que está a correr
-f indica que o nome do ficheiro especificado depois desta flag é o ficheiro de script.
-z indica que o nome que vem de seguida será o nome do ficheiro onde será armazenado o output da execução para analisar posteriormente (é opcional, mas recomendado)

Depois da flag **-t** ser usada e não for indicado nenhum separador, assume-se que o ponto e vírgula como o delimitador das *queries*. Poderão surgir situações em que outro delimitador seja necessário. Por exemplo, um script contendo código SQL PL necessita de um caracter diferente porque o ponto e vírgula são usados na definição de objectos de SQL PL para terminar certos procedimentos.

Por exemplo, no ficheiro abaixo com o nome "functions.db2", contendo a linguagem de definição de dados (DDL) necessária para criar uma função, o ponto e vírgula é necessário para terminar a sintaxe do SELECT necessário no interior da função. Para o comando CREATE FUNCTION o delimitador utilizado é o ponto de exclamação (!). Se usarmos um ponto e vírgula como delimitador haveria um conflito de execução, que seria reportado como um erro pelo DB2.

```
CREATE FUNCTION f1()
   SELECT …   ;
   …
END!
```

Ficheiro functions.db2

Para informar o DB2 que está a ser usado um delimitador diferente, utilize a flag **–d**, seguida pelo delimitador desejado:

```
db2 –td! –v –f functions.db2 –z functions.log
```

A descrição de outras flags que podem ser utilizadas, podem ser obtidas executando o seguinte comando na *Command Window* ou Linux shell:

```
db2 list command options
```

5.5.2 Scripts do Sistema Operativo (shell)

Os scripts do sistema operativo fornecem uma óptima flexibilidade e poder, uma vez que possibilitam adicionar lógica de programação. São dependentes da plataforma mas suportam parâmetros e variáveis. Abaixo encontra-se um exemplo de um script simples para Windows.

```
set DBPATH=c:
set DBNAME=PRODEXPR
set MEMORY=25
db2 CREATE DATABASE %DBNAME% ON %DBPATH% AUTOCONFIGURE USING
      MEM_PERCENT %MEMORY% APPLY DB AND DBM
db2 CONNECT TO %DBNAME% USER %1 USING %2
del schema.log triggers.log app_objects.log
db2 set schema user1
db2 –t –v –f schema.db2 –z schema.log
db2 –td@ -v –f triggers.db2 –z triggers.log
db2 –td@ -v –f functions.db2 –z functions.log
```

Ficheiro create_database.bat

Para executar este script do sistema operativo a partir da linha de comandos, use o seguinte comando:

```
create_database.bat db2admin ibmdb2
```

No Windows a extensão “bat” indica ao sistema operativo que aquele é um executável de *batch*.

Em Linux, é necessário executar o comando `chmod +x` para indicar ao sistema operativo que se trata de um executável. Após isso, o procedimento é o mesmo.

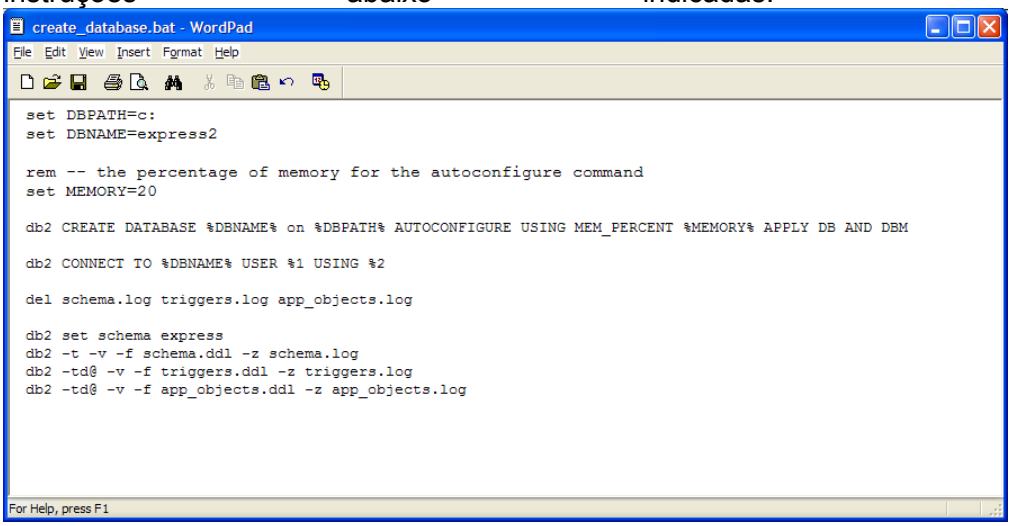
Quicklab #5: Criar um script de instalação para a base de dados EXPRESS

Objectivo

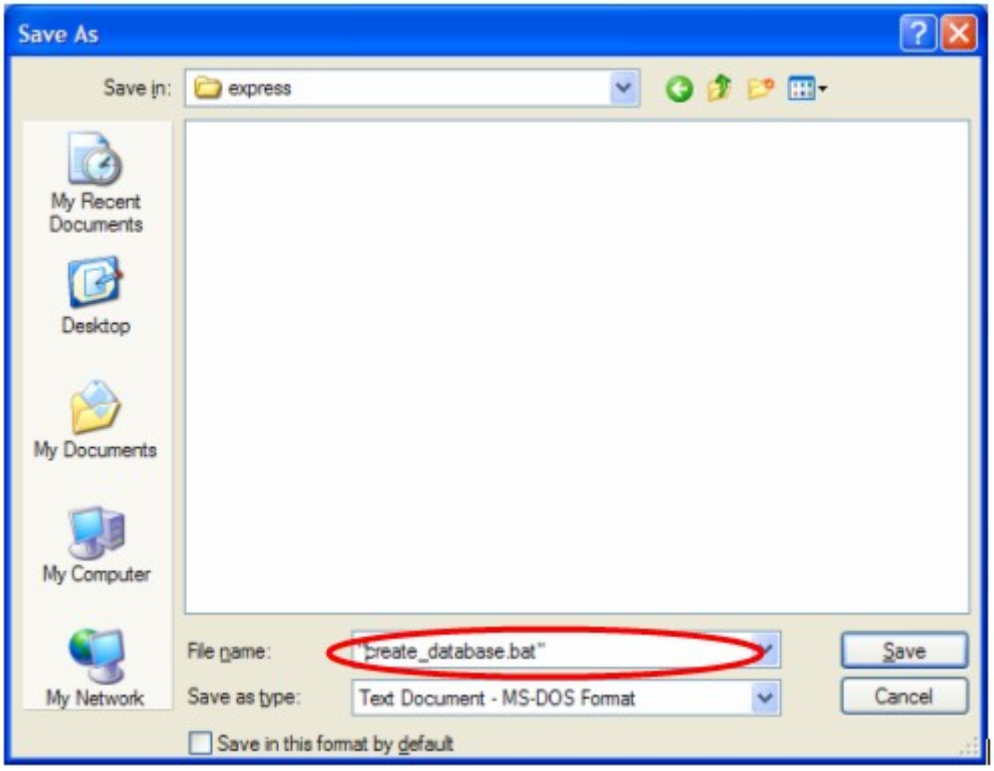
Os scripts são um mecanismo poderoso de executar tarefas repetitivas como recolha de estatísticas, *backups* e *deployments*. Os scripts do sistema operativo têm a vantagem de suportar parâmetros, tornando-os flexíveis. Neste Quicklab, irá criar um script do sistema operativo para fazer o *deploy* da base de dados EXPRESS. O script irá evocar scripts SQL posteriores sobre os objectos da base de dados. De forma a poupar espaço, este Quicklab mostra scripts e comandos específicos para Windows. Se prefere Linux, assegure-se que faz as alterações apropriadas às seguintes expressões.

Procedimentos

- Abrir um editor de texto, como o Bloco de Notas ou o Wordpad e insira as instruções abaixo indicadas.

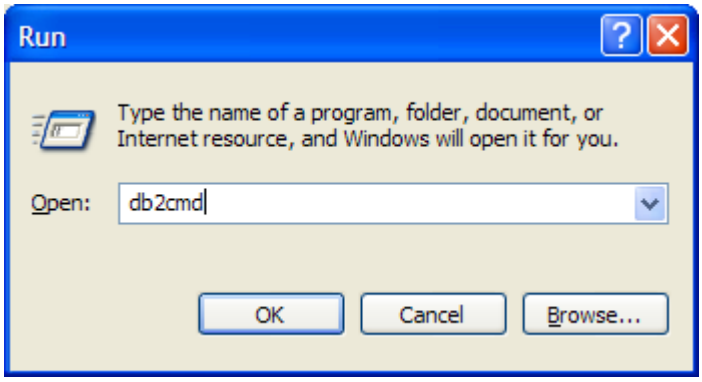


- Guardar o ficheiro numa directoria e atribuir-lhe o nome `create_database.bat`. Na janela *Save As* assegure-se que escolhe a opção *MS-DOS Format* (no Wordpad). Se guardar noutro formato o Wordpad poderá introduzir caracteres invisíveis que podem causar problemas na execução do script. Assegure-se também que coloca o nome do ficheiro entre aspas, como mostra na figura abaixo, para que o Windows não coloque a extensão .TXT.



- Para correr scripts que interagem com o DB2, terá que ter um ambiente de linha de comandos DB2. Para tal, vá a Iniciar > Programas > IBM DB2 > DB2COPY1 (default) > *Command Line Tools* > *Command Window*.

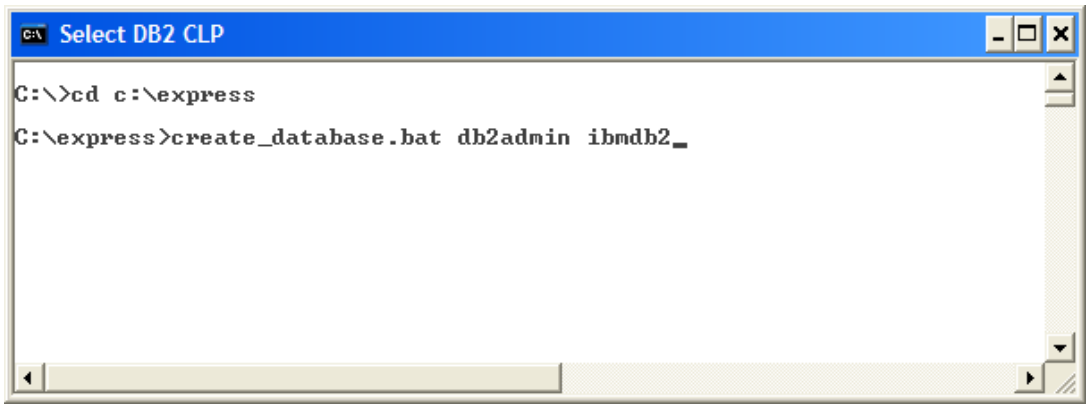
  Alternativamente, pode ir a Iniciar > Executar, e escrever `db2cmd`.



- Depois para correr o script, escreva os seguintes comandos:

```
cd C:\express
```

```
create_database.bat db2admin ibmdb2
```



- Demore o tempo necessário até se familiarizar com o script que criou. Compreenda o que cada linha significa.

- Tente responder às seguintes questões:

    1. Onde é estabelecida a ligação à base de dados?

    2. O que significa `%1` e `%2`?

    3. O que faz a seguinte linha? Onde é usada? Para que serve?
    `SET DBPATH=C:`

    4. O que faz a seguinte linha de código?
    `del schema.log, triggers.log, app_objects.log`

    5. O que acontece quando o script é executado sem parâmetros?

    6. Porque é que os scripts SQL usados não contêm o comando CONNECT TO? Como é que se ligam à base de dados?

5.6 *Task Center*

A ferramenta gráfica *Task Center* (Centro de Tarefas) permite-lhe criar tarefas; um conjunto de operações como, por exemplo, correr comandos DB2, comandos do sistema operativo, ou scripts. Acções subsequentes podem ser executadas se a tarefa falhar ou se for bem sucedida. Por exemplo, se uma tarefa envolver o *backup* de uma base de dados importante às 03h00, caso seja bem sucedida, um email pode ser enviado ao administrador da base de dados (DBA) fornecendo essa informação. Por outro lado, se o *backup* falhar, o *Task Center* pode enviar um sinal para o telemóvel do DBA. A Figura 5.15 mostra o *Task Center*.

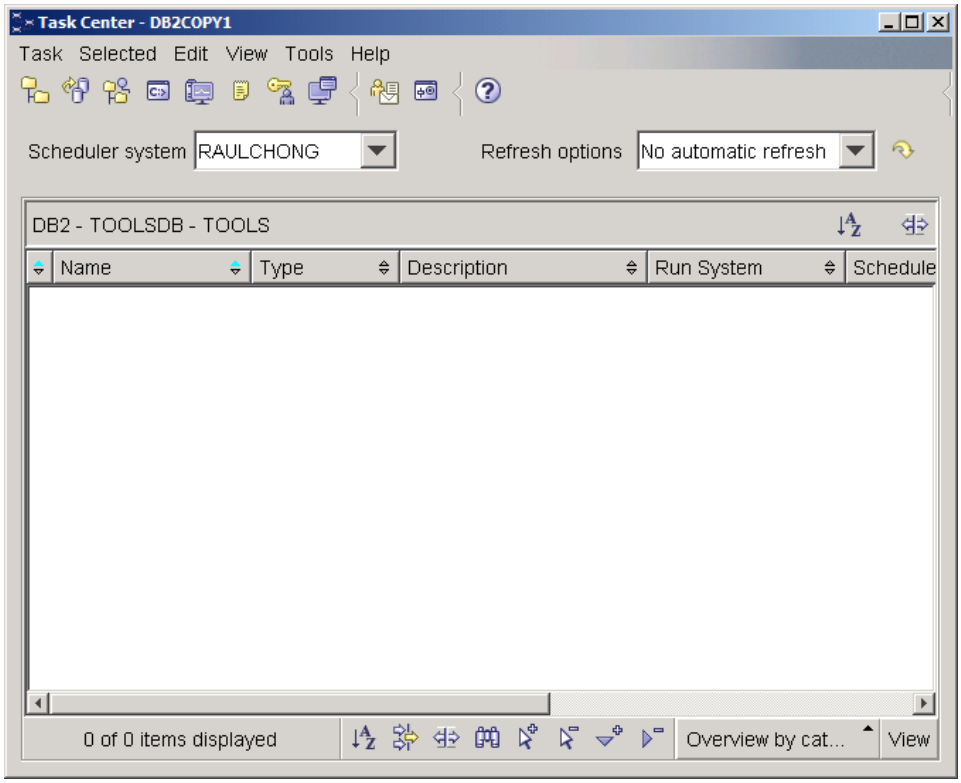


Figura 5.15 – ***Task Center***

5.6.1 Base de dados do *Tools Catalog* (Catálogo de Ferramentas)

Todos os detalhes sobre as suas tarefas e agendamento de tarefas são guardados numa base de dados DB2 separada. Esta base de dados deve existir por forma a agendar tarefas. Para criar uma base de dados do *Tools Catalog*, use o seguinte comando:

```
CREATE TOOLS CATALOG systools CREATE NEW DATABASE toolsdb
```

No exemplo anterior, `systools` é o nome do *schema* de todas as tabelas da base de dados, e o nome da base de dados é `toolsdb`. Iremos falar mais sobre *schemas* no Capítulo 8 – Trabalhar com Objectos da Base de Dados.

Executar o *Task Center*
Pode executar o *Task Center* a partir do *Control Center* clicando em *Tools* > *Task Center*, como mostra a Figura 5.16. Alternativamente, pode abrir esta ferramenta a partir do menu Iniciar do Windows: *Iniciar* > *Programas* > *IBM DB2* > *DB2COPY1* > *General Administration Tools* > *Task Center*

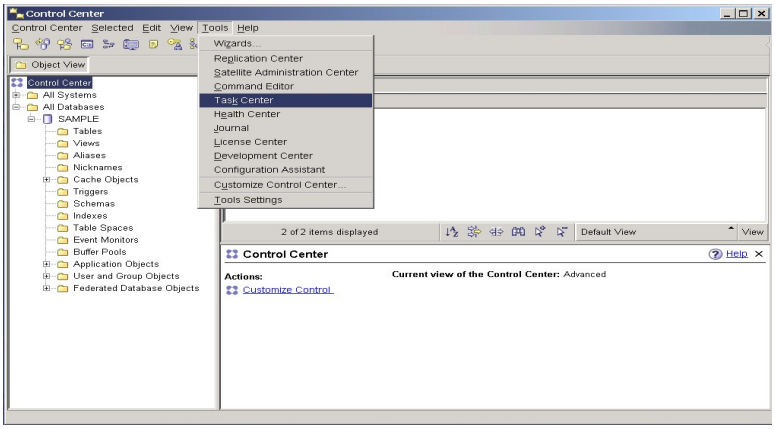


Figura 5.16 – Abrir o *Task Center*

Agendar com o *Task Center*
Qualquer tipo de script pode ser agendado usando o *Task Center* (quer sejam ou não criadas a partir de uma ferramenta gráfica do DB2). As tarefas são executadas na data agendada no sistema onde foi criada a base de dados do catálogo de ferramentas. Encorajamos a que explore o *Task Center* por si mesmo.

5.7 Jornal

A ferramenta gráfica Jornal DB2 fornece ao administrador de base de dados um jornal de actividades num formato *online*. A Figura 5.17 mostra o Jornal e a Tabela 5.2 a informação que pode obter a partir do Jornal.

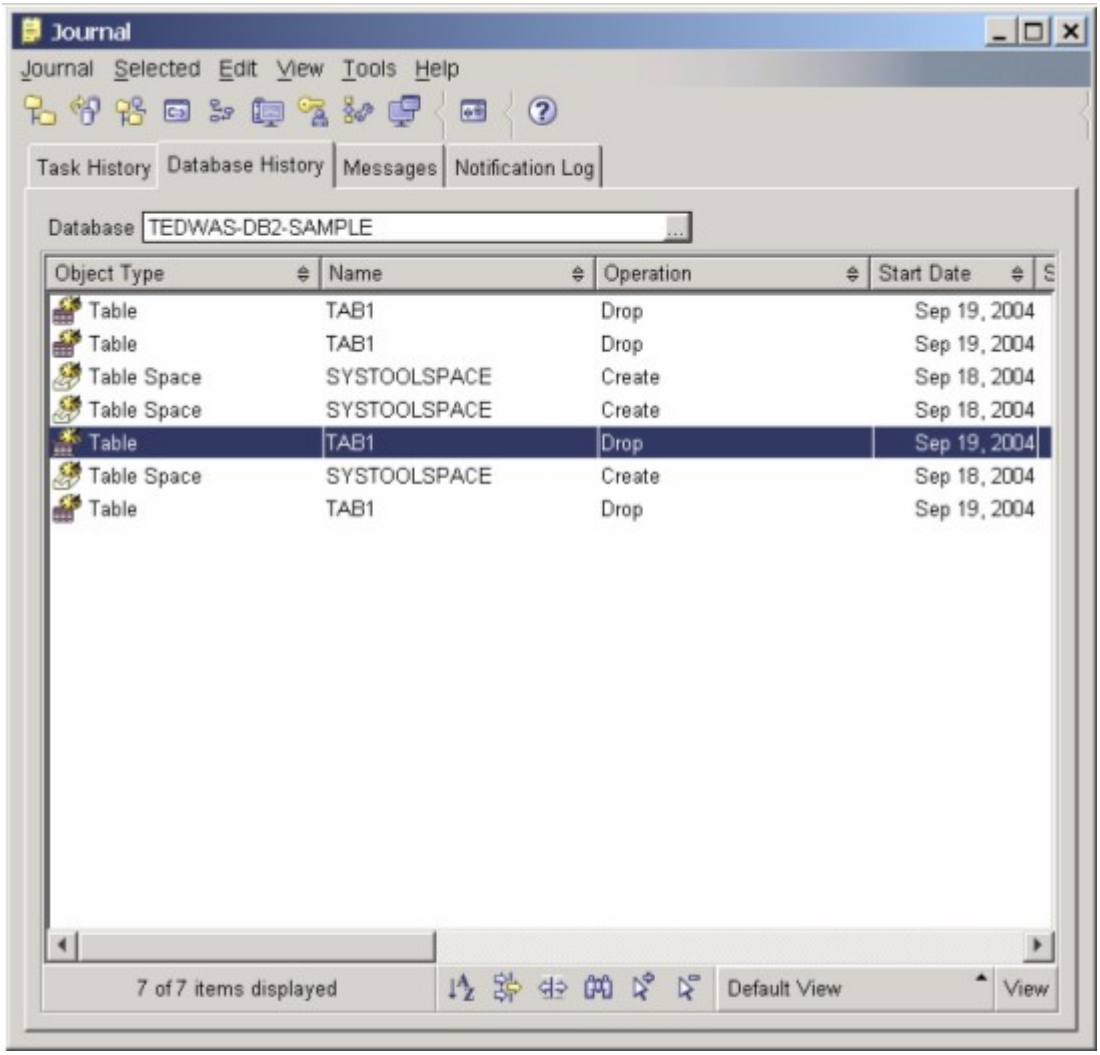


Figura 5.17 – O Jornal

Tipo de Informação	Descrição
Task History	Todas as tarefas agendadas e o seu estado.
Database History	Um registo das actividades executadas sobre as bases de dados (*backups*, *restores*, REORGs, etc.).
Message	Histórico de mensagens devolvidas pelas ferramentas DB2. Isto é útil se desejar rever e comparar mensagens de erro passadas, ou se fechar uma caixa de mensagem rápido de mais ou por acidente.
Notification Log	Contem mensagens do sistema. Erros críticos são guardados aqui.

Tabela 5.2 – Informação fornecida a partir do Jornal

Iniciar o Jornal

Pode abrir o Jornal desde o *Control Center*, clicando em *Tools* > *Journal*, como mostra a Figura 5.18. Alternativamente, pode iniciar esta ferramenta a partir do menu Iniciar do Windows: *Iniciar* > *Programas* > *IBM DB2* > *DB2COPY1* > *General Administration Tools* > *Journal.*

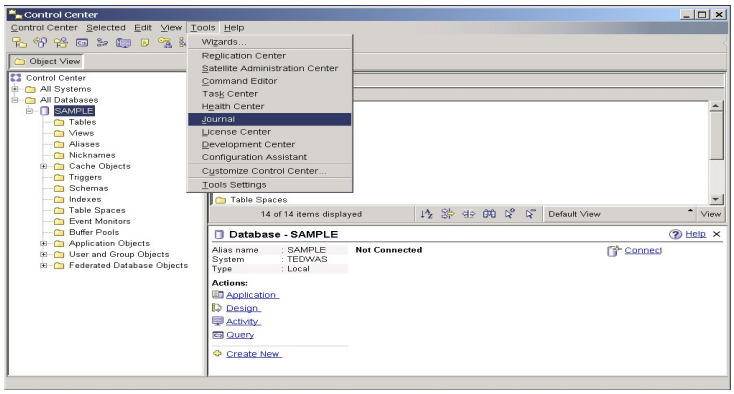


Figura 5.18 – Abrindo o Jornal

5.8 *Health Monitor*

O *Health Monitor* é o agente pré-definido que corre juntamente com o motor DB2, monitorizando todos os aspectos da "saúde" da base de dados (memória, gestão de espaço, automatização de tarefas definidas, etc.). Quando algum parâmetro do DB2 está a funcionar fora dos parâmetros definidos, uma excepção é lançada e mostrada ao administrador da base de dados. Existem três tipos de alertas de estado:

1. *Attention*: um estado anormal.
2. *Warning*: um estado não-crítico que não requer atenção imediata, mas que indica um sistema não-óptimo.
3. *Alarm*: uma condição crítica que requer atenção imediata.

O *Health Monitor* pode ser ligado ou desligado usando o parâmetro de configuração do gestor de bases de dados HEALTH_MON.

5.8.1 *Health Center*

O *Health Center* é a ferramenta gráfica para interagir com o *Health Monitor*. O *Health Center* liberta alertas ao nível do sistema por instância, base de dados, e tabela. A Figura 5.19 mostra o *Health Center*.

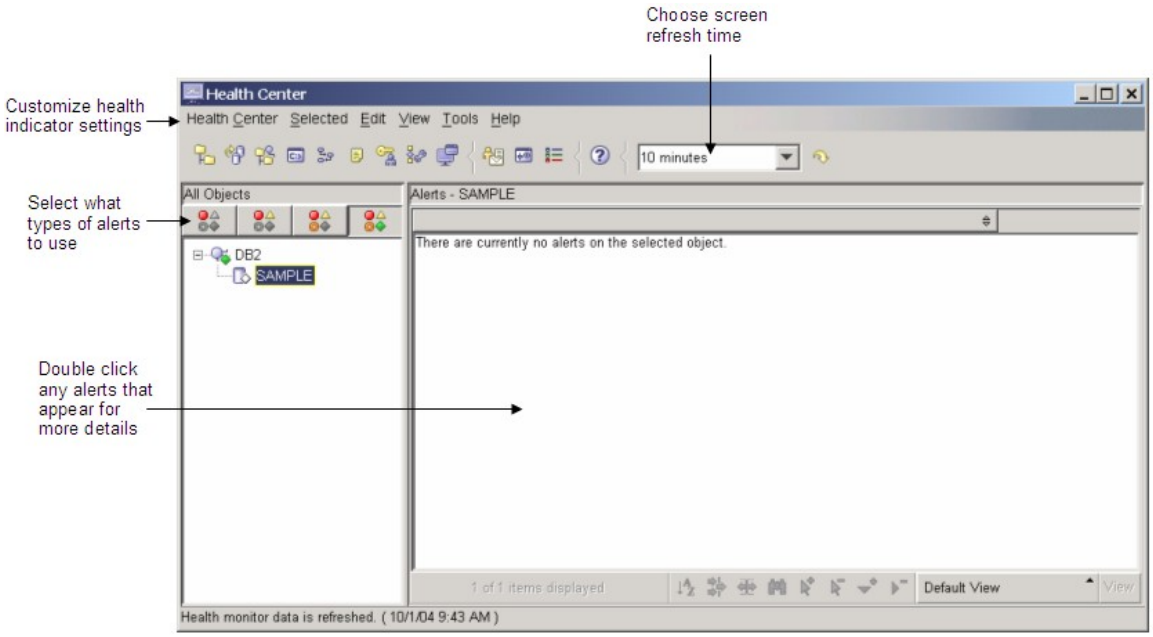


Figura 5.19 – O *Health Center*

Abrir o *Health Center*

Pode abrir o *Health Center* desde o *Control Center* através do menu *Tools* > *Health Center* (ver Figura 5.20). Alternativamente, pode abrir o *Health Center* a partir do menu Iniciar do Windows: *Iniciar* > *Programas* > *IBM DB2* > *DB2COPY1* >*Monitoring Tools* > *Health Center*

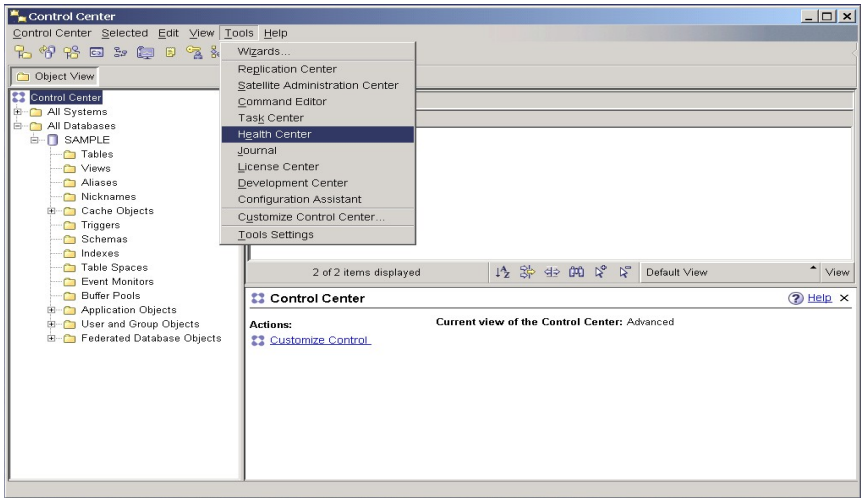


Figura 5.20 – Iniciar o *Health Center*

Configurar as notificações de alerta

Uma vez o *Health Center* aberto, pode configurar as notificações de alerta clicando no menu *Heath Center* > *Configure* > *Alert Notification* como mostra a Figura 5.21. As notifica-

ções de alerta permite-lhe inserir e-mails ou números de *pager* de pessoas que serão contactadas caso o alerta despolete.

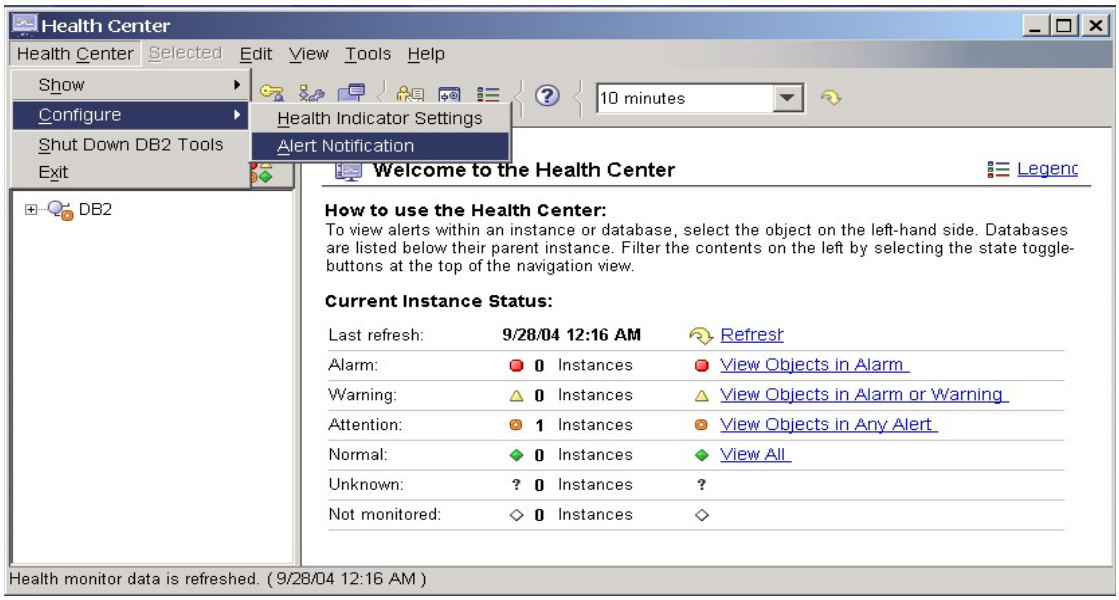


Figura 5.21 – Notificações de Alerta

Parte II – Aprender DB2: Administração de Base de Dados

6

Capítulo 6 – Arquitectura do DB2

Neste capítulo vamos discutir brevemente a arquitetura do DB2:

- O modelo de processamento do DB2
- O modelo de memória do DB2
- O modelo de armazenamento do DB2

Nota:

Para mais informação sobre a arquitectura do DB2, veja este video:
http://www.channeldb2.com/video/video/show?id=807741:Video:4482

6.1 O modelo de processamento do DB2

A figura 6.1 retrata o modelo de processamento do DB2. Nesta figura, os rectângulos representam processos e as elipses representam threads. O processo principal DB2 é chamado de *db2sysc*. Sob este processo existem várias *threads*, uma das principais é também chamada de *db2sysc*. Esta é a *thread* principal que cria outras *threads*. Quando uma aplicação remota tenta ligar-se ao servidor utilizando um comando SQL CONNECT, os *listeners* remotos para o protocolo de comunicação irão receber esse pedido e entrar em contacto com um agente DB2 (*db2agent*). Um agente DB2 é como um pequeno trabalhador que realiza operações em nome do DB2. Quando o pedido for local, ou seja, sendo executados no mesmo servidor do DB2, os passos são muito semelhantes, só que um agente *db2ipccm* lida com o pedido em vez da *thread db2tcpcm*. Em alguns casos, como quando o paralelismo está activo, um *db2agent* pode produzir outros agentes que aparecem como *threads db2agntp*. Outros agentes, ilustrados na figura como *db2pfchr, db2loggr, db2dlock* também podem ser utilizados para fins diferentes. Os processos mais comuns são descritos na tabela 6.1, e as *threads* mais comuns estão descritas na Tabela 6.2.

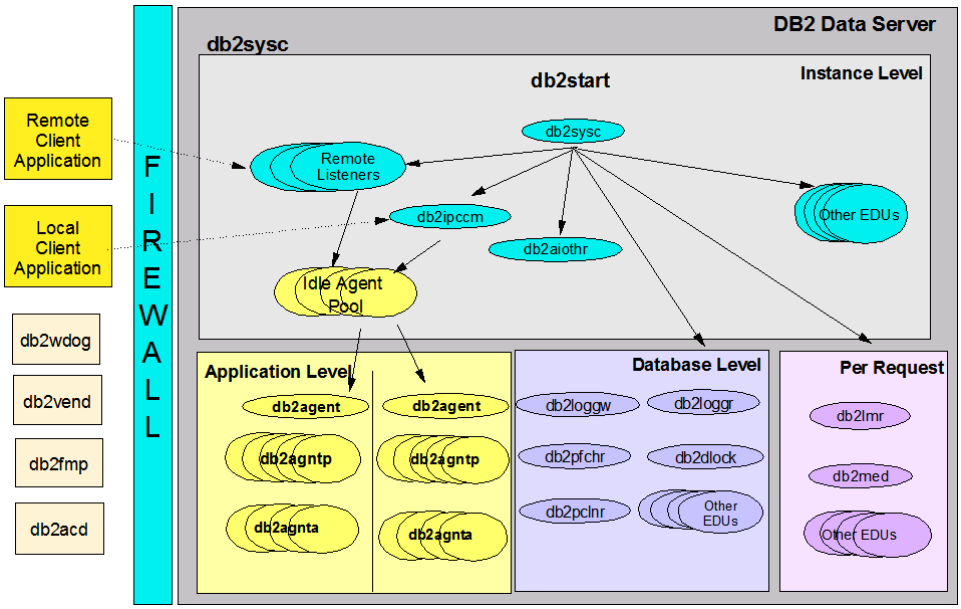


Figura 6.1 – O modelo de processamento do DB2

Nome do processo	Descrição
db2sysc (Linux) db2syscs (Win)	O principal sistema do DB2, o motor (*engine*) da base de dados. No DB2 9.5, existe apenas um processo principal *multi-threaded* para toda a partição. Todas as Engine Dispatchable Units (EDUs) são *threads* dentro deste processo. Sem esse processo, o servidor de base de dados não pode funcionar.
db2acd	O *deamon* de computação autónoma. É utilizado para executar tarefas automáticas do lado do cliente, como o monitor de estado, utilitários de manutenção automática, e o *scheduler* do administrador. Este processo era chamado anteriormente *db2hmon*.
db2wdog	O processo observador (*watchdog*) do DB2. Este processo é parente do processo do motor principal, *db2sysc* e limpa recursos se o db2sysc terminar anormalmente.
db2vend	O *fenced vendor process* que foi introduzido no *DB2 9.5*. Trata-se de uma nova funcionalidade que permite executar código externo neste processo fora do motor. Por código externo, entende-se programas não-IBM que interagem com DB2; por exemplo, armazenamento de logs pode ser por um programa externo especificando o parametro da rotina de saída que aponta para este código.

db2fmp	Processos vedados que correm código do utilizador no servidor fora da *firewall* para ambos os procedimentos armazenados e funções definidas pelo utilizador. Este processo substitui os processos *db2udf* e *db2dari* que foram utilizados nas versões anteriores do DB2.

Tabela 6.1 – Processos comuns no DB2

Nome da *Thread*	**Descrição**
db2sysc	A *thread* controladora do sistema. Esta é responsável pelo arranque, paragem e gestão da instância actual
db2tcpcm	*Listener* de comunicação TCP/IP
db2agent	Agente coordenador que realiza operações na base de dados em nome das aplicações (pelo menos 1 por cada conexão, dependendo se o *Connection Concentrator* está activo).
db2agntp	O sub agente activo é ramificado se a flag INTRA_PARALLEL está definida como YES. Irá realizar operações de dados para a aplicação. O *db2agent* irá coordenar o trabalho entre os diferentes sub agentes *db2agntp*.
db2pfchr	*Prefetcher* de dados I/O assincrono (NUM_IOSERVERS)
db2pclnr	Escritor de dados I/O assincrono (NUM_IOCLEANERS)

Tabela 6.2 – *Threads* DB2 comuns

6.2 Modelo de memória DB2

O modelo de memória DB2 consiste em diferentes áreas: memória ao nível da instância, nível de base de dados, da aplicação e do agente, como indica a Figura 6.2. Não iremos explicar em detalhe cada uma das diferentes áreas em memória neste livro, apenas fornecer uma breve prespectiva.

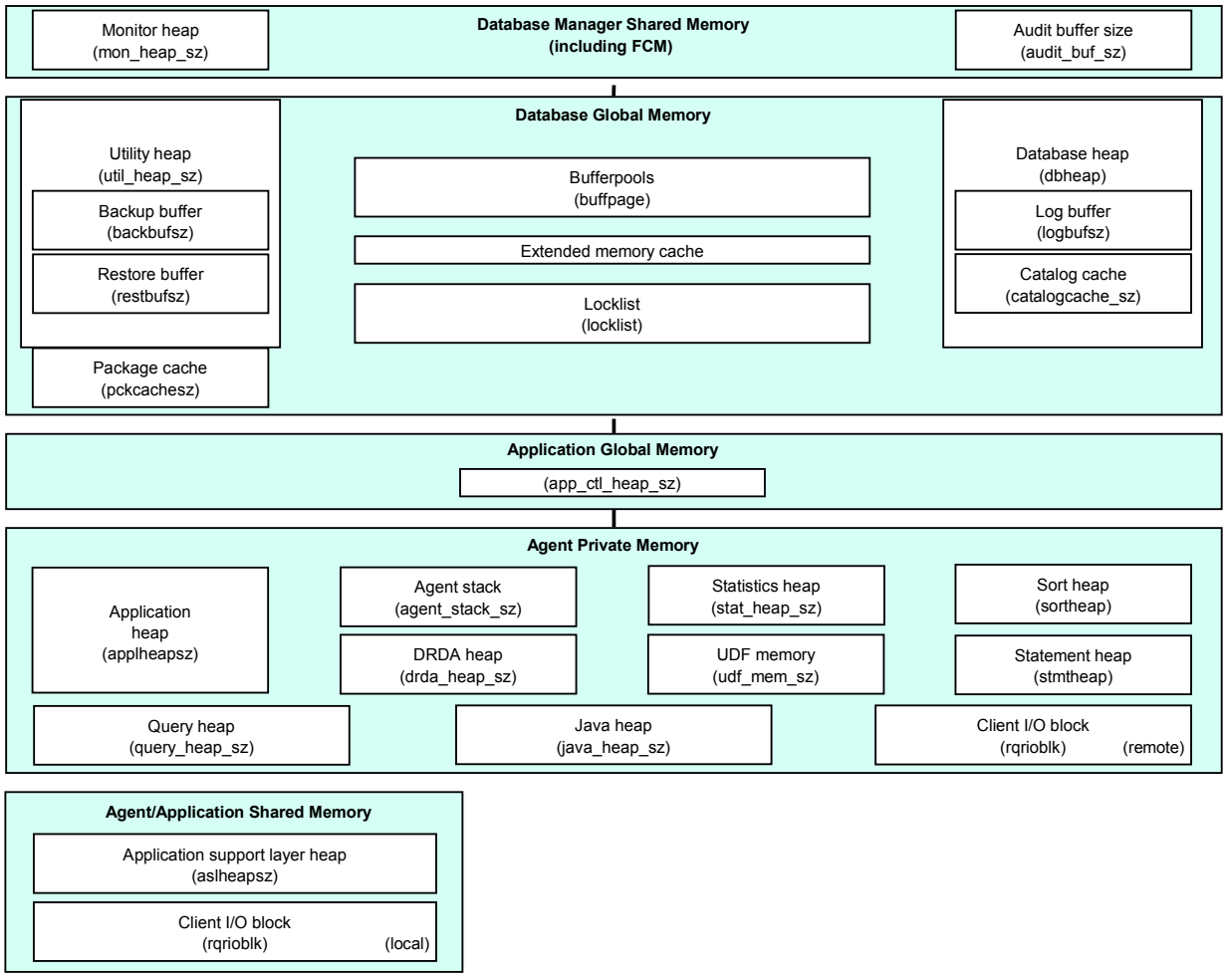


Figura 6.2 – Modelo de memória do DB2

Quando uma instância é iniciada, o gestor de memória partilhada da base de dados é alocado. Isso normalmente não requer muito espaço. Quando se ligar pela primeira vez, a uma base de dados, a Memória Global da Base de Dados (*Database Global Memory*) é alocada. Neste bloco, o *buffer pool* é um dos componentes mais importantes, especialmente para melhorar as consultas. O tamanho dos *buffer pools* irá determinar o tamanho da Memória Global da Base de Dados.

O agente privado de memória é a memória usada por cada agente DB2. Sem usar concentrações de ligação, cada ligação requer um agente. Tipicamente, um agente pode usar cerca de 3 a 5 MB. Com concentração de ligações, várias ligações podem utilizar um agente, reduzindo a necessidade de mais memória física.

6.3 Modelo de armazenamento DB2

Nesta secção iremos descrever os seguintes conceitos:

- Páginas e *Extents*
- *Buffer pool*
- *Table space*

6.3.1 Páginas e *Extents*

Uma página é a unidade mínima de armazenamento no DB2. Tamanhos de páginas permitidos são: 4K, 8k, 16K e 32kB. Um *extent* é um conjunto de páginas. Trabalhar com uma só página de cada vez é dispendioso em termos de desempenho, por isso, o DB2 trabalha com um *extent* de cada vez. O tamanho da página e o tamanho do *extent* são definidos quando se trabalha com *buffer pools* e *table spaces* como veremos nas próximas secções.

6.3.2 *Buffer pools*

Um *buffer pool* é um verdadeiro *cache* de memória para dados de tabelas e indexes. Além disso, melhora o desempenho através da redução directa sequêncial de I/O e promove leituras assíncronas (pre-fetching), bem como escritas. Ou seja, o DB2 prevê as páginas que serão necessárias e pré-recupera-as a partir do disco para a área do *buffer pool* a fim de que estejam prontos para uso.

Buffer pools estão alocados em unidades de memória de 4K, 8k, 16K, e 32kB páginas. Deve haver no mínimo um *buffer pool* por base de dados, e, pelo menos, uma correspondência *buffer pool* para a *table space* de um determinado tamanho de página.

Criar um *Buffer Pool*
Para criar um *buffer pool* pode usar o comando `CREATE BUFFERPOOL`. Em alternativa, usando o *Control Center*, pode seleccionar com o botão direito sobre a pasta do *buffer pool* dentro de um determinada base de dados e escolher `Create` como mostra na Figura 6.3

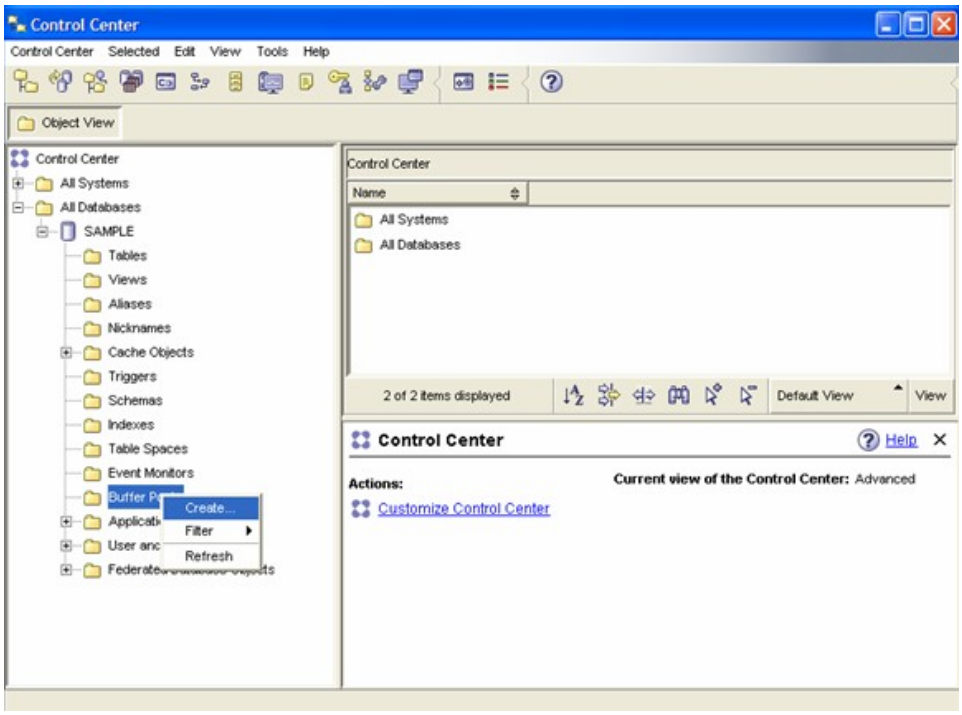


Figura 6.3 – Criar um *buffer pool*

Após clicar em `Create`, uma janela irá aparecer (Figura 6.4)

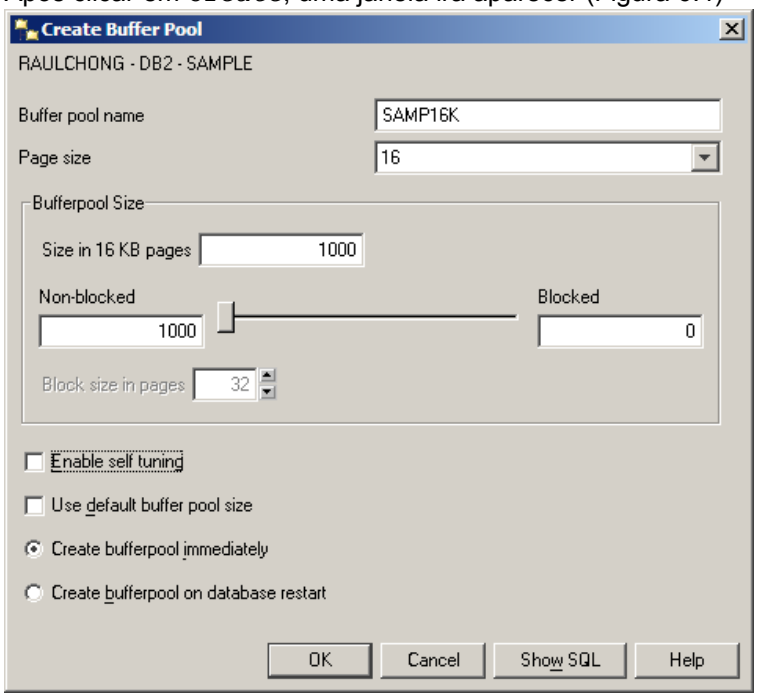


Figura 6.4 – Janela de criação do *buffer pool*

A maior parte das entradas na figura 6.4 são auto-explicativas. Os campos "*Non-blocked*" e "*Blocked*" referem-se ao número de páginas que deve existir como não-bloqueadas e como por bloqueadas. *Blocked-based buffer pools* garantem que páginas contíguas no disco são deslocadas para o *buffer pool* também de forma contígua numa área de blocos. Isso pode melhorar o desempenho. O número de páginas não deve ser superior a 98 por cento do número de páginas para o *buffer pool*. Ao especificar o valor 0 desactiva o bloco I/O.

Uma vez criado o *buffer pool*, este seria exibido no Control Center (Figura 6.5).

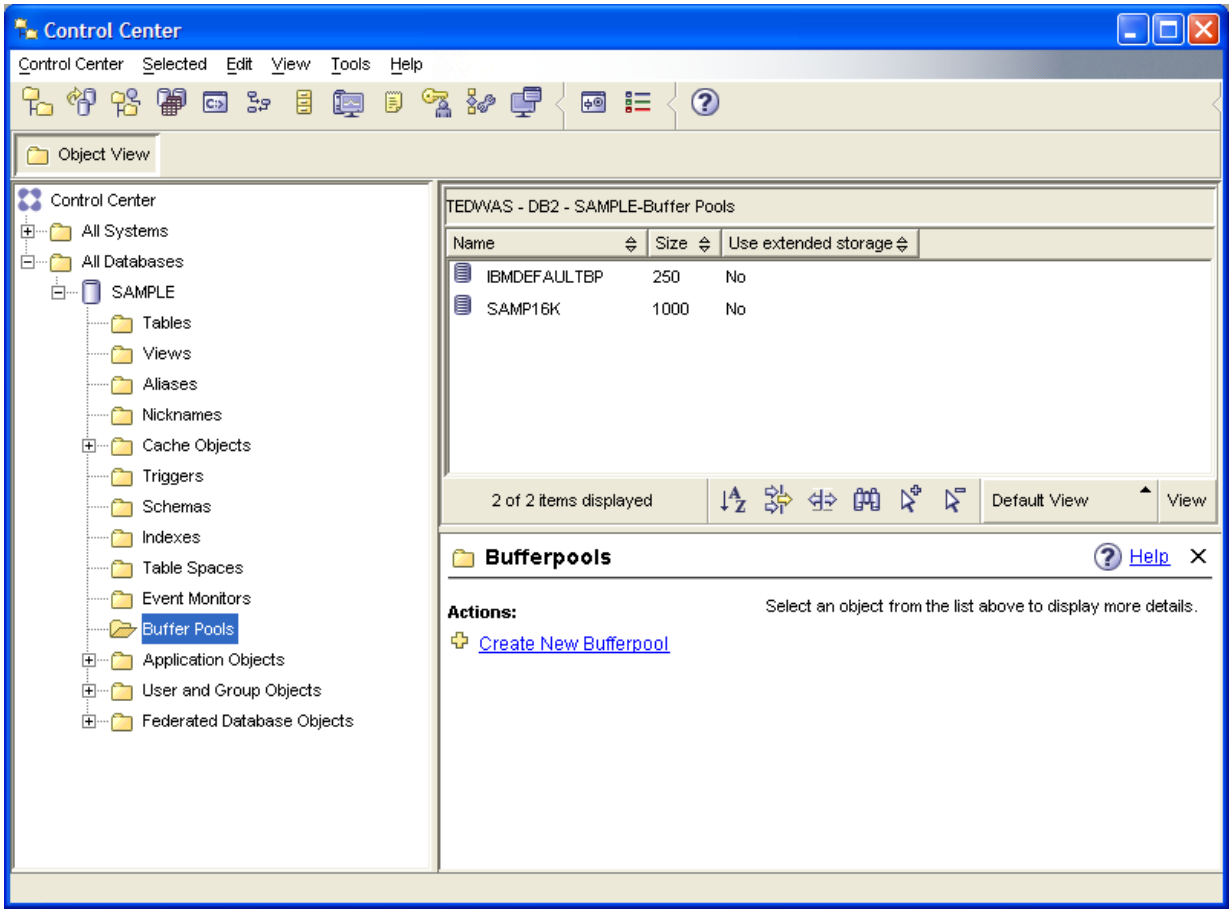


Figura 6.5 – *Control Center* depois da criação do *buffer pool* "SAMP16K"

6.3.3 *Table spaces*

Table spaces são uma interface lógica entre tabelas lógicas e a memória fisica do sistema (*buffer pool*) e os contentores (discos rígidos). Use a declaração `CREATE TABLESPACE` para criar uma *table space* onde pode especificar:

- O tamanho da página para a *table space* (4KB, 8KB, 16KB, or 32KB). A página terá de corresponder a um *buffer pool* com o mesmo tamanho de página.
- O nome do *buffer pool* associado com esta *table space*.
- O tamanho do *Extent*
- O tamanho para *Pre-fetch*

Tipos de *Table space*
Existem três tipos de *table spaces*:

- Normal
  Utilizados para as tabelas to utilizador. Por exemplo, a *table space* USERSPACE1 cria aquando da criação da base de dados, é um *table space* do tipo normal.

- Grande
  Estes *table spaces* são utilizados para separar opcionalmente dados LOB (*Large Object*). Também é usado para armazenar dados XML para bases de dados criadas com o suportr pureXML e o tipo de dados XML é utilizado nas colunas. Os *table spaces* grandes são os pré-definidas.

1. Temporário
   Existem dois tipos de *table spaces* temporários:
   1. Temporário de sistema
      São usados pelo DB2 para operações internas, como ordenações. Por exemplo, o *table space* TEMPSPACE1, criado aquando da criação de uma base de dados é um *table space* temporário de sistema.
   2. Temporário de utilizador
      Estas são usadas para criar *table spaces* definidas pelo utilizador (tabelas temporárias em memória). Estas são algumas vezes confundidas com as *table spaces* temporárias de sistema.

Gestão de *table spaces*
Os *table spaces* podem ser classificados com base no modo como são geridos. Isto pode ser especificado no comando CREATE TABLESPACE:

Gerido pelo sistema
Este tipo de *table space* é conhecido como *System Managed Storage* (SMS). Isso significa que o sistema operativo gere o armazenamento. São fáceis de gerir, e os *containers* são directorias do sistema de ficheiros. O espaço não é pré-alocado, mas os ficheiros crescem dinamicamente. Depois de especificar os *containers*, estes são fixados no tempo de criação e outros *containers* não podem ser adicionados mais tarde, a menos que um restauro do sistema seja utilizado. Ao usar o *table space* SMS, os dados da tabela, índice e LOB não podem ser repartidos por diferentes *table spaces*.

Gerido pela base de dados
Este tipo de *table space* é conhecido como *Database Managed Storage* (DMS). Isto significa que o DB2 gere o armazenamento. A gestão do espaço requer uma maior intervenção manual a partir de um DBA. Os *containers* podem ser ficheiros pré-alocados ou dispositivos. Para os dispositivos, os dados são escritos directamente sem *caching* efectuado pelo sistema operativo.

Os *containers* podem ser adicionados, removidos ou redimensionados. Os *table space* DMS são melhores para a performance, e as tabelas de dados, índices, e dados LOB podem ser divididas em *table spaces* distintos, o que melhora o desempenho.

Gerido pelo armazenamento automático
Este tipo de *table space* é gerido pelo armazenamento automático, e pode beneficiar da facilidade de uso semelhante ao *table space* SMS, mas com um melhor desempenho e flexibilidade do que o *table space* DMS. Por isso, começando com DB2 9, este é o tipo de *table space* pré-definido. Para estes table spaces, um utilizador especifica primeiro um grupo lógico de dispositivos de armazenamento. Não é necessário especificar containers, uma vez que estes são automaticamente criados. O crescimento dos actuais *containers* e adição de novos é totalmente gerido pelo DB2.

Para permitir o armazenamento automático, primeiro é preciso criar uma base de dados com armazenamento automático activo (este é o comportamento padrão) e associar um conjunto de caminhos (*paths*) de armazenamento com essa mesma base de dados. Após a criação, se fôr necessário, pode-se redefinir os caminhos de armazenamento usando a operação de base de dados RESTORE. Então, pode-se criar *table spaces* para uso automático de armazenamento (mais uma vez, este é o comportamento padrão).

Exemplo de armazenamento automático
Primeiro crie a base de dados com armazenamento automático activada como nos seguintes exemplos:

O armazenamento automático está activado por omissão:
CREATE DATABASE DB1

O armazenamento automático é explicitamente especificado:
CREATE DATABASE DB1 AUTOMATIC STORAGE YES

O armazenamento automático está activado por omissão, mas os caminhos de armazenamento são indicados:
CREATE DATABASE DB1 ON /data/path1, /data/path2

O armazenamento automático é desactivado explicitamente:
CREATE DATABASE DB1 AUTOMATIC STORAGE NO

De seguida, crie a tabela com o espaço de armazenagem automática activado como nestes exemplos:

O armazenamento automático para *table spaces* também é activado por omissão:
CREATE TEMPORARY TABLESPACE TEMPTS

O armazenamento automático é explicitamente especificado para o *table space*:
CREATE TABLESPACE TS2 MANAGED BY AUTOMATIC STORAGE

O armazenamento automático é implicitamente especificado, o tamanho inicial é alocado, assim como a quantidade a aumentar, e o tamanho máximo que pode aumentar.

```
CREATE TABLESPACE TS1
        INITIALSIZE 500 K
        INCREASESIZE 100 K
        MAXSIZE 100 M
```

Como é que a informação é armazenada nos *table spaces*
Por omissão, o DB2 irá escrever nos blocos do disco paralelamente em todos os *containers* . Por exemplo, considerando um *table space* de 4K com um tamanho 8 utilizando 3 *containers* básicos sobre uma *table space* DMS, isto significa que 32kB de dados (4K x 8 páginas por medida = 32kB) serão gravados num disco antes de escrever para o seguinte. Isso é ilustrado na Figura 6.6. Note que as tabelas não partilham os blocos (*extents*).

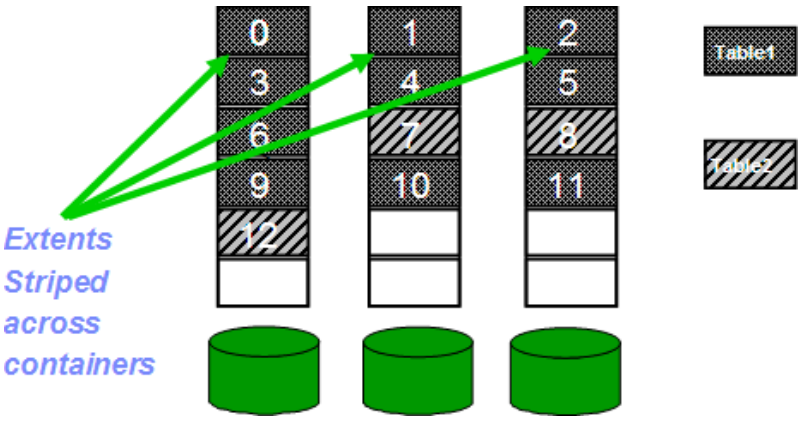


Figura 6.6 – Escrever dados em *table spaces*

Criar um *Table Space* usando o *Control Center*
Para criar um *table space* a partir do *Control Center*, clique no botão direito sobre a pasta *Table Spaces* dentro de um determinada base de dados e escolha `Create` (Figura 6.7). O "Create table space wizard" irá aparecer, como mostra na Figura 6.8.

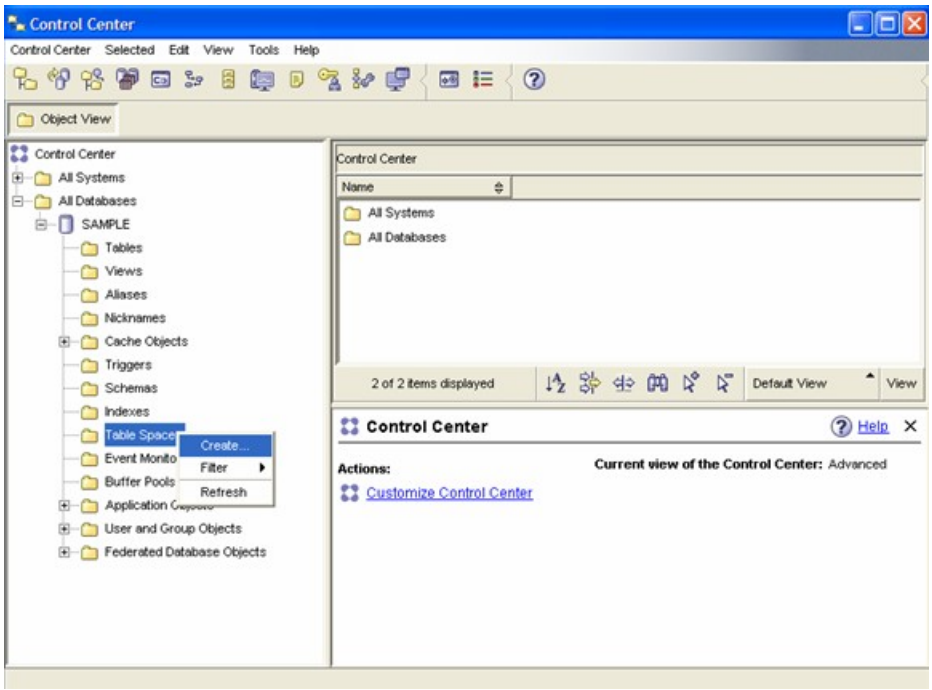


Figura 6.7 – Criar uma *Table Space* a partir do *Control Center*

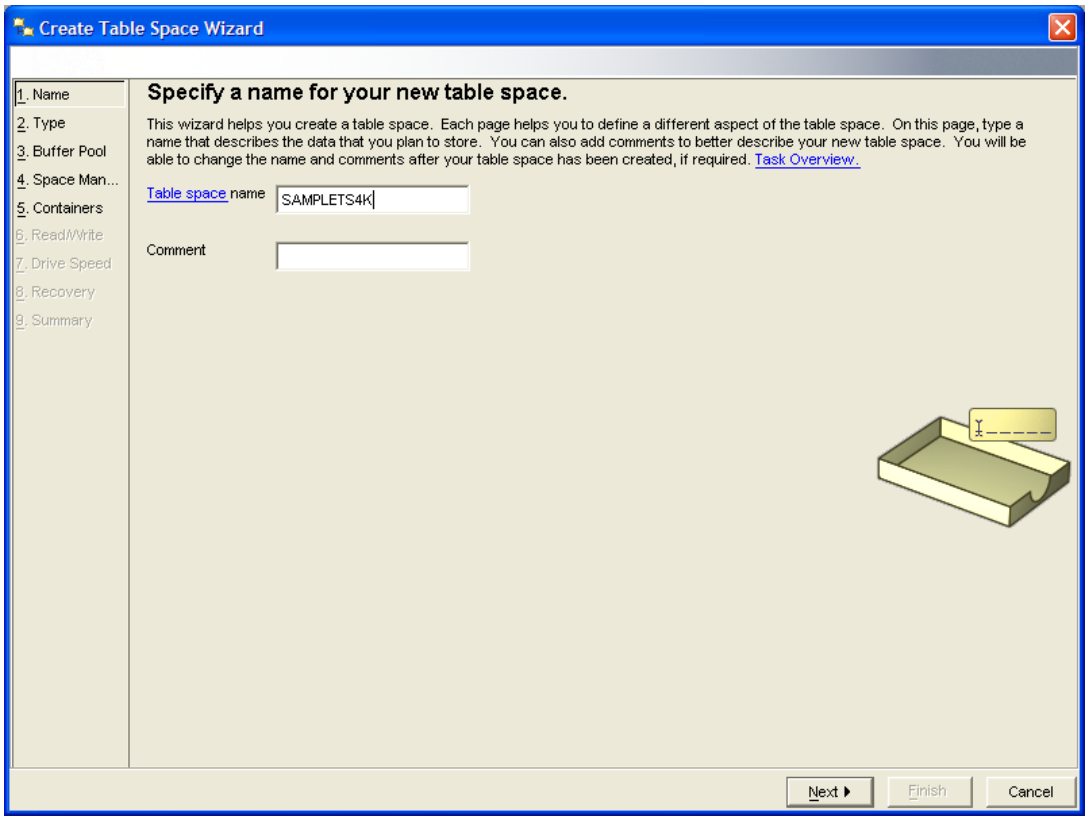


Figura 6.8 – *Create Table Space Wizard*

7

Capítulo 7 – Clientes DB2

Este capítulo cobre a instalação que é necessária para permitir a ligação entre um cliente DB2 a um servidor DB2 utilizando TCP/IP. Note-se que um servidor DB2 também contém uma componente de cliente, podendo portanto comportar-se como tal e ligar-se a outro servidor DB2. Existem várias maneiras de configurar a ligação enquanto cliente DB2; no entanto, neste capítulo iremos discutir apenas o método mais fácil, que é utilizando o Assistente de Configuração (*Configuration Assistant)*.

Nota:

Para mais informação acerca da arquitectura DB2, consulte o video:
http://www.channeldb2.com/video/video/show?id=807741:Video:4222

7.1 Directórios DB2

Os directórios DB2 são ficheiros binários que guardam a informação acerca das bases de dados a que se pode ligar a partir da sua máquina. Existem quatro directórios:

1. System database directory
Este directório pode comparar-se a um índice de um livro. Mostra todas as bases de dados, quer sejam locais ou remotas, às quais se pode ligar. Para cada base de dados local, existirá um apontador para *Local database directory*. Para as bases de dados remotas, existirá um apontador para uma entrada de *Node directory*. Para rever os conteúdos deste directório utilize o comando:

```
list db directory
```

2. Local database directory
Este directório contém informação acerca das bases de dados a que se pode ligar e que residem na sua máquina. Para rever os conteúdos deste directório utilize o comando:

```
list db directory on <drive/path>
```

3. Node directory
Este directório contém informação sobre como ligar-se a uma determinada base de dados. Por exemplo, se o protocolo TCP/IP é utilizado, uma entrada de nodo TCP/IP incluiria o endereço IP do servidor onde a base de dados DB2 a que se está a tentar ligar reside, e o porto da instância onde a base de dados reside. Para rever os conteúdos deste directório utilize o comando:

```
list node directory
```

4. DCS directory
Este directório aparecerá apenas se tiver instalado o software de ligação DB2 para se ligar a DB2 instalado em z/OS (mainframe), ou DB2 instalado em i5/OS. Para rever os conteúdos deste directório utilize o comando:

```
list dcs directory
```

Consultar e actualizar os conteúdos de todos estes directórios é possível utilizando o Assistente de Configuração.

7.2 Assistente de Configuração (*Configuration Assistant)*

Utilizando o assistente de configuração, consegue facilmente configurar a ligação entre um cliente DB2 e um servidor DB2.

Para iniciar o assistente de configuração, em Windows, escolha: Iniciar > Programas > IBM DB2 > DB2COPY1 > Set-Up Tools > Configuration Assistant.

A partir da linha de comandos, pode correr a ferramenta usando o comando `db2ca`.
A Figura 7.1 mostra o assistente de configuração.

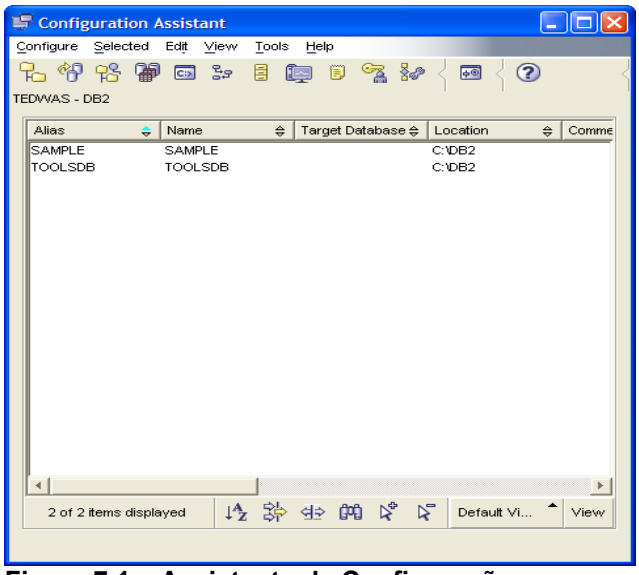


Figura 7.1 – Assistente de Configuração

7.2.1 Configuração necessária no servidor

Há duas coisas que têm de ser configuradas no servidor:

1) DB2COMM
Esta variável de registo determina quais os *listeners* do protocolo de comunicação que devem monitorizar os pedidos dos clientes. Tipicamente, TCP/IP é o protocolo de comunicação mais usado.
Alterar este parâmetro requer que se reinicie a instância. Para rever e modificar o valor do DB2COMM no assistente de configuração, seleccione *Configure -> DB2 Registry* como é mostrado na Figura 7.2 e Figura 7.3.

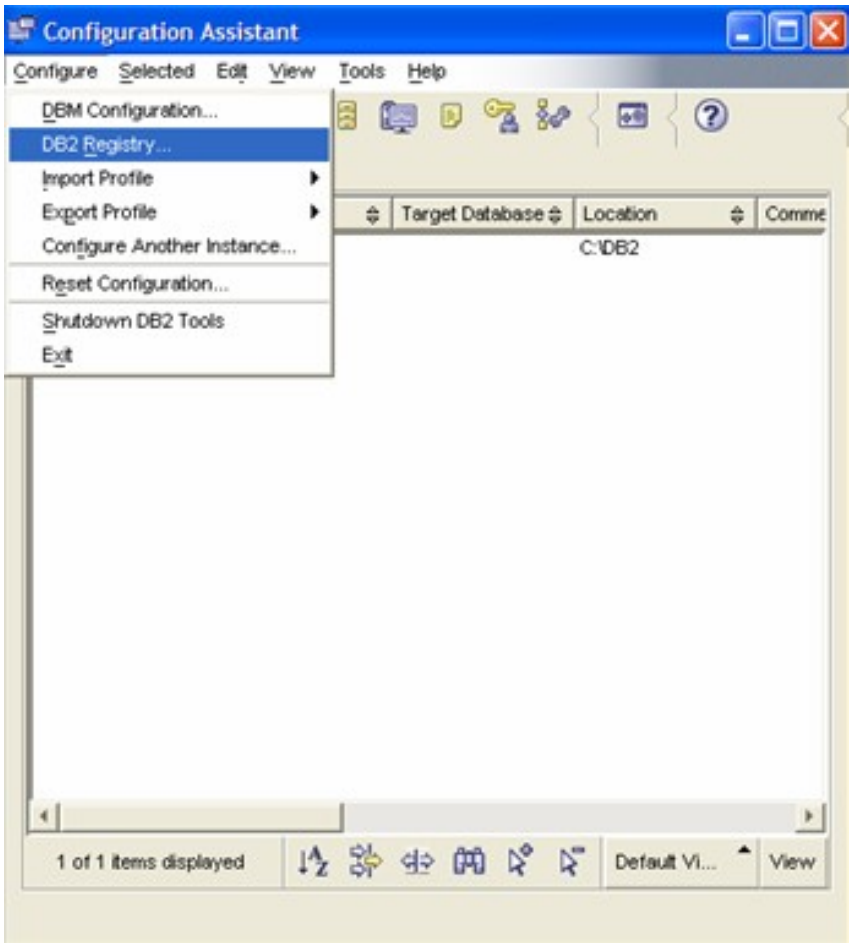


Figura 7.2 – Aceder ao Registo DB2

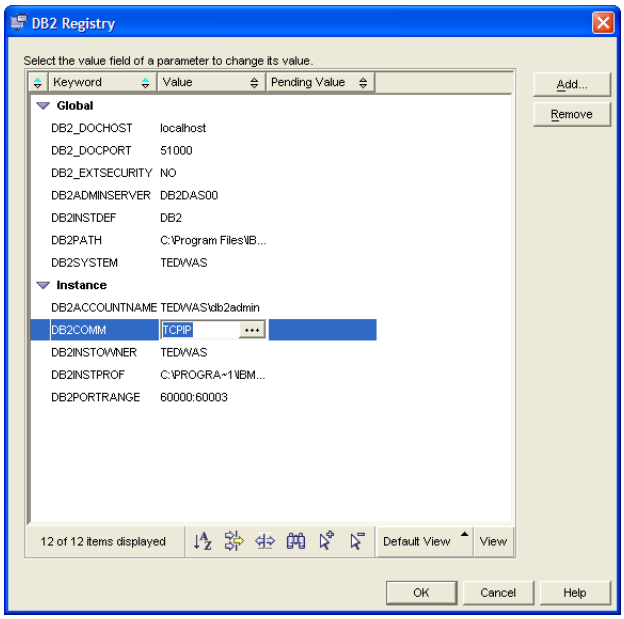


Figura 7.3 – Verificar o valor da variável de registo DB2COMM DB2

2) SVCENAME
Este parâmetro *dbm cfg* deve guardar o nome do serviço (como definido no ficheiro de serviços TCP/IP) ou o porto a usar aquando do acesso a bases de dados desta instância. A partir do Assistente de Configuração, escolha *Configure* > *DBM configuration* como mostrado na Figura 7.4

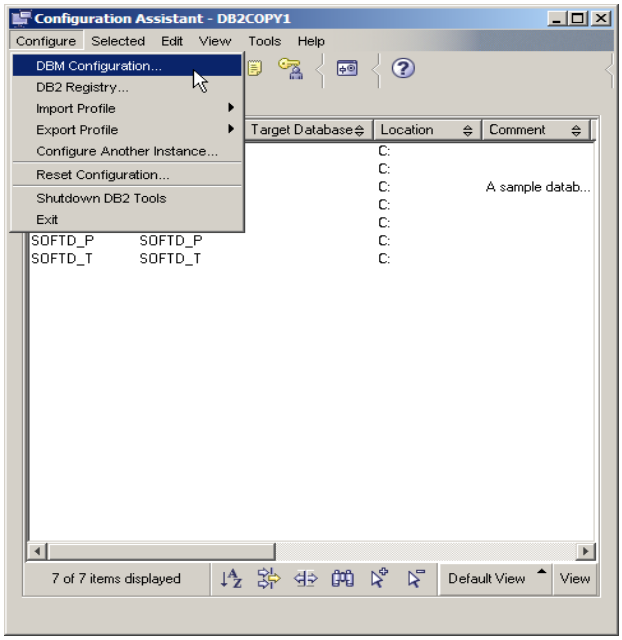


Figura 7.4 – Verificar o *dbm cfg* a partir do Assistente de Configuração

Uma vez que se encontra na janela de configuração DBM, seleccione a secção *Communications*, e procure SVCENAME. Pode alterar o valor para uma *string* ou ainda para o valor de um porto, se necessário. Veja o exemplo na Figura 7.5.

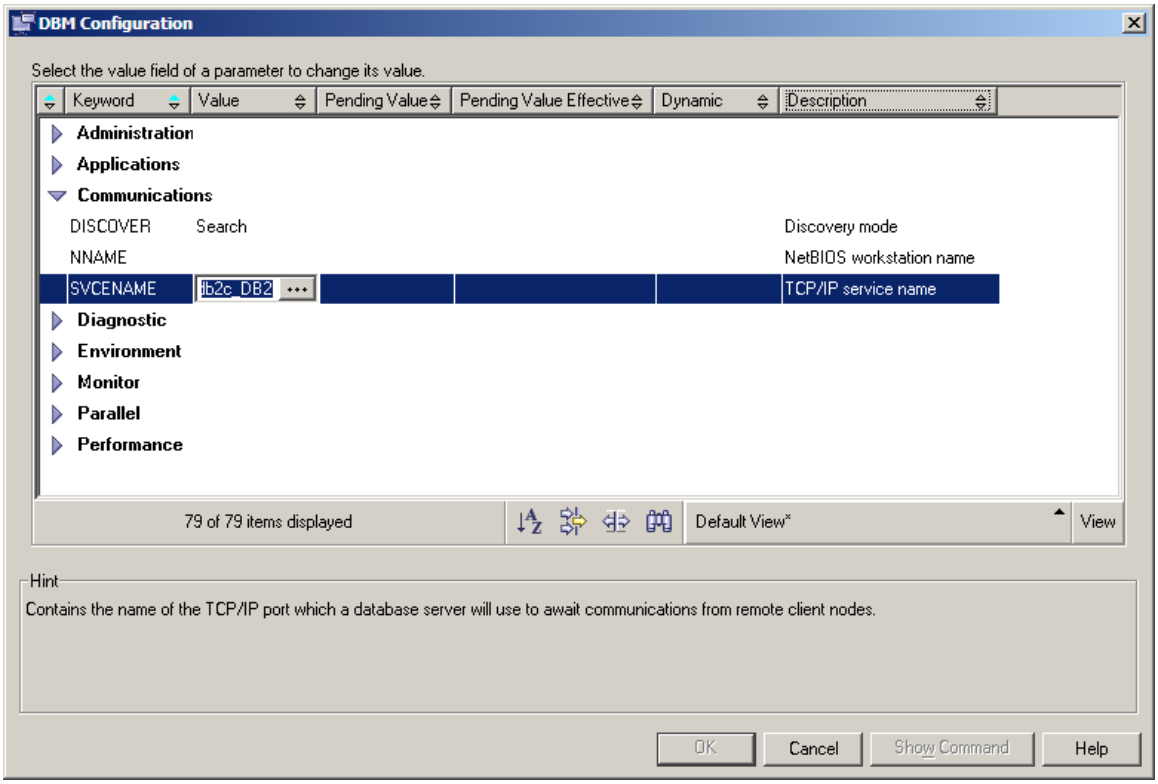


Figura 7.5 –Verificar o parâmetro SVCENAME do *dbm cfg*

7.2.2 Configurações necessárias no cliente

No cliente, precisa de recolher as seguintes informações previamente:

- O nome da base de dados à qual se quer ligar.
- O número do porto da instância DB2 do servidor onde reside a base de dados. Alternativamente, pode usar o nome de um serviço, desde que haja uma entrada correspondente no ficheiro de serviços do TCP/IP.
- O nome de utilizador e palavra chave do sistema operativo para se ligar à base de dados. É necessário que este nome de utilizador tenha sido previamente definido no servidor.

A informação acima pode ser inserida a partir do cliente DB2 utilizando o Assistente de Configuração. Primeiro, execute *Add Database Wizard*, seleccionando para isso a opção *Selected -> Add Database Using Wizard*, como é mostrado na Figura 7.6.

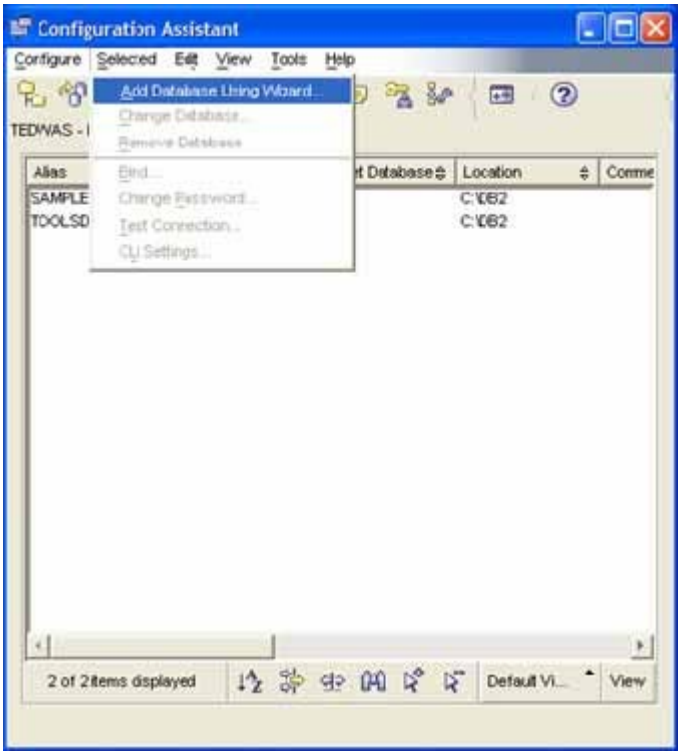


Figura 7.6 – Correr o *Add Database Wizard*

Pode também acessar este assistente clicando com o botão direito no espaço em branco do Assistente de Configuração, e escolhendo *Add Database Wizard.*

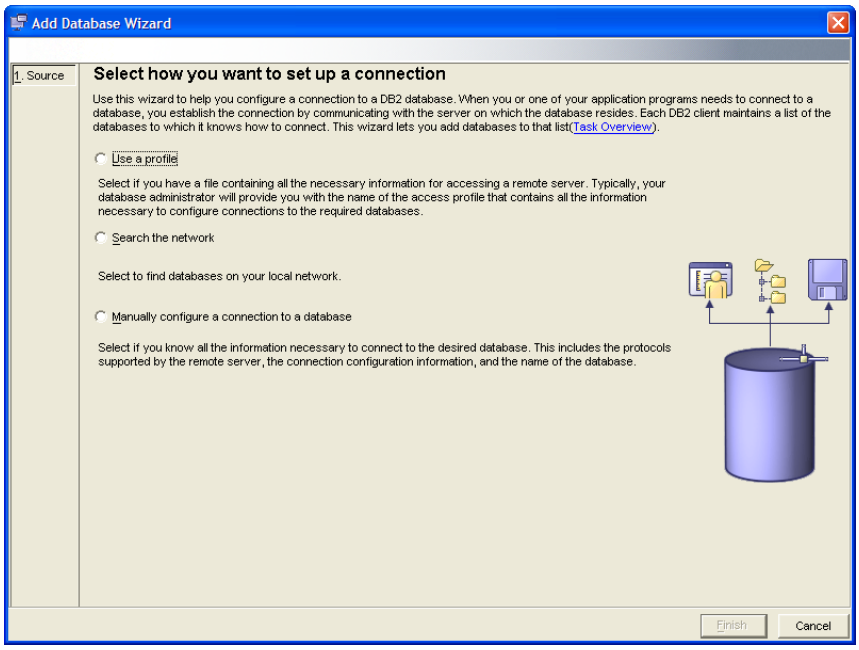


Figura 7.7 – *Add Database Wizard*

Neste assistente, há três opções:

Utilizar um *Profile*
Pode haver situações em que surja a necessidade de configurar vários clientes que se irão ligar ao mesmo servidor DB2. Nestas situações, é conveniente efectuar todas as configurações para um cliente, e guardá-las num ficheiro "profile". Com este ficheiro, pode carregar toda a informação directamente para os restantes clientes. Na Figura 7.7, se seleccionar "Use a Profile" poderá então carregar a informação de um "profile" já existente. Mais detalhes acerca deste assunto serão descritos mais para o fim deste capítulo, descrevendo como criar *profiles* de cliente e de servidor.

Procurar na rede
Este método, também conhecido como "Discovery", diz ao DB2 para procurar na rede um dado servidor, instância, e base de dados. Para este método funcionar, o DAS tem de estar a correr em cada servidor DB2 onde estiverem as bases de dados a ser encontradas. Com este método, há duas formas de pesquisar:

- *Search*:
  Procurar na rede inteira. Isto não é recomendado se a rede for grande e houverem muitos *hubs*, uma vez que pode demorar muito tempo a devolver os dados de todos os sistemas.
2. *Known*:
   Procurar na rede por um servidor conhecido, localizado num endereço a fornecer.

Os dois métodos estão ilustrados na Figura 7.8

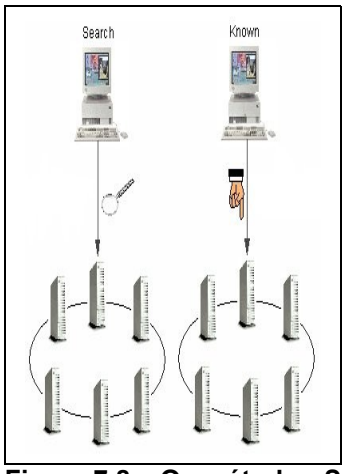


Figure 7.8 – Os métodos *Search* e *Known* (ou *Discovery*)

Pode haver situações em que o admnistrador do sistema não permita que os clientes efectuem pesquisas na rede para bases de dados com informação confidencial. Este limite pode ser feito no DAS, na instância, ou ao nível da base de dados. A Figura 7.9 mostra os detalhes acerca disto.

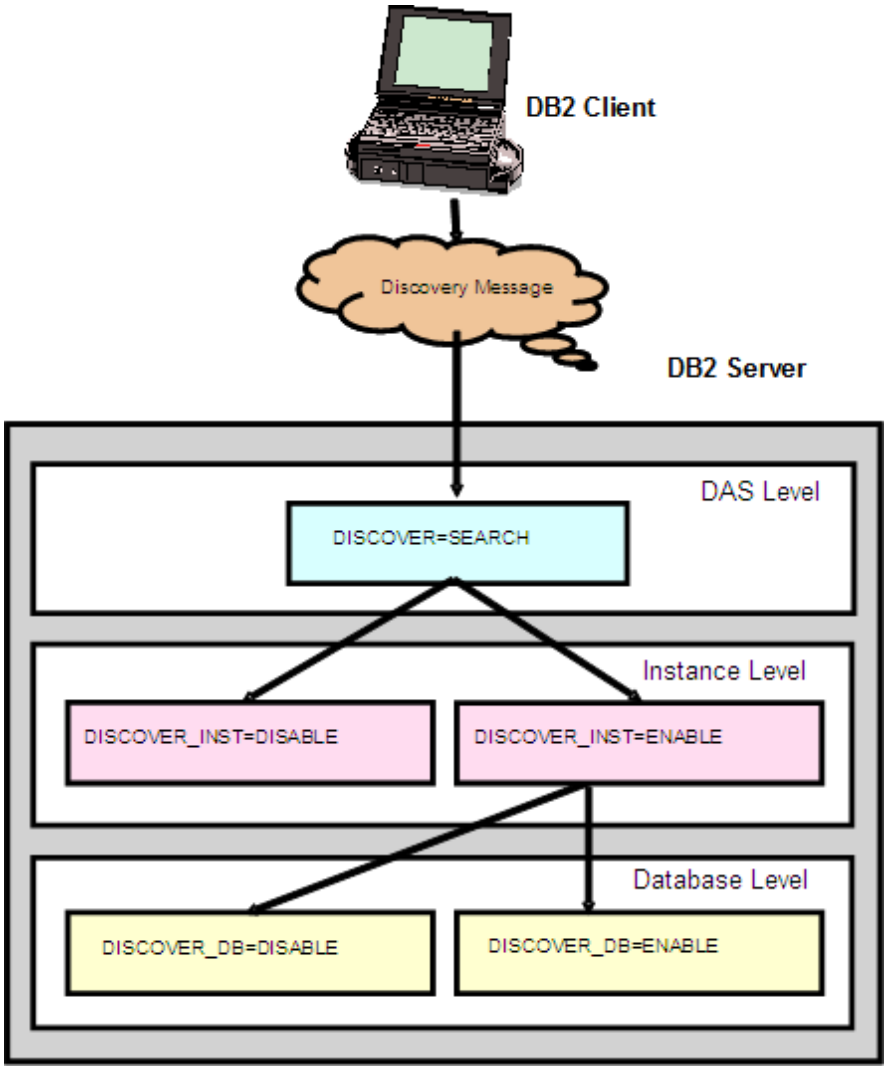


Figure 7.9 – Configurar os parâmetros de permissão para o *discovery*

A Figura 7.9 mostra os diferentes níveis onde pode activar ou desactivar o *discovery*. Ao nível do DAS, pode dar ao parâmetro DISCOVERY o valor SEARCH ou KNOWN. Ao nível da instância, o parâmetro do dbm cfg DISCOVERY_INST pode ser configurado para DISABLE ou ENABLE. Finalmente, ao nível da base de dados, o parâmetro DISCOVERY_DB pode também ser alterado para ENABLE ou DISABLE. Configurar estes parâmetros em conformidade garante-lhe granularidade para a procura de bases de dados.

Configuração manual de uma ligação a uma base de dados
Usando este método, pode adicionar manualmente o *host name*, números dos portos e informação relativa à base de dados no Assistente de Configuração, que de seguida irá gerar comandos “catalog” para executar a configuração da ligação. O Assistente de Configuração não irá verificar se a informação está correcta. Saberá se a informação está incor-

recta se não se conseguir ligar ao servidor. Certifique-se também que o nome de utilizador e a palavra chave que forneceu para aceder à base de dados estão correctos. Por omissão, a autenticação ocorre no servidor DB2 a que se está a tentar ligar, portanto, deve fornecer um nome de utilizador e palavra chave definidos nesse servidor.

7.2.3 Criar *Profiles* de Cliente e Servidor

Se estiver a configurar um número grande de servidores ou clientes, em vez de o fazer sequencialmente, pode configurar apenas um, e de seguida exportar o *profile* (i.e. ficheiro de configuração) de forma a aplicar as mesmas configurações aos outros clientes/servidores. Isto permite economizar muito tempo de administração na configuração do sistema.

Para criar um *profile* personalizado a partir do Assistente de Configuração, seleccione *Configure Menu*, e de seguida *Export Profile => Customize*, como é mostrado na Figura 7.10

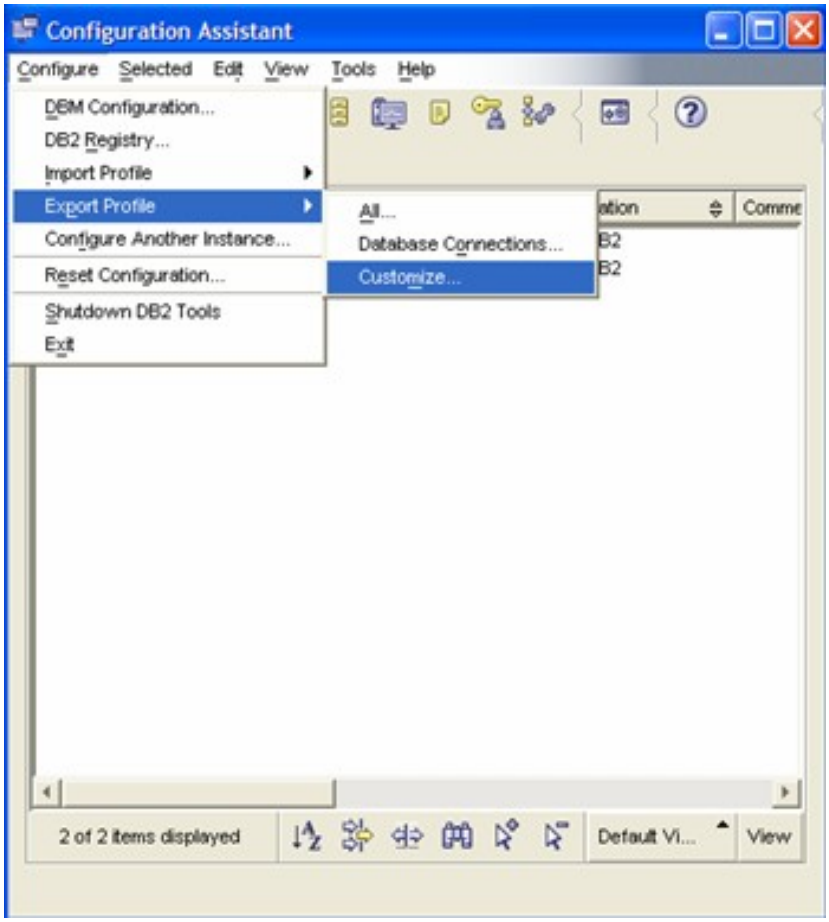


Figura 7.10 – Exportar um *Profile*

A Figura 7.11 mostra os campos que têm de ser preenchidos para exportar um ficheiro de *profile*.

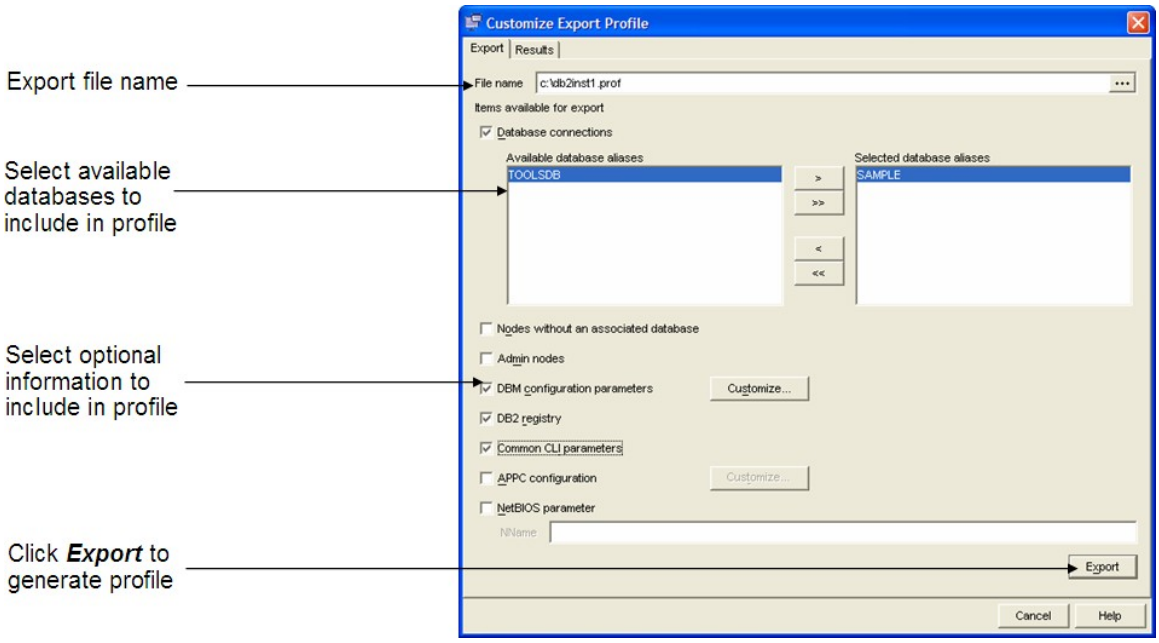


Figura 7.11 – Janela de *Customize Export Profile*

A Figura 7.12 mostra os resultados depois da selecção de "Export" na janela *Customize Export Profile*.

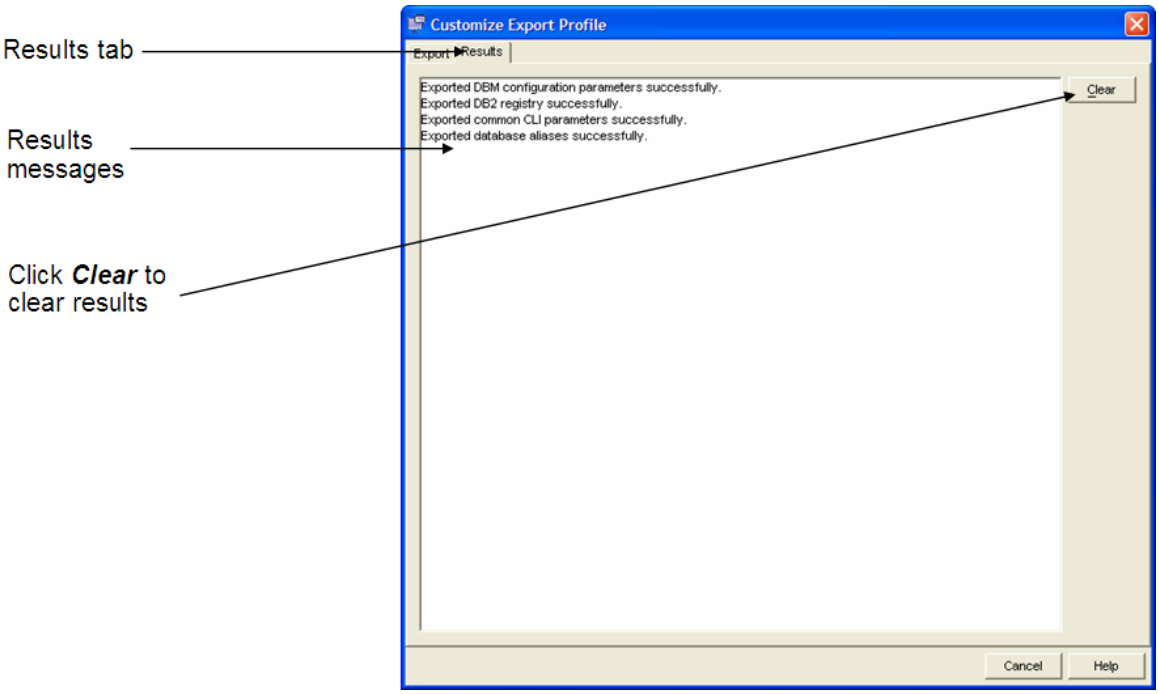


Figura 7.12 – Resultados da Exportação de *profile*

Para importar um *profile* personalizado a partir do Assistente de Configuração, seleccione *Configure Menu*, e de seguida *Import Profile* => *Customize* (Figura 7.13).

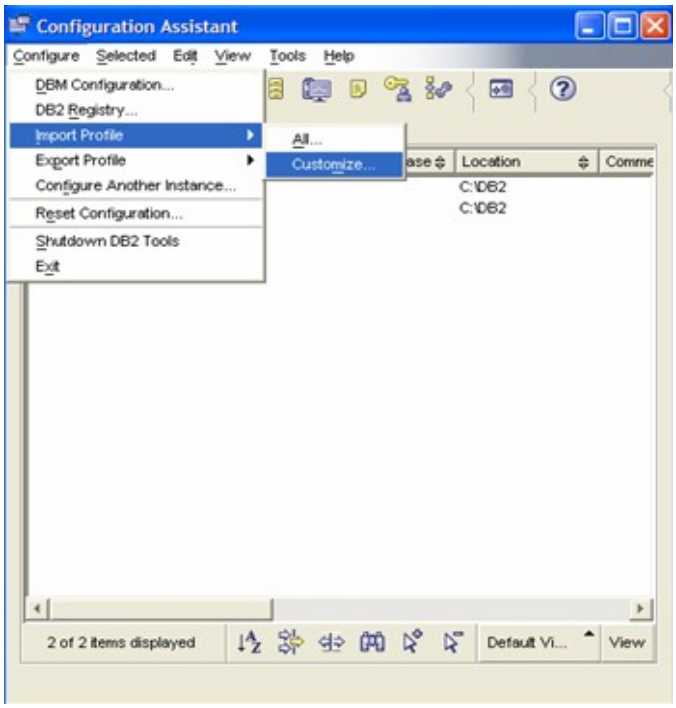


Figura 7.13 – Importar um *profile*

A Figura 7.14 mostra os campos que têm de ser preenchidos para importar um *profile*.

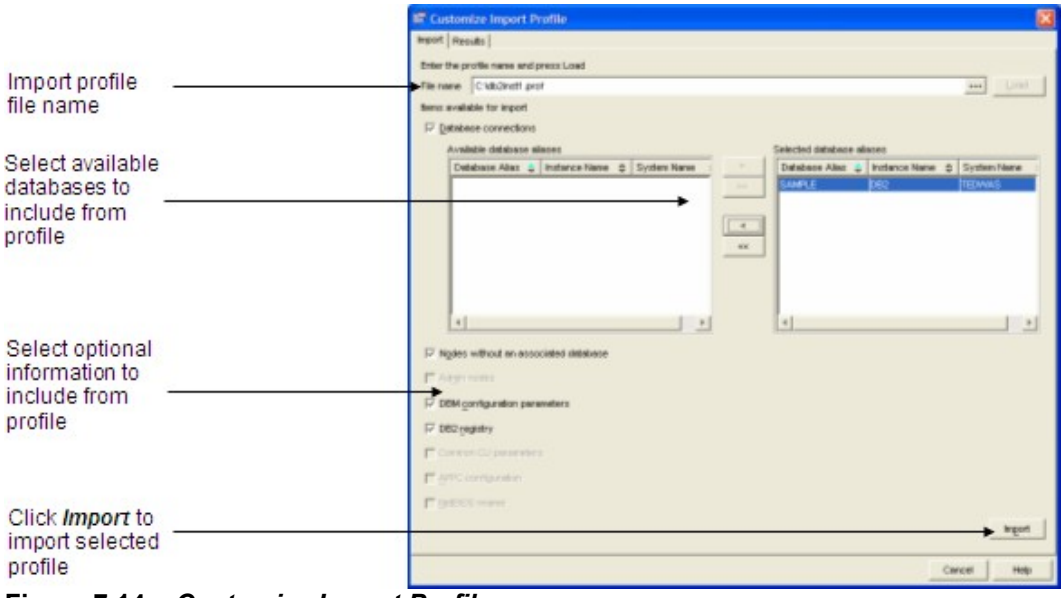


Figura 7.14 – *Customize Import Profile*

Quicklab #6: Utilizar o Assistente de Configuração

Objectivo

O Assistente de Configuração pode ser utilizado para configurar de forma fácil e rápida ligações a bases de dados remotas. Neste Quicklab, irá catalogar uma base de dados presente num servidor DB2 remoto (representada pela estação do seu vizinho, usando os modos *Search* e *Discovery*. Uma vez catalogada a base dados, poderá acedê-la como se esta estivesse presente no seu sistema local. O DB2 efectua todos os processos de comunicação de forma invisível).

Nota:
Este Quicklab assume que está a trabalhar dentro de uma rede. Se não for o caso, pode sempre utilizar o seu computador pessoal simultaneamente como cliente e servidor, e seguir as instruções abaixo para se ligar ao seu próprio sistema.

Procedimento

- Solicite ao seu vizinho (ou instrutor) a seguinte informação:

  Info da Base de Dados remota:

(PR)	Protocolo	__TCP/IP__ __
(IP)	Endereço IP ou hostname	_____
(PN)	Número do Porto da Instância	_____
(DB)	Nome da Base de Dados	_ SAMPLE __

  Dicas:
  1. Para obter o *hostname* em Windows, digite *hostname* numa linha de comandos
  2. Para obter o endereço IP em Windows, digite *ipconfig* numa linha de comandos

- Abra o Assistente de Configuração. (Dica: está disponível a partir do menu Iniciar).

- Abra o menu *Selected* e seleccione *Add Database Using Wizard*.

- Na página *Source* do assistente, seleccione a opção *Manually Configure a Connection to a Database*. Seleccione o botão *Next* para avançar para a próxima etapa do assistente.

- Na página *Protocol* do assistente, seleccione a opção TCP/IP. Seleccione o botão *Next* para avançar para a próxima etapa do assistente.

- Na página *TCP/IP* do assistente, insira a hostname ou o endereço IP de que tomou nota no passo (1). Insira o número da porta de que tomou nota no passo (1). Seleccione o botão *Next* para avançar para a próxima etapa do assistente.
  Nota: A opção para *Service Name* pode ser usada se dispuser de uma entrada no ficheiro local de Serviços com o número de uma porta correspondente com o porto no qual a instância do servidor remoto se encontra à escuta. Quando usar esta opção, o DB2 irá procurar no ficheiro de services na máquina local, e não no servidor. Terá de adicionar uma entrada neste ficheiro se pretender usar esta opção.

- Na página *Database* do assistente, insira o nome da base de dados definida no servidor remoto de que tomou nota no passo (1) no campo *Database Name*. Repare como o campo *Database Alias* é automaticamente preenchido com o mesmo valor. O *alias* da base de dados é um nome que as aplicações locais irão usar para se ligarem a ela. Uma vez que já dispõe de uma base de dados local chamada *SAMPLE*, o DB2 não irá permitir que catalogue outra base de dados com o mesmo nome. Portanto, terá de utilizar um nome de *alias* diferente. Para este exemplo, altere o *alias* da base de dados para *EXEMPLO1*. Pode ainda, se desejar, adicionar um comentário acerca desta base de dados. Seleccione o botão *Next* para avançar para a próxima etapa do assistente.

- Na página *Data Source* do assistente, pode opcionalmente registar esta nova base de dados como sendo uma fonte de dados ODBC. Isto irá automaticamente registá-la no *ODBC Manager* do Windows. Para este exemplo, retire o visto em *Register this database for ODBC*, uma vez que não irá usar ODBC. Seleccione o botão *Next* para avançar para a próxima etapa do assistente.

- Na página *Node Options* do assistente, especifique o sistema operativo do servidor onde se encontra a base de dados remota. Uma vez que todas as estações neste lab usam o Microsoft Windows, certifique-se que o item *Windows* na lista *drop-down* se encontra seleccionado. O campo com o nome da instância deverá estar preenchido com *DB2*. Se não estiver, altere-o para *DB2*. Seleccione o botão *Next* para avançar para a próxima etapa do assistente.

- Esta página *System Options* do assistente dá-lhe a oportunidade de assegurar que o sistema e o hostname estão correctos, e de verificar a configuração referente ao sistema operativo. Seleccione o botão *Next* para avançar para a próxima etapa do assistente.

- A página *Security Options* do assistente permite-lhe especificar o sítio onde quer que a autenticação tome lugar, e que método quer utilizar para

tal. Seleccione a opção *Use authentication value in server's DBM Configuration*.

Isto irá usar o método especificado pelo parâmetro `AUTHENTICATION` do ficheiro de configuração da instância remota. Seleccione o botão *Finish* para catalogar a base de dados remota e feche o assistente. Uma janela de confirmação deverá aparecer. Seleccione o botão *Test Connection* para assegurar que se consegue ligar com sucesso à base de dados. Certifique-se também que o nome de utilizador e palavra chave fornecidos são válidos e *definidos no servidor remoto* (uma vez que é provável que o parâmetro `AUTHENTICATION` do servidor esteja configurado para `SERVER`). Se o teste de ligação for bem sucedido, então catalogou correctamente e com sucesso a base de dados remota. Se o teste não for bem sucedido, volte atrás no assistente e confirme que todos os valores estão correctos. (Seleccione o botão *Change* para voltar atrás no assistente de configuração).

- Abra o *Control Center* e tente ver as diferentes tabelas na nova base de dados catalogada.

- Volte novamente ao Assistente de Configuração e tente catalogar uma nova base de dados, utilizando desta vez a opção *Search the Network*. Avance no assistente da mesma forma que fez na configuração manual da ligação. Tenha em conta que em redes grandes a procura *discovery* pode levar muito tempo até trazer alguns resultados.

8

Capítulo 8 – Trabalhar com Objectos da Base de Dados

Este capítulo aborda os objectos da base de dados como *schemas*, tabelas, *views*, *indexes*, *sequences*, entre outros. Os objectos mais avançados da base de dados como *triggers*, *user defined functions* (UDFs) e *stored procedures* são discutidos no Capítulo 14 – SQL PL Stored Procedures, e no Capítulo 15 – Inline SQL PL, Triggers, e UDFs.

Nota:

Para mais informação sobre como trabalhar com objectos da base de dados veja este vídeo: http://www.channeldb2.com/video/video/show?id=807741:Video:4242

8.1 *Schema*

Schemas são *name spaces* para uma colecção de objectos da base de dados. São usados principalmente para:

2. Fornecer uma indicação do dono do objecto ou relação para uma aplicação
3. Agrupar logicamente objectos relacionados

Todos os objectos da base de dados DB2 possuem um nome completo qualificado baseado em duas partes; o *schema* é a primeira parte dessa nome:

<schema_name>.<object_name>

O nome de um objecto completo qualificado tem de ser único. Quando nos ligamos a uma base de dados e criamos ou referenciamos um objecto sem especificar o *schema*, o DB2 usa o nome de utilizador utilizado na ligação à base de dados para a parte *schema* do nome. Por exemplo, se nos ligarmos à base de dados SAMPLE com o utilizador "arfchong", e criarmos uma tabela com o comando CREATE TABLE:

```
CREATE TABLE artists …
```

O nome completo qualificado da tabela criada é na realidade `arfchong.artists`.

8.2 Tabelas

Uma tabela é uma colecção de dados relacionados organizados logicamente em colunas e linhas. O comando abaixo é um exemplo da criação de uma tabela através do comando `CREATE TABLE`.

```
CREATE TABLE artists
 (artno           SMALLINT    not null,
  name            VARCHAR(50) with default 'abc',
  classification  CHAR(1)     not null,
```

```
 bio             CLOB(100K)  logged,
 picture         BLOB(2M)    not logged compact
)
IN mytbls1
```

Nas secções seguintes, iremos descrever as partes principais deste comando `CREATE TABLE`.

8.2.1 Tipos de Dados

A Figura 8.1 lista os tipos de dados suportados pelo DB2

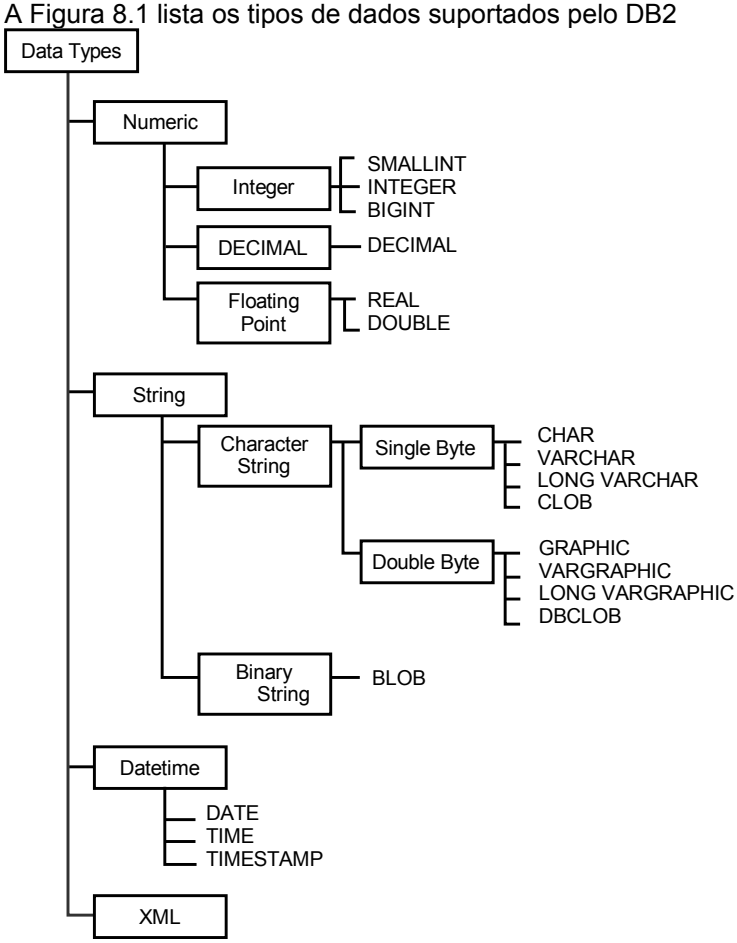


Figura 8.1 – Tipos de dados *built-in* do DB2

Tipo de dados *Large Object* (LOB)
Tipos de dados de *large objects* são usados para guardar sequências de caracteres de grande dimensão, sequências binárias de grandes dimensão ou ficheiros, como ilustra a Figura 8.2

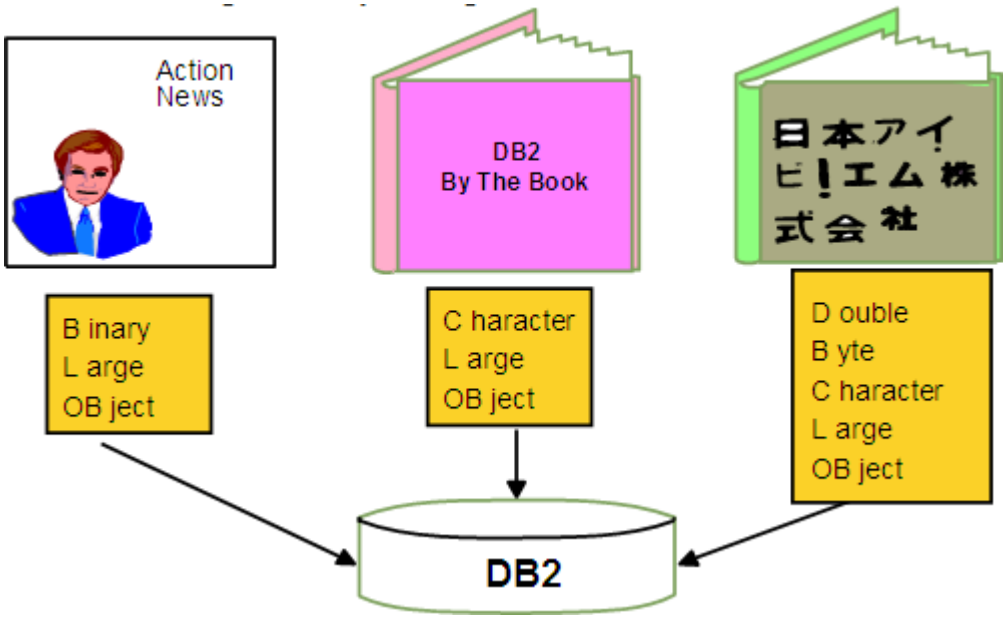


Figura 8.2 – Tipos de dados de LOBs

Os *large object* binários são normalmente abreviados para se tornarem mais claros: um *binary large object* é BLOB, um *character large object* é um CLOB, e um *double byte character large object* é também conhecido como DBCLOB.

Tipos definidos pelo utilizador (*User-defined types*)
O DB2 permite definir os nossos próprios tipos de dados, baseados nos tipos *built-in*. Estes são conhecidos como os tipos *user-defined* (UDTs). Os UDTs são úteis quando:

- há necessidade de contextualizar os valores
- há necessidade do DB2 reforçar a tipagem de dados

Os comandos seguintes ilustram um exemplo de como e quando devemos usar UDTs:

```
CREATE DISTINCT TYPE POUND AS INTEGER WITH COMPARISONS

CREATE DISTINCT TYPE KILOGRAM AS INTEGER WITH COMPARISONS

CREATE TABLE person
   (f_name    VARCHAR(30),
    weight_p  POUND NOT NULL,
    weight_k  KILOGRAM NOT NULL )
```

Neste exemplo, dois UDTs foram criados: POUND e KILOGRAM. Ambos são baseados no tipos de dados *built-in* INTEGER. As cláusulas WITH COMPARISONS definidas como uma parte da sintaxe indicam que também serão criadas as funções de conversão com o mesmo nome do que os tipos de dados.

A tabela `person` utiliza as duas novas UDTs nas colunas `weight_p` e `weight_k`, respectivamente. Se executarmos o comando seguinte:

```
SELECT F_NAME FROM PERSON
  WHERE weight_p > weight_k
```

iremos receber uma mensagem de erro porque estão a ser comparadas duas colunas com tipos de dados diferentes. Apesar de weight_p e weight_k usarem os tipos de dados POUND e KILOGRAM respectivamente, ambos criados baseados no tipo de dados INTEGER, através dos UDTs, tornamos este tipo de comparação impossível. Isto é exactamente o que queremos, porque na vida real, o que significa uma comparação entre libras e kilogramas? Não faz sentido.

No próximo exemplo desejamos comparar a coluna weight_p com um *integer*; no entanto, estes dois tipos de dados são diferentes, e por isso receberemos uma mensagem de erro excepto quando usarmos a função *cast* para converter os dados.

```
SELECT F_NAME FROM PERSON
  WHERE weight_p > 30
```

Como podemos ver no comando em baixo, usamos a função de conversão POUND() para que a comparação seja possível. Como indicado anteriormente, a função de conversão POUND() foi criada com a UDT quando invocamos a clausula WITH COMPARISONS do comando CREATE DISTINCT TYPE.

```
SELECT F_NAME FROM PERSON
  WHERE weight_p > POUND(30)
```

Valores nulos
Um valor nulo representa um estado desconhecido. No entanto, o comando CREATE TABLE pode definir uma coluna usando a clausula NOT NULL. Esta assegura que essa coluna contém valores de dados conhecidos. Pode também especificar um valor default para a coluna se for declarado NOT NULL. O próximo comando fornece exemplos deste comportamento:

```
CREATE TABLE Staff (
     ID       SMALLINT NOT NULL,
     NAME     VARCHAR(9),
     DEPT     SMALLINT NOT NULL with default 10,
     JOB      CHAR(5),
     YEARS    SMALLINT,
     SALARY   DECIMAL(7,2),
     COMM     DECIMAL(7,2) with default 15
     )
```

8.2.2 *Identity Columns*

Uma *identity column* é uma coluna numérica que gera automaticamente um valor numérico único para cada linha inserida. Só é possível uma *identity column* por tabela.

Existem duas formas de gerar valores para uma *identity column*, dependendo da sua definição:

- **Gerados sempre:** os valores são sempre gerados pelo DB2. Não são permitidas às aplicações fornecer valores explícitos.
- **Gerados por omissão:** os valores podem ser fornecidos explicitamente por uma aplicação ou, se nenhum valor for dado, o DB2 gera um. O DB2 não garante unicidade. Esta opção é intencionada para propagação de dados, e para *unloading* e *reloading* de uma tabela.

Vejamos o próximo exemplo:

```
CREATE TABLE subscriber(subscriberID INTEGER GENERATED ALWAYS AS
                                     IDENTITY (START WITH 100
                                     INCREMENT BY 100),
                        firstname VARCHAR(50),
                        lastname  VARCHAR(50) )
```

Neste exemplo, a coluna subscriberID é um INTEGER definido como uma *identity column* que é sempre gerado. O valor gerado é inicialmente 100, e irá ser incrementado em 100.

8.2.3 Objectos SEQUENCE

Objectos s*equence* geram um número único dentro da base de dados. Ao contrário das *identity columns*, *sequences* são independentes das tabelas. Os comandos seguintes fornecem um exemplo:

```
CREATE TABLE t1 (salary int)

CREATE SEQUENCE myseq
   START WITH 10
   INCREMENT BY 1
   NO CYCLE

INSERT INTO t1 VALUES (nextval for myseq)
INSERT INTO t1 VALUES (nextval for myseq)
INSERT INTO t1 VALUES (nextval for myseq)

SELECT * FROM t1

SALARY
-----------
         10
         11
         12
  3 record(s) selected.
```

```
SELECT prevval for myseq FROM sysibm.sysdummy1

1
-----------
         12
  1 record(s) selected
```

PREVVAL dá o valor actual do *sequence*, enquanto NEXTVAL fornece o próximo valor.

O exemplo anterior também usa a tabela SYSIBM.SYSDUMMY1. Esta é uma tabela do catálogo do sistema que contém uma coluna e uma linha. Pode ser usado quando uma *query* retorna *output* baseado apenas num valor. Tabelas de catálogo do sistema são descritas na próxima secção.

8.2.4 Tabelas de catálogo do sistema

Cada base de dados possui as suas próprias tabelas de catálogos de sistema e *views*. Estas guardam *metadata* sobre os objectos da base de dados. Pode fazer consultas sobre estas tabelas como uma tabela normal da base de dados.
Três *schemas* são usados para identificar as tabelas de catálogo do sistema:

- SYSIBM: tabelas base, optimizadas para DB2
- SYSCAT: *views* baseadas nas tabelas SYSIBM, optimizadas para facilidade de uso
- SYSSTAT: estatística da base de dados

Seguem-se alguns exemplos de *views* de catálogos:

- SYSCAT.TABLES
- SYSCAT.INDEXES
- SYSCAT.COLUMNS
- SYSCAT.FUNCTIONS
- SYSCAT.PROCEDURES

8.2.5 Tabelas temporárias (*Declared temporary tables*)

Tabelas temporárias são tabelas criadas em memória. São utilizadas por uma aplicação e depois eliminadas automaticamente quando a aplicação termina. Estas tabelas são acedidas apenas pela aplicação que as criou. Não existe nenhuma entrada no catálogo de tabelas do DB2. O acesso a estas tabelas resulta numa grande eficiência de desempenho porque não passa por etapas como *catalog contention*, *locking of rows*, *default logging* (o *logging* é opcional), e não há verificação de autoridade. Existe também um *index* que suporta estas tabelas temporárias, isto é, qualquer *index* normal pode ser criado numa tabela temporária. Pode também executar o RUNSTATS nestas tabelas.

Tabelas temporárias residem dentro de um *table space* temporário do utilizador, que precisa de ser definido préviamente à criação de qualquer tabela temporária. Os comandos seguintes fornecem um exemplo de como criar três tabelas temporárias:

```
CREATE USER TEMPORARY TABLESPACE apptemps
     MANAGED BY SYSTEM USING ('apptemps');
```

```
DECLARE GLOBAL TEMPORARY TABLE temployees
     LIKE employee NOT LOGGED;

DECLARE GLOBAL TEMPORARY TABLE tempdept
      (deptid CHAR(6), deptname CHAR(20))
       ON COMMIT DELETE ROWS NOT LOGGED;

DECLARE GLOBAL TEMPORARY TABLE tempprojects
     AS ( fullselect ) DEFINITION ONLY
     ON COMMIT PRESERVE ROWS NOT LOGGED
     WITH REPLACE IN TABLESPACE apptemps;
```

Quando uma tabela temporária é criada, a sua *schema* é SESSION, e precisa de ser especificada. O nome do utilizador é utilizado para criar uma tabela temporária e terá todos os privilégios sobre a tabela. Cada aplicação que cria uma tabela temporária terá a sua própria cópia independente como ilustra a Figura 8.5.

Figura 8.5 – Espaço das tabelas temporárias globais

Quicklab #7: Criar uma nova tabela

Objectivo

Até agora, estivemos a usar as tabelas existentes na base de dados *SAMPLE* para ilustrar os conceitos. Posteriormente, precisaremos de criar as nossas próprias tabelas na base de dados. Neste Quicklab, iremos usar o *Create Table Wizard* para criar duas novas tabelas na base de dados *SAMPLE.*

Procedimento

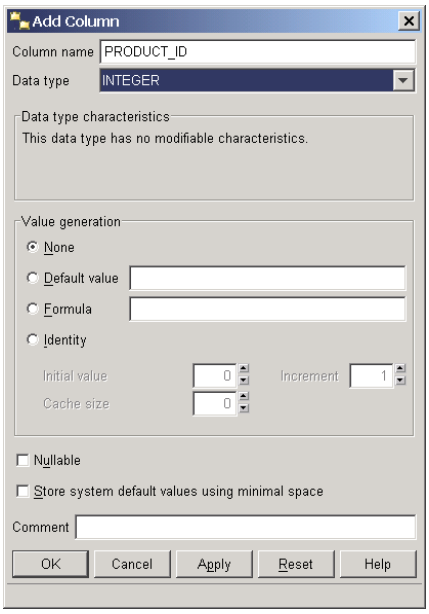
3. Execute o *Create Table Wizard* como previamente foi mostrado na apresentação. (*opção Control Center -> All Databases -> SAMPLE -> (right-click) Tables object -> Create …)*

4. Defina o nome da tabela, definições das colunas, e quaisquer restrições (*constraints)*. A tabela irá ser utilizada para guardar informação sobre as reservas de um escritório usada por um projecto na base de dados *SAMPLE.* Cada vez que um item for comprado, uma linha irá ser inserida na tabela. A tabela terá seis colunas:

- product_id: identificador único do item adquirido
- description: descrição do item
- quantity: quantidade comprada
- cost: custo do item
- image: uma imagem do item (se disponível)
- project_num: o projecto para o qual o item foi comprado

- Na primeira página do *wizard*, para o nome da *schema*, introduza o nome de utilizador que está a usar, e utilize o nome de tabela: *SUPPLIES*. Pode opcionalmente adicionar um comentário. Clique no botão *NEXT* para continuar para a próxima página do *wizard*.

- Nesta página, pode adicionar colunas a uma tabela. Clique no botão *ADD* para adicionar colunas.



Introduza *"product_id"* como nome da coluna e seleccione o tipo de dados: *INTEGER*. Elimine a opção *Nullable*, e clique no botão *Apply* para definir a coluna.

Repita este passo para as restantes colunas da tabela utizando as opções da coluna abaixo. Quando todas as colunas estiverem adicionadas, clique no botão *OK* e a lista das colunas recém criadas deverão ser sumarizadas. Clique no botão *Next* para continuar para a próxima página do *wizard*.

Nome da coluna	**Atributos**
product_id	INTEGER, NOT NULL
description	VARCHAR, length 40, NOT NULL
quantity	INTEGER, NOT NULL
cost	DECIMAL, Precision 7, Scale 2, NOT NULL
image	BLOB, 1MB, NULLABLE, NOT LOGGED
project_num	CHAR, length 6, NOT NULL

Nota: A opção *NOT LOGGED* pode ser especificada quando declaramos colunas LOB. É imperativo para colunas de tamanho superior a 1GB. É também recomendado para LOBs maiores do que 10MB pois as alterações a colunas grandes podem "encher" rapidamente o ficheiro log. Mesmo que *NOT LOGGED* seja usado, alterações a ficheiros LOB durante a transacção podem ser bem sucedidas. Repare também que a coluna imagem é a única definida como coluna "*NULLABLE*". Porque será?

- Neste ponto, toda a informação importante para a criação de uma tabela foi fornecida. Saltando as outras páginas, estará a escolher valores default para

essas opções. Pode sempre adicionar chaves e restrições depois de uma tabela ser criada.

- Adicione uma restrição à tabela para restringir valores na coluna *quantity*. Na página *Constraint* do wizard, clique no botão *ADD*. No campo *Check Name*, introduza: *valid_quantities*. No campo *Check Condition*, introduza: *quantity > 0*

  Clique no botão *OK*. Deverá ver o sumário da restrição que criou na página *Constraint* do *wizard*. Clique no botão *Next* para continuar para a próxima página do *wizard*.

- Pode continuar através do *wizard*, alterando os outros parâmetros da tabela. Alternativamente, pode saltar para página *Summary*, ou simplesmente clicar no botão *Finish* para criar a tabela.

- No *Control Center*, clique na pasta *Tables* dentro da base de dados *SAMPLE* na árvore de objectos. A tabela recém criada deverá aparecer na lista. Pode ser necessário fazer um *refresh* na *view* do *Control Center* para ver as alterações.

8.3 *Views*

Uma *view* é uma representação dos dados em tabelas. Os dados para uma *view* não são guardados separadamente, mas são obtidos quando a *view* é invocada. *Views* aninhadas, isto é, uma *view* criada com base em outras *views*, são suportadas. Toda a informação sobre *views* é guardada nos seguintes catálogos de *views* do DB2: SYSCAT.VIEWS, SYSCAT.VIEWDEP, e SYSCAT.TABLES. Eis um exemplo de como criar e usar uma *view*.

```
CONNECT TO MYDB1;

CREATE VIEW MYVIEW1
  AS SELECT ARTNO, NAME, CLASSIFICATION
  FROM ARTISTS;

SELECT * FROM MYVIEW1;

Output:

ARTNO            NAME                 CLASSIFICATION
------           -----------------    --------------
  10             HUMAN                  A
  20             MY PLANT               C
  30             THE STORE              E
  ...
```

8.4 *Indexes*

Um *index* é um conjunto de chaves ordenadas que apontam para uma linha numa tabela. Um *index* permite unicididade, e também melhora o desempenho. Podemos definir algumas características dos *indexes*:

- A ordem do *index* pode ser ascendente ou descendente
- As chaves do *index* podem ser únicas ou não-únicas
- Várias colunas podem ser usadas para o *index* (*index* composto)
- Se o *index* e os dados estão agregados numa sequencialmente de modo semelhante, então formam um *index* agregado (*clustered index*).

Por exemplo:

```
CREATE UNIQUE INDEX artno_ix ON artists (artno)
```

8.4.1 *Design Advisor*

O *Design Advisor* é uma ferramenta excelente para aconselhar sobre o *design* óptimo da base de dados para uma dada carga de trabalho baseada em SQL. O *design advisor* pode ajuda-lo com o *design* dos seus *indexes*, *Materialized Query Tables* (MQTs), *Multi-dimension clustering* (MDC), e a funcionalidade de particionamento da base de dados. O *Design Advisor* é invocado através do *Control Center*; botão direito sob a base de dados e seleccione "*Design Advisor*" como ilustra a Figura 8.6.

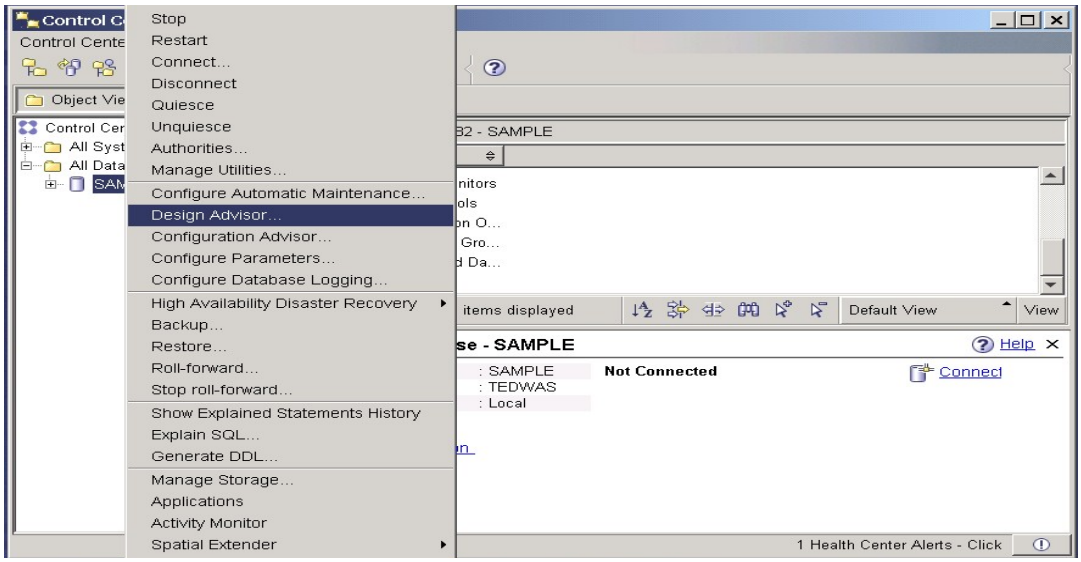


Figura 8.6 – Invocar o *Design Advisor* através do *Control Center*

A Figura 8.7 mostra o *Design Advisor*. Siga os passos nesse *wizard* para obter as recomendações de *design* do DB2.

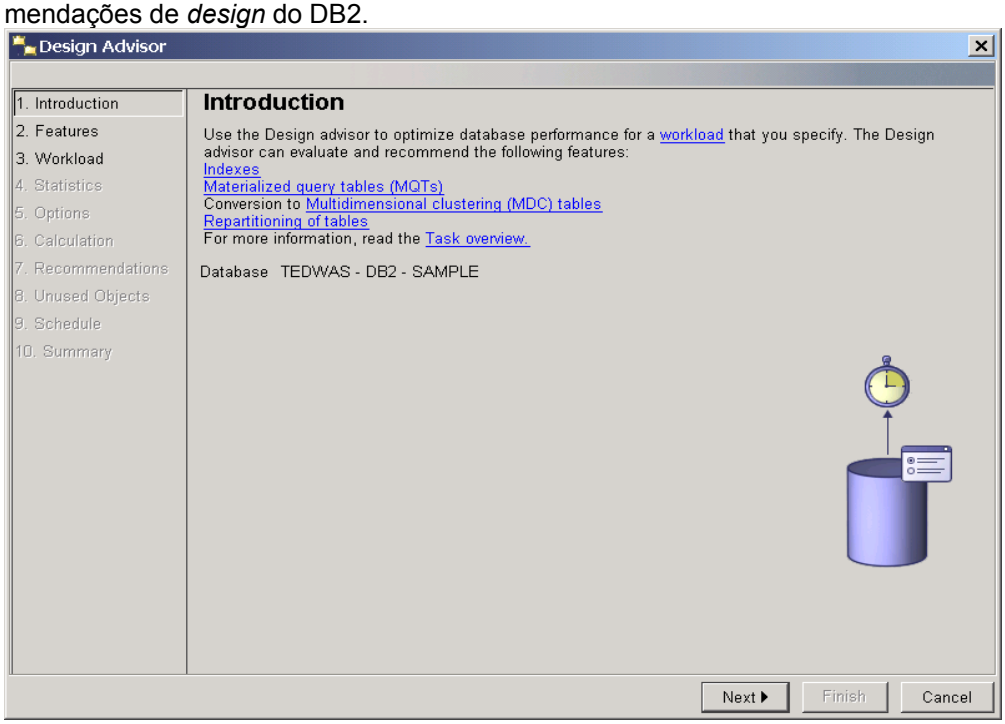
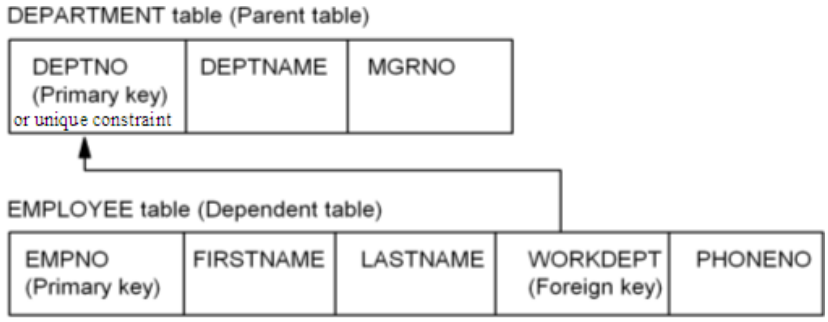


Figura 8.7 – O *Design Advisor*

8.5 Integridade referencial

Integridade referencial permite à base de dados gerir as relações entre tabelas. Pode estabelecer relações do tipo pai-filho entre tabelas como mostra a Figura 8.8. Na figura, há duas tabelas, DEPARTAMENT e EMPLOYEE, relacionadas pelo número do departamento. A coluna WORKDEPT na tabela EMPLOYEE só pode conter números de departamentos que já existam na coluna DEPTNO da tabela DEPARTAMENT. Isto porque neste exemplo, a tabela DEPARTAMENT é a tabela pai, e a tabela EMPLOYEE é a tabela filho, ou dependente. A figura também mostra o comando CREATE TABLE apropriado para estabelecer a relação na tabela EMPLOYEE.




```
create table employee (empno .....
  primary key (empno)
   foreign key (workdept)
   references department on delete no action)
in DMS01
```

Figura 8.8 – Exemplo de integridade referencial entre tabelas

Em integridade referencial, os seguintes conceitos são muitas vezes usados

Conceito	Descrição
Tabela Pai	Tabela de dados de controlo onde está a a chave pai
Tabela dependente	Tabela dependente dos dados na tabela pai. Também contém a chave estrangeira. Para uma linha existir numa tabela dependente, uma linha correspondente existe previamente na tabela pai.
Chave primária	Define a chave pai da tabela pai. Não pode conter valores NULL e os valores são únicos. A chave primária consiste numa ou mais colunas na tabela.
Chave estrangeira	Referencia a chave primária de uma tabela parente.

Dados em tabelas podem ser relacionados com dados numa ou mais tabelas com integridade referencial. Podem ser impostas restrições nos valores dos dados para que eles respeitem certas propriedades ou regras comerciais. Por exemplo, se uma coluna de uma ta-

bela guarda o sexo de uma pessoa, a restrições pode reforçar que os valores permitidos sejam “M” para masculino, e “F” para feminino.

9

Capítulo 9 – Ferramentas para movimentação de dados

As ferramentas ou comandos descritos nesta secção são utilizados para mover dados dentro da mesma base de dados ou entre base de dados na mesma ou em diferentes plataformas. A Figura 9.1 ilustra de forma geral as ferramentas na movimentação de dados.

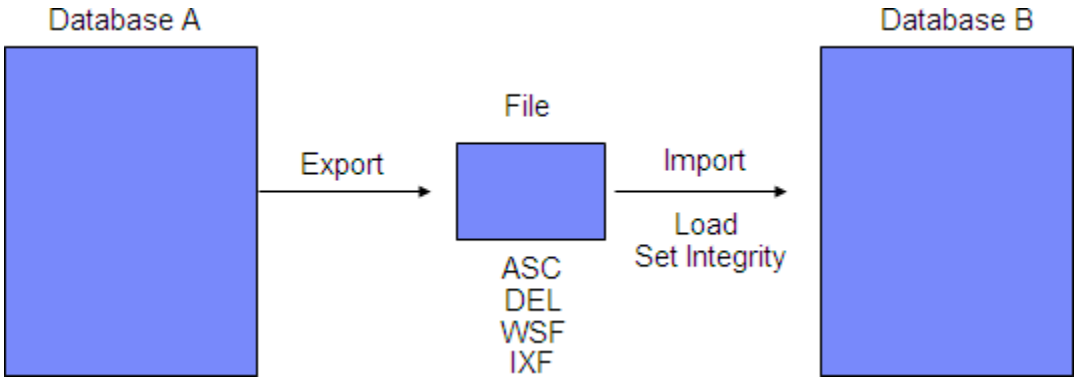


Figura 9.1 – Ferramentas na movimentação de dados

A Figura 9.1 ilustra duas bases de dados, A e B. Com a ferramenta EXPORT, podemos exportar os dados de uma tabela para um ficheiro. O ficheiro pode ter um dos seguintes formatos:

ASC = ASCII
DEL = Delimited ASCII
WSF = Worksheet Format
IXF = Integrated Exchange Format

Os ficheiros ASC e DEL são ficheiros de texto que podem ser abertos e revistos por qualquer editor de texto. WSF é um formato que pode mover dados para folhas de cálculo como Excel ou Lotus® 1-2-3. IXF é um formato que não só contém dados como também a Linguagem de Definição de Dados (LDD) [1] da tabela em questão. Isto é útil quando a tabela precisa de ser reconstruída, porque pode ser realizado directamente a partir de um ficheiro com o formato IXF, ao contrário dos outros formatos.

Tendo um ficheiro com os dados exportados, a ferramenta IMPORT pode ser usada para importar dados do ficheiro para outra tabela. A tabela tem de existir à priori para os formatos ASC, DEL e WSF, mas este requisito não é obrigatório para o formato IXF. Outro método para carregar dados para uma tabela é usar a ferramenta LOAD. A ferramenta LOAD é mais rápida pois vai directamente às páginas da base de dados sem interagir com o motor DB2; no entanto, este método não verifica restrições (*constraints*) e os *triggers* não serão accionados. Para garantir consistência ao carregar dados usando LOAD, o comando SET INTEGRITY é muitas vezes usado depois.

---

[1] É mais comum o termo em inglês, Data Definition Language (DDL). Iremos representar a sigla na forma inglesa.

As próximas secções descrevem as ferramentas EXPORT, IMPORT e LOAD detalhadamente.

Nota:

Para mais informações sobre como trabalhar com as ferramentas de movimentação de dados, veja este video:
http://www.channeldb2.com/video/video/show?id=807741:Video:4262

9.1 Ferramenta EXPORT

A ferramenta EXPORT é usada para extrair dados de uma tabela para um ficheiro num dos formatos apresentados anteriormente. Neste caso o que realmente está a ser processado é uma operação SQL SELECT. O exemplo seguinte exporta para o ficheiro *employee.ixf* do formato IXF, 10 linhas da tabela *employee*.

```
EXPORT TO employee.ixf OF IXF
   SELECT * FROM employee
   FETCH FIRST 10 ROWS ONLY
```

Recomendamos que teste o exemplo anterior. A tabela employee é parte da base de dados *SAMPLE*, por isso é necessário primeiro ligar-se a esta base de dados criada no capítulo anterior.
Se preferir trabalhar com as ferramentas GUI, a ferramenta EXPORT pode ser invocada através do *Control Center* ilustrada na Figura 9.2.

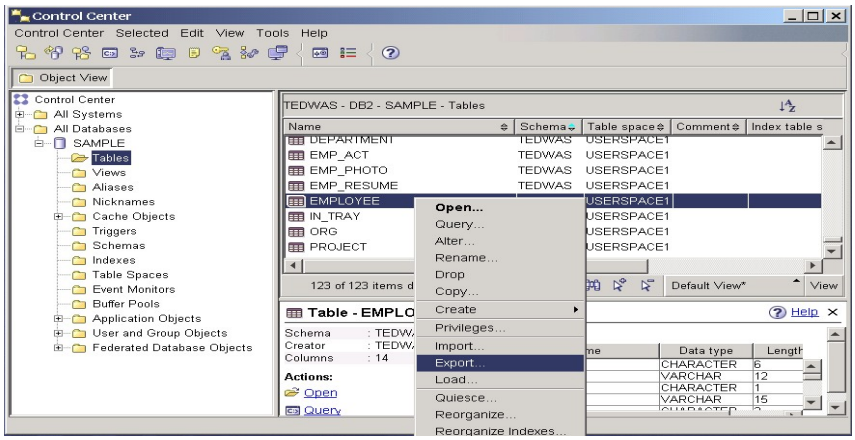


Figura 9.2 – Execução da ferramenta EXPORT

Como se pode visualizar na figura, primeiro selecciona-se a tabela *employee* com um clique, e depois botão direito na tabela para obter o menu pop-up onde se pode escolher a opção EXPORT. Depois de escolher esta opção, uma janela com instruções aparecerá. Basta apenas seguir as instruções para completar a operação.

9.2 Ferramenta IMPORT

A funcionalidade IMPORT é usada para carregar dados de um ficheiro para uma tabela. Neste caso, o que realmente se processa é uma operação SQL INSERT. Quando uma operação INSERT está a ser executada, todos os *triggers* são activados, todas as restrições são aplicadas imediatamente, e a *buffer pool* da base de dados é usada. O exemplo seguinte carrega todos os dados de um ficheiro com o formato IXF *employee.ixf* para a tabela *employee_copy.* A opção REPLACE_CREATE é uma das muitas opções disponíveis com a ferramenta IMPORT. Esta opção irá substituir o conteúdo da tabela *employee_copy* se esta já existir antes do IMPORT ser executado, ou irá criar a tabela e carregar os dados se a tabela não existir. Para executar o exemplo abaixo é necessário executar o exemplo sobre o EXPORT da secção anterior. Recomendamos que teste o exemplo.

```
IMPORT FROM employee.ixf OF IXF
   REPLACE_CREATE
   INTO employee_copy
```

Se preferir trabalhar a partir do *Control Center*, pode executar a ferramenta IMPORT seleccionando qualquer tabela, carregar no botão direito sobre esta, e escolher a opção IMPORT como ilustra a Figura 9.3

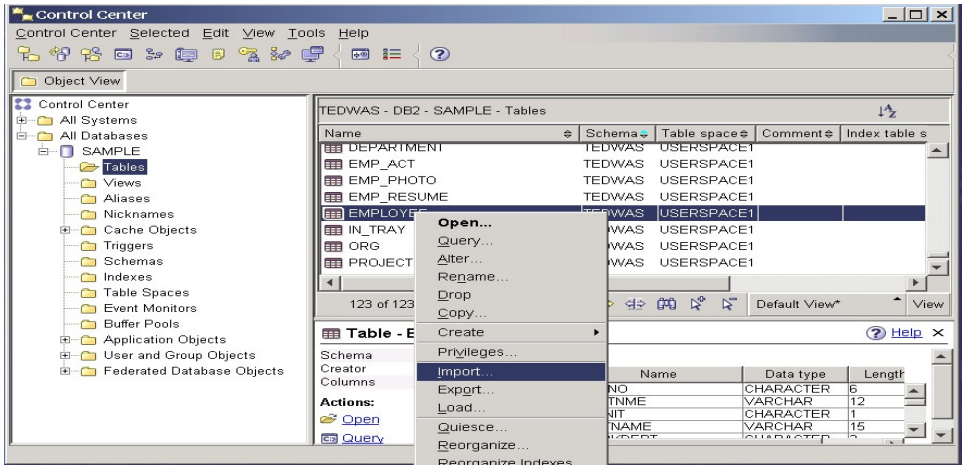


Figura 9.3 – Execução da ferramenta IMPORT

9.3 LOAD

A funcionalidade LOAD é uma maneira mais rápida de carregar dados para uma tabela. Como discutido anteriormente, a ferramenta LOAD não interage com o motor DB2, e portanto os *triggers* não são activados, a *bufferpool* não é usada e as restrições podem ser aplicadas mas apenas como um procedimento diferente. Por outro lado. a operação LOAD é mais rápida do que a IMPORT porque é um carregador de dados de baixo nível ,*low level data loader*, que acede directamente ás páginas de dados no disco. Esta operação é um produto de três fases: LOAD, BUILD, e DELETE.

O exemplo seguinte carrega todos os dados de um ficheiro IXF chamado *employee.ixf* para uma tabela *employee_copy*. A opção REPLACE é uma das muitas opções disponíveis com a LOAD. Neste caso é usada para substituir o conteúdo da tabela *employee_copy.*

```
LOAD FROM employee.ixf OF IXF
REPLACE INTO employee_copy
```

Depois de executar o comando acima (recomendado), o *tablespace* onde ela reside pode estar com o estado CHECK PENDING. Isto quer dizer que precisa de correr o comando SET INTEGRITY para verificar a consistência dos dados, exemplificado a seguir:

```
SET INTEGRITY FOR employee_copy
   ALL IMMEDIATE UNCHECKED
```

Se preferir trabalhar através do *Control Center*, pode executar as operações LOAD e SET INTEGRITY, ilustradas nas Figuras 9.4 e 9.5 respectivamente.

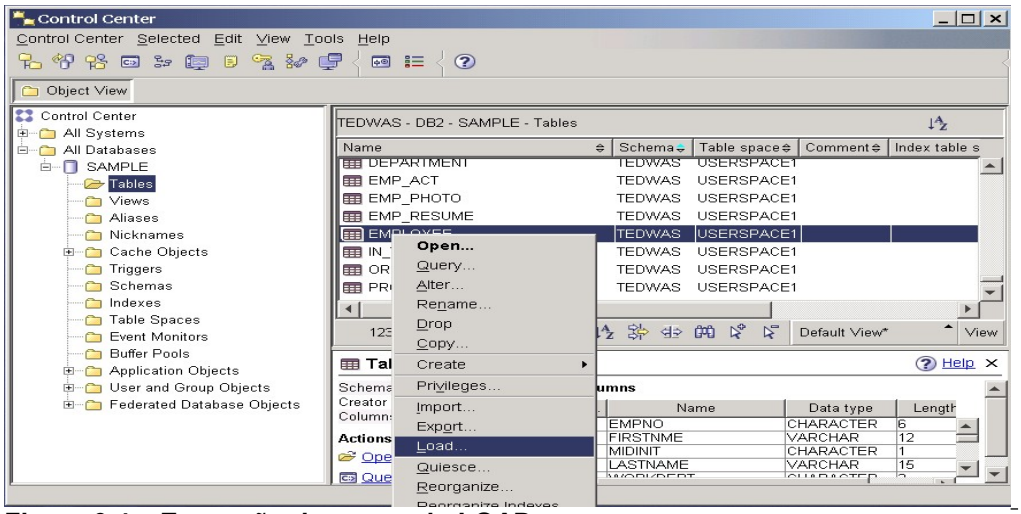


Figura 9.4 – Execução do comando LOAD

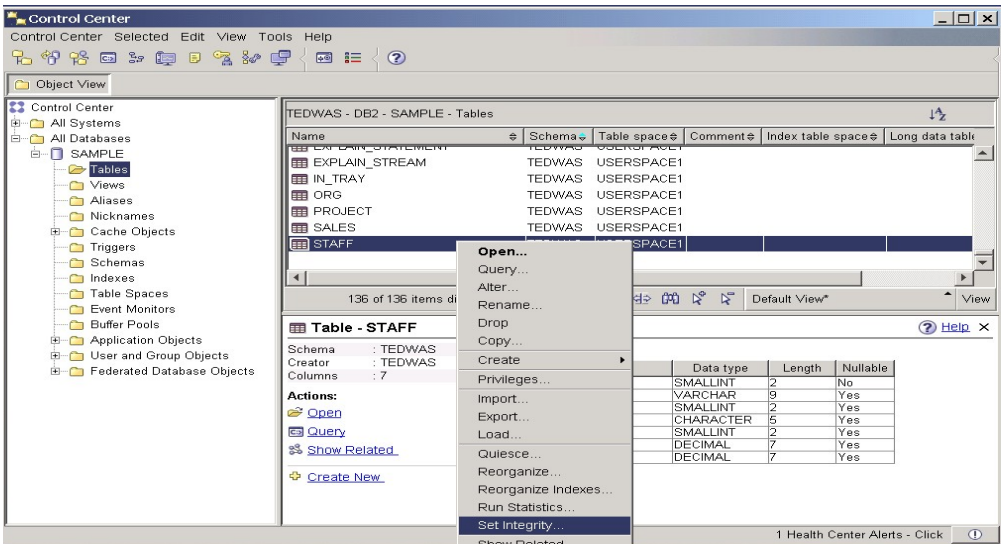


Figura 9.5 – Execução do comando SET INTEGRITY

9.4 A ferramenta *db2move*

As ferramentas EXPORT, IMPORT e LOAD só funcionam para uma tabela de cada vez. Para não escrever um script que iria aplicar um dos comandos para cada uma das tabelas da base de dados, existe uma ferramenta chamada *db2move* que já faz isso. A ferramenta *db2move* apenas funciona para ficheiros IXF, e o nome dos ficheiros serão gerados automaticamente pelo *db2move*. Os exemplos que se seguem mostram como utilizar *db2move* com as opções *export* e *import*, respectivamente usando a base de dados SAMPLE.

```
db2move sample export
db2move sample import
```

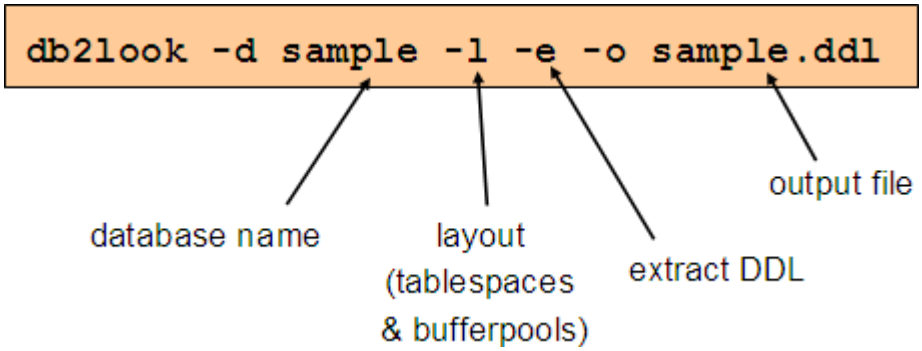
O *Control Center* não tem a opção *db2move.*

9.5 A ferramenta *db2look*

Enquanto as ferramentas EXPORT, IMPORT, LOAD e *db2move* permitem a movimentação de dados de uma tabela para outra, dentro de uma base de dados ou mesmo através de várias, a ferramenta *db2look* pode ser usada extrair comandos DDL, estatísticas da base de dados e as características do *table space* para uma base de dados e guarda-los num script que pode mais tarde ser executado noutro sistema. Por exemplo, se quiser fazer um clone a uma base de dados que está a correr num servidor DB2 na plataforma Linux para outro a correr em Windows; poderá primeiro executar a ferramenta *db2look* no servidor DB2 Linux para obter a estrutura da base de dados e guardar essa estrutura num ficheiro. De seguida, basta copiar o esse mesmo ficheiro para o servidor DB2 Windows, e executá-lo para começar a construir o clone da base de dados. Neste ponto, a estrutura da base de dados foi clonada. O próximo passo é executar a ferramenta *db2move* com a opção *export* no servidor DB2 Linux, e copiar todos os ficheiros gerados para o servidor DB2 Windows, e executar aí a ferramenta *db2move* com as opções *import* ou *load.* A base de dados estará agora completamente clonada de um servidor para o outro em diferentes plataformas.

O cenário acima descrito pode ser necessário quando se trabalha com base de dados em plataformas diferentes como Linux e Windows. Se ambos os servidores estão a ser executados na mesma plataforma, geralmente usar-se-ia os comandos *backup* e *restore*, que tornam o processo mais fácil e simples. Os comandos *backup* e *restore* serão discutidos com mais detalhe no próximo capítulo.

Os exemplos seguintes extraem os *layouts* do *table space* e *bufferpool*, assim como os comandos DDL da base de dados SAMPLE, e guardam-nos no ficheiro "sample.dll". Recomendamos que execute o comando e analise o resultado no ficheiro de texto "sample.dll".



O comando *db2look* tem demasidas opções para serem analisadas neste livro; no entanto usando a *flag* -h pode-se obter uma breve descrição das opções disponíveis:

```
db2look -h
```

A ferramenta *db2look* pode também ser invocada através do *Control Center* (Figura 9.6).

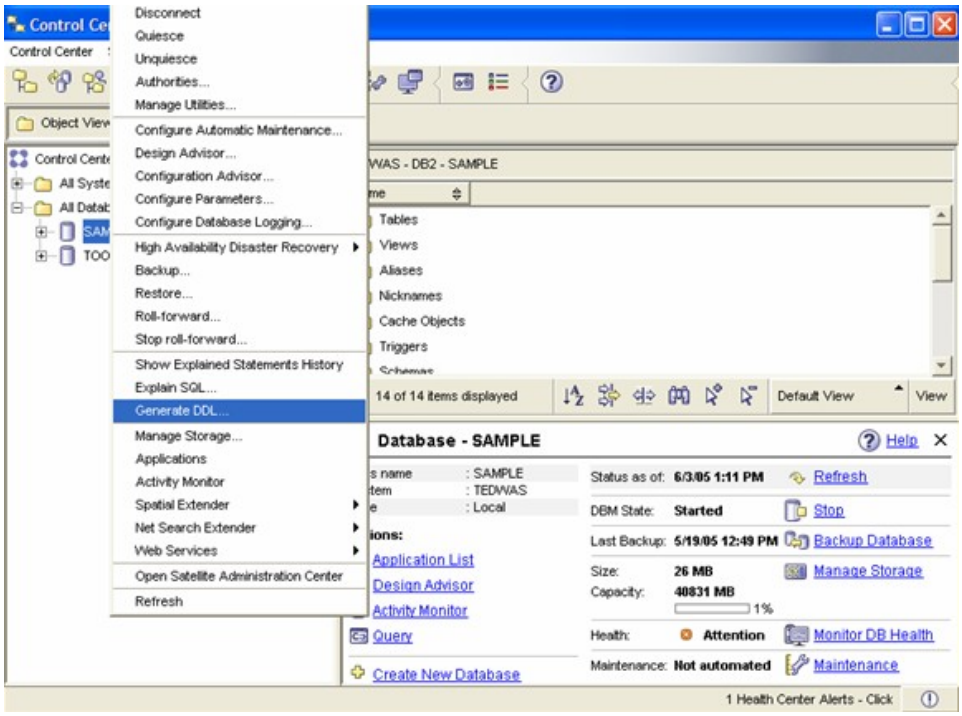


Figura 9.6 – Extracção da DDL através do *Control Center*

Na Figura 9.6, selecciona-se a base de dados sobre a qual queremos obter a DDL, botão direito na opção, e seleccionar “Generate DDL”. A janela Generate DDL aparecerá, que mostra algumas opções de extracção ilustrada Figura 9.7

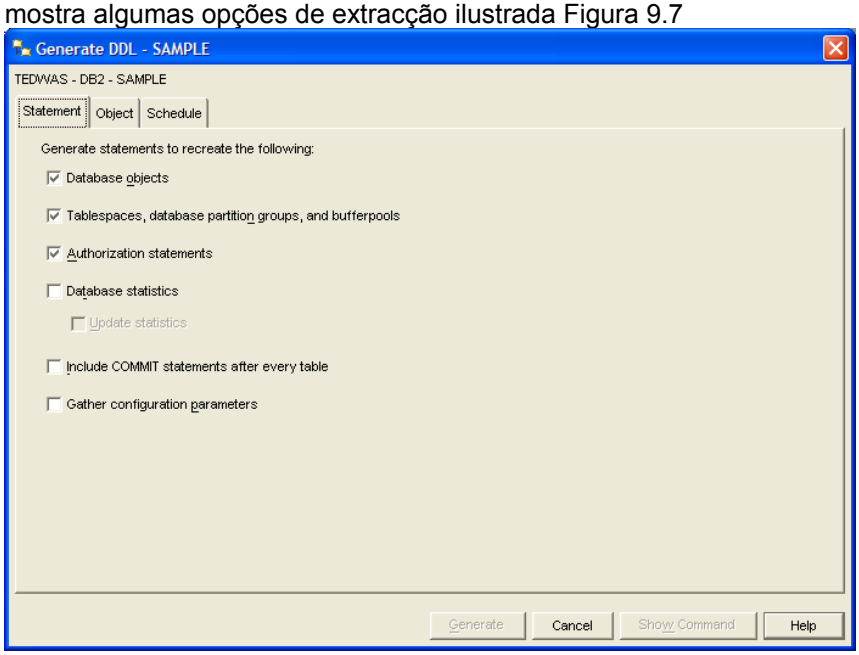


Figura 9.7 – Extracção da DDL através do *Control Center*

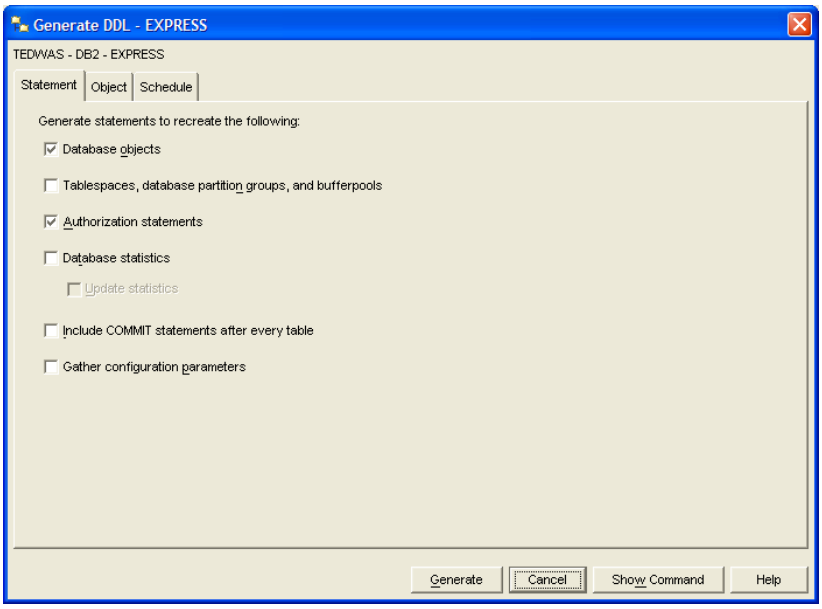
Quicklab #8 – Extracção da DDL da base de dados EXPRESS

Objectivo

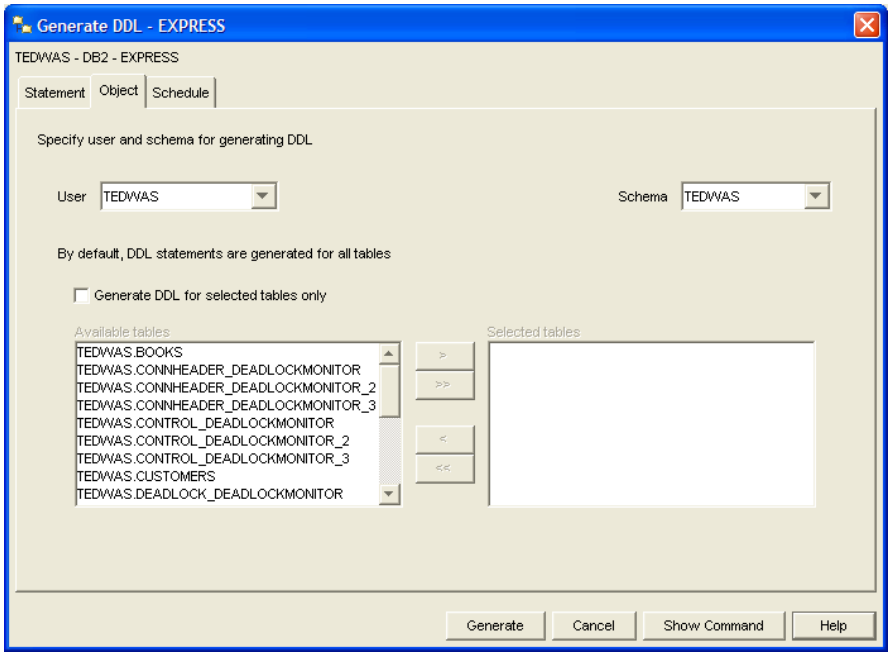
Quando se faz o clone a uma base de dados, o objectivo deverá ser criar a re-criação da base de dados mais directa e repetível possível. Isto é normalmente produzido usando scripts SQL, que pode ser executados imediatamente após a instalação do DB2. Neste Quicklab, irá extrair as definições do objecto (object definitions) da base de dados *EXPRESS* (criada na Quicklab #2) usando o *Control Center*.

Procedimento

- Abrir o *Control Center*.
- Clique com o botão direito na base de dados *EXPREESS* na árvore de objectos e seleccionar *Generate DDL.* Isso irá lançar a janela interactiva *Generate DDL*.
- Na janela *Generate DDL*, especifique as opções para gerar a DDL, como ilustra a figura. Se criou objectos adicionais no seu ambiente, tal como *table spaces*, *buffer pools*, etc., deve selecciona-los também. Como estes não foram criados, não seleccionamos a respectiva caixa. Estatísticas da base de dados não foram incluídas porque o ambiente de produção irá provavelmente conter um conjunto diferente de estatísticas do ambiente de desenvolvimento. Similarmente, os parametros de configuração irão ser diferentes também. No seu ambiente, se tudo estiver exactamente configurado da forma que irá ser desenvolvido, poderá escolher estas opções.

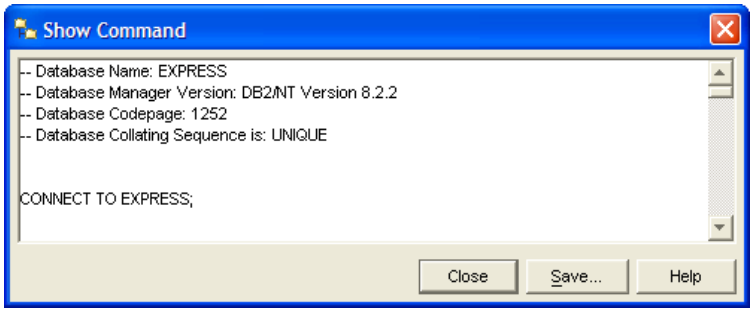


- Vá para o separador *Object*. Podemos escolher especificamente quais sobre quais objectos queremos gerar a DDL. Neste caso, seleccione um utilizador e o *schema* que usou para criar todos os objectos e gere a DDL para todos os objectos desse *schema*. Clique no botão *Generate* para iniciar o processo.



- Reveja a DDL resultante. O resultado do passo anterior é um único script com todos os comandos SQL para os objectos escolhidos. Irá agora organizar o script em grupos lógicos.

- Crie a directoria `C:\express` no seu sistema de ficheiros e guarde o ficheiro com a DDL gerada nesta directoria para um ficheiro chamado `schema.ddl`. (Clique no botão *Save*)



- Abra o novo ficheiro no *Command Editor*. (Atalho/Dica: No *Command Editor*, escolha *File* => *Open*)

- Apesar de apenas queremos a DDL para as tabelas, irá reparar que foram incluídas as DDL para outros objectos de base de dados. Mova todos os comandos CREATE TRIGGER para um ficheiro novo chamado `triggers.ddl`. Mesmo que só tenhamos criado apenas um *trigger*, é um procedimento útil e aconselhável separar todos os objectos por tipo.

- Por agora, também recomendamos que remova todos:

11. CONNECT TO comandos base de dados
12. DISCONNECT comandos

Neste ponto deverá ter dois scripts:

DDL para as tabelas, *views*, *indexes*, e restrições
`C:\express\schema.ddl`

DDL para *triggers*
`C:\express\triggers.ddl`

- Optimize o script para distribuição:

- Remova comentários desnecessários (e.g. -- CONNECT TO...)

Separe as funções e os procedimentos em diferentes ficheiros (útil quando temos muitas funções e procedimentos). Poderá também querer agrupa-los por função ou aplicação (e.g. `billing.ddl`, `math.ddl`, `stringfunc.ddl`, etc.)

13. Notará que um caracter especial é usado para delimitar o fim dos *triggers*, funções e procedimentos (@). Isto é necessário para delimitar o final do comando `CREATE <object>` em oposição ao final de um comando procedimental dentro do objecto.

10

Capítulo 10 – Segurança em base de dados

Este capítulo discute como é tratada a segurança em DB2. A Figura 10.1 fornece uma visão geral.

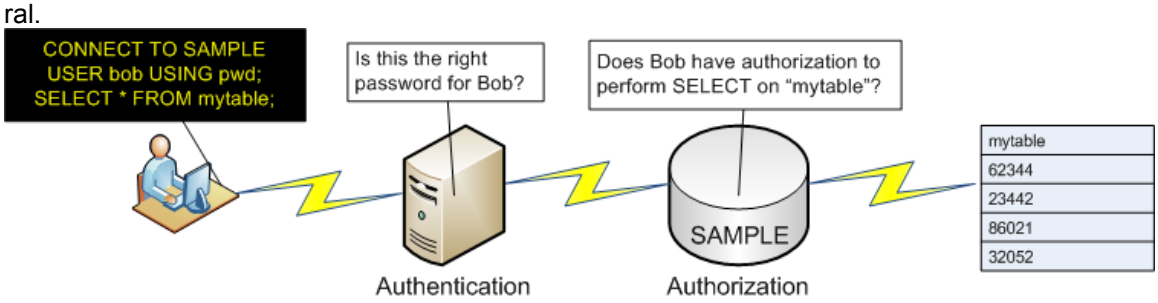


Figura 10.1 – Visão geral sobre a segurança em DB2

Como ilusta a Figura 10.1, a segurança em DB2 consiste em duas partes:

Autenticação
É o processo em que a identidade do utilizador é validada. A autenticação é efectuada através um sistema de segurança exterior ao DB2 (tipicamente pelo sistema operativo, algum método de autenticação pela rede, ou um *plugin* desenvolvido especificamente para autenticação). A autenticação baseada no SO é a opção por omissão. Quando se usa a autenticação baseada pelo SO, o nome de utilizador e a password são transferidos para o servidor da base de dados (e.g. como uma parte do comando *connect*). O servidor da base de dados invoca então a autenticação pelo SO para validar a identidade do utilizador e a password.

Autorização
Neste ponto, o DB2 verifica se o utilizador autenticado pode executar a operação requirida. A informação sobre a autorização é guardada num catálogo DB2 e num ficheiro de configuração DBM.

Por exemplo, na Figura 10.1, o utilizador “bob” liga-se à base de dados SAMPLE através do comando:

```
CONNECT TO sample USER bob USING pwd
```

Ambos os argumentos, “bob” e “pwd”, são passados para o sistema operativo ou para uma entidade exterior de autenticação que aprova a autenticação, verificando se o nome do utilizador “bob” já está definido, e se a password dada corresponde à do utilizador. Se este processo for bem sucessido, o sistema operativo irá retornar o controlo de segurança para o DB2. De seguida, o utilizador executa um comando como:

```
SELECT * FROM mytable
```

Neste momento, o DB2 torna-se responsável pelo controlo de segurança e realiza o processo de verificação da autorização, para confirmar se o utilizador “bob” possui o privilégio SELECT sobre a tabela “mytable”. Se a verificação da autorização falhar, o DB2 irá retornar uma mensagem de erro. Caso contrário, a instrução será executada sobre a tabela “mytable”.

Nota:

Para mais informação sobre como trabalhar com segurança em DB2, veja este video: http://www.channeldb2.com/video/video/show?id=807741:Video:4267

10.1 Autenticação

Apesar da autenticação actual ser realizada pelo sistema operativo (ou por outro mecanismo de segurança externo), o DB2 decide em que nivel é que esta autenticação ocorre.

O parâmetro AUTHENTICATION no DBM CFG, accionado no servidor DB2, tem um leque de valores possíveis. Por exemplo, quando o parâmetro é configurado como SERVER (por omissão), a autenticação é realizada pelo sistema operativo ou mecanismo exterior de segurança no servidor. No entanto, se a AUTHENTICATION é configurada como CLIENT, a autenticação é realizada pelo sistema operativo ou mecanismo exterior de segurança no cliente. Este facto é ilustrado pela Figura 10.2.

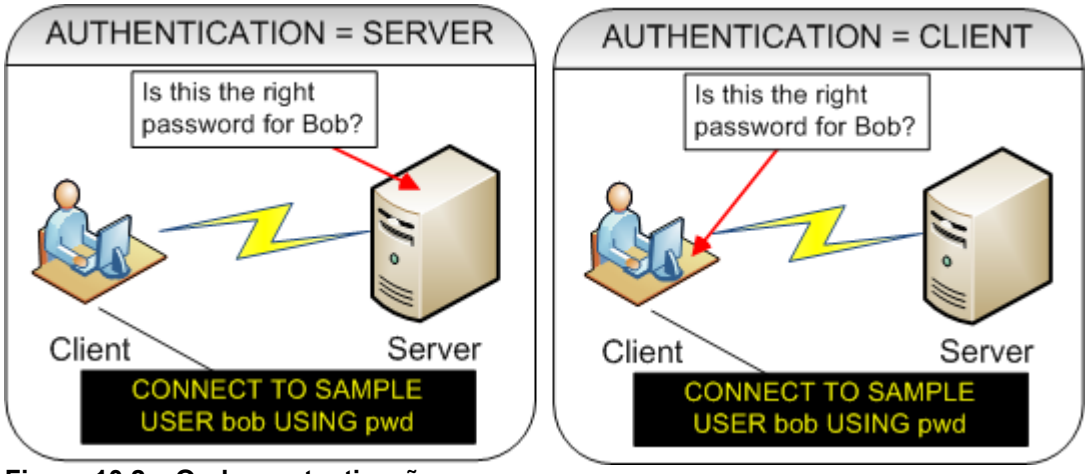


Figura 10.2 – Onde a autenticação ocorre

O parâmetro AUTHENTICATION pode configurado com qualquer dos valores listados na Tabela 10.1

Comando	Descrição
SERVER (default)	Autenticação ocorre no servidor

CLIENT	Autenticação ocorre no cliente
SERVER_ENCRYPT	Igual ao SERVER mas os *users Ids* e as *passwords* estão encriptadas
KERBEROS	Autenticação ocorre usando o mecanismo externo de segurança Kerberos
SQL_AUTHENTICATION_DATAENC	A autenticação é efectuada no servidor e as ligações tem que usar encriptação de dados
SQL_AUTHENTICATION_DATAENC_CMP	Como acima, excepto na encriptação de dados que apenas é usada quando disponível
GSSPLUGIN	Autenticação utiliza um mecanismo de segurança externo GSS API-based plug-in.

Tabela 10.1 – Valores dos pârametos para uma AUTHENTICATION válida

10.2 Autorização

A autorização consiste nos privilégios e autoridades que estão guardados nas tabelas do sistema DB2 e são geridas pelo DB2.

Um privilégio permite ao utilizador executar um tipo simples de operação na base de dados, tal como CREATE, UPDATE, DELETE, INSERT, etc.

Uma autoridade é um papel pré-definido que consiste em vários privilégios. A Figura 10.3 mostra as diferentes autoridades e privilégios no DB2.

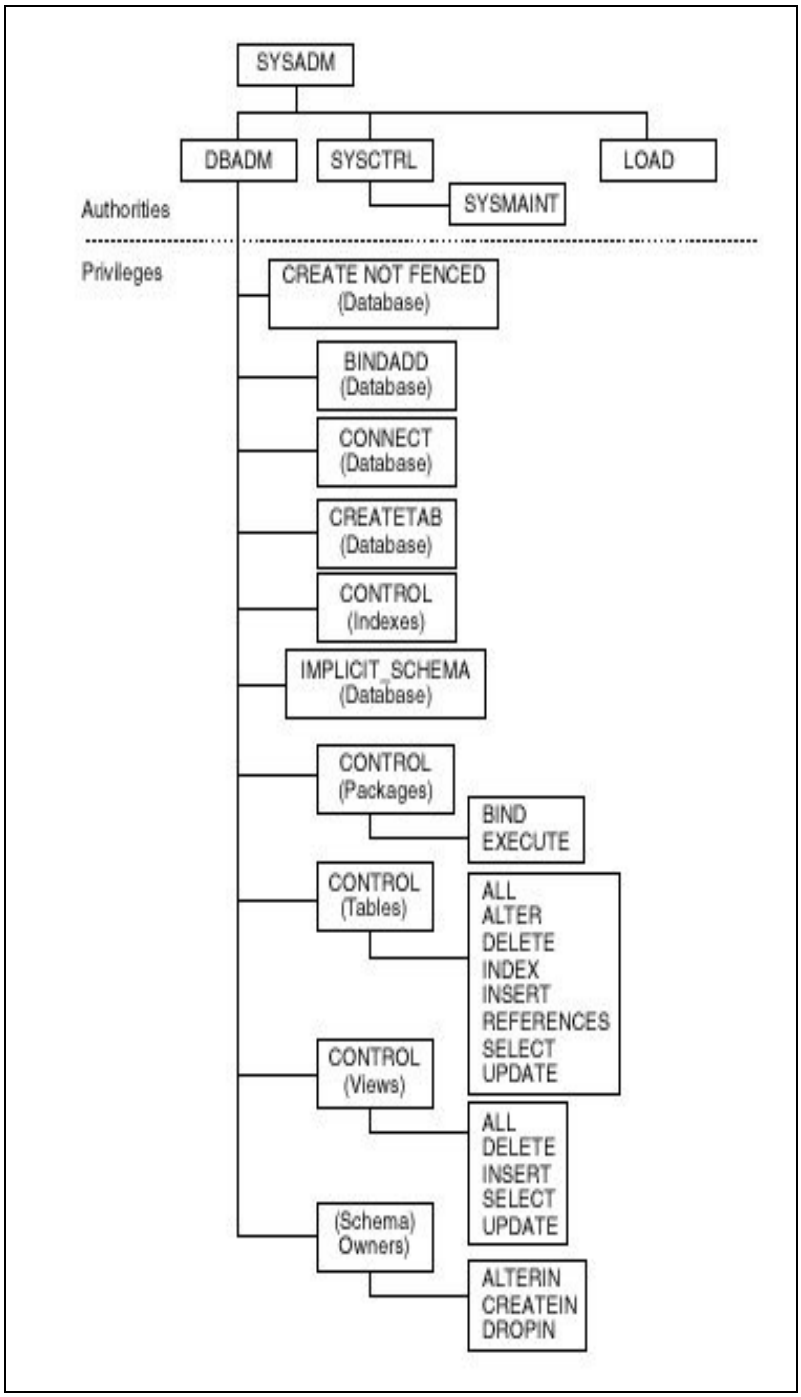


Figura 10.3 – Autoridades e privilégios

A Tabela 10.2 mostra as diferentes funções que cada autoridade pode realizar. Como se pode ver, SYSADM possui a maior autoridade enquanto SYSMON a menor.

Função	*SYSADM*	*SYSCTRL*	*SYSMAINT*	*SYSMON*	*DBADM*	*LOAD*
Update DBM CFG	Y					
Grant/revoke DBADM	Y					
Establish/change SYSCTRL	Y					
Establish/change SYSMAINT	Y					
Establish/change SYSMON	Y					
Force users off database	Y	Y				
Create/drop database	Y	Y				
Restore to new database	Y	Y				
Update DB CFG	Y	Y	Y			
Backup database/table space	Y	Y	Y			
Restore to existing database	Y	Y	Y			
Perform roll-forward recovery	Y	Y	Y			
Start/stop instance	Y	Y	Y			
Restore table space	Y	Y	Y			
Run trace	Y	Y	Y	Y		
Obtain monitor snapshots	Y	Y	Y			
Query table space state	Y	Y	Y			
Prune log history files	Y	Y	Y			
Quiesce table space	Y	Y	Y		Y	Y
LOAD tables	Y				Y	Y
Set/unset check pending state	Y				Y	
Create/drop event monitors	Y				Y	

Tabela 10.2 – Autoridades e privilégios em DB2

Para garantir a autoridade SYSADM, SYSCRTL ou SYSMAINT a um grupo, os parâmetros de DBM CFG, SYSADM_GROUP, SYSCRTL_GROUP, e SYSMAINT_GROUP pode ser atribuidos a um grupo do sistema operativo.

Por exemplo, para dar a autoridade SYSADM ao grupo “db2admns” do sistema operativo, basta executar o comando:

```
update dbm cfg using SYSADM_GROUP db2admns
```

Cada instância do DB2 tem as suas definições de autoridade de grupo.

Em Windows, estes parâmetros estão vazios por omissão, o que significa que o grupo local de Administradores do Windows será SYSADM. Em Linux, a instância root do grupo é a default grupo SYSADM.

10.3 Autoridade DBADM

A autoridade DBADM (*DataBase ADMinistrator*) é a de super utilizador para a base de dados. Esta não é uma autoridade ao nivel da instância; por isso não está listada na secção anterior. Para garantir a autoridade DBADM, utilize o comando GRANT como mostra o exemplo abaixo.

```
connect to sample
grant DBADM on database to user <userid>
```

Repare que é necessária uma ligação à base de dados, a base de dados “sample” neste caso, para poder garantir a autoridade DBADM a um utilizador. Apenas um utilizador com privilégios SYSADM pode garantir a autoridade DBADM.

Note que um DBADM não pode criar *table spaces*, apesar de serem objectos dentro da base de dados, porque uma *table space* lida com *containers* (disco) e *buffer pools* (memória) que são recursos fisícos do sistema.

10.4 O grupo PUBLIC

O DB2 define um grupo interno chamado PUBLIC. Qualquer utilizador identificado pela autenticação do sistema operativo ou da rede é implicitamente um membro do grupo PUBLIC. Quando uma base de dados é criada, certos privilégios estão garantidos automaticamente ao grupo PUBLIC:

- CONNECT,
- CREATETAB,
- IMPLICIT SCHEMA,
- BINDADD

Para segurança adicional, recomendamos que revogue estes privilégios do grupo PUBLIC como se mostra abaixo:

```
REVOKE CONNECT             ON DATABASE FROM PUBLIC
REVOKE CREATETAB           ON DATABASE FROM PUBLIC
REVOKE IMPLICIT_SCHEMA     ON DATABASE FROM PUBLIC
REVOKE BINDADD             ON DATABASE FROM PUBLIC
```

10. Os comandos GRANT e REVOKE

Os comandos GRANT e REVOKE são parte do standard SQL, e são usados para dar ou remover privilégios a um utilizador ou grupo. Seguem-se alguns exemplos desses comandos:

Para garantir o privilégio SELECT na tabela T1 ao utilizador USER1:

```
GRANT SELECT ON TABLE T1 TO USER user1
```

Para garantir todos os privilégios na tabela T1 ao grupo GROUP1:

```
GRANT ALL ON TABLE T1 TO GROUP group1
```

Para revogar todos os privilégios na tabela T1 do grupo GROUP1:

```
REVOKE ALL ON TABLE T1 FROM GROUP group1
```

Para garantir o privilégio EXECUTE no procedimento p1 ao utilizador USER1:

```
GRANT EXECUTE ON PROCEDURE p1 TO USER user1
```

Para revogar o privilégio EXECUTE no procedimento p1 do utilizador USER1:

```
REVOKE EXECUTE ON PROCEDURE p1 FROM USER user1
```

10.6 Verificação da Autorização e Privilégio

A maneira mais fácil de verificar a autorização e os privilégios é através do *Control Center*. A Figura 10.4 mostra como verificar os privilégios para a tabela EMPLOYEE no *Control Center*.

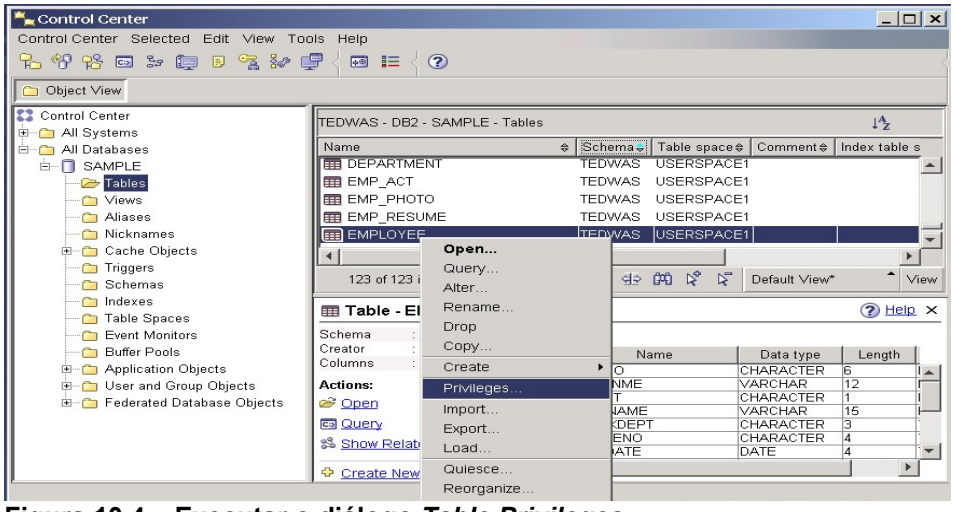


Figura 10.4 – Executar o diálogo *Table Privileges*

Como ilustra a Figura 10.4, seleccionamos a tabela que desejamos, botão direito sobre esta, e escolhemos a opção *Privileges.* Uma vez seleccionada, a caixa de diálogo *Table Privileges* aparece como mostra a Figura 10.5. Esta figura também explica os diferentes campos e elementos da caixa de diálogo.

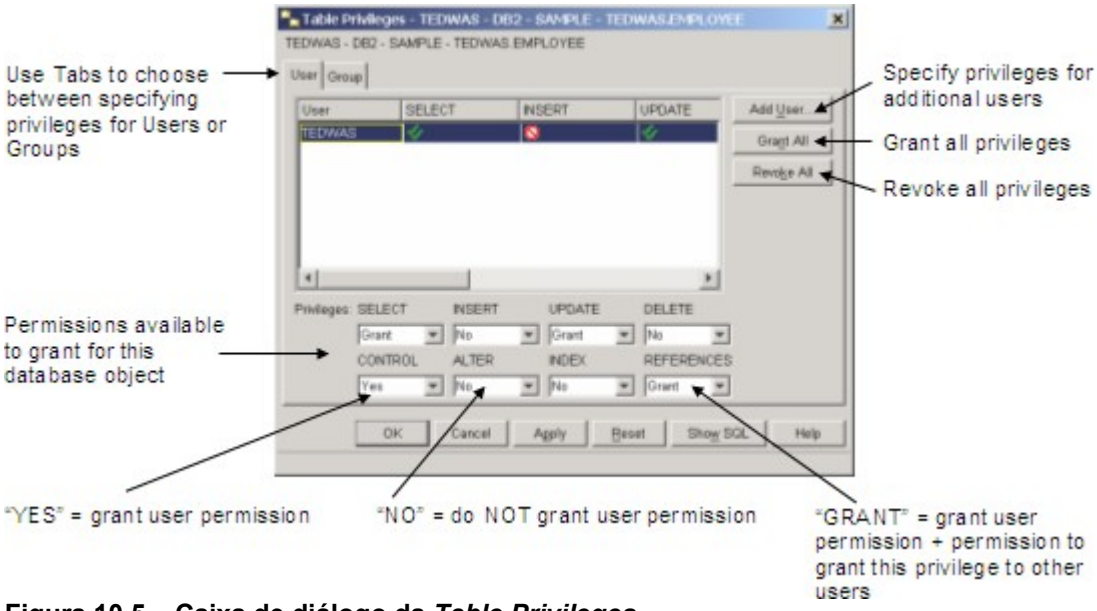


Figura 10.5 – Caixa de diálogo da *Table Privileges*

Alternativamente, pode fazer uma consulta às *views* do catálogo DB2 SYSCAT que contém a informação de autorização. Por exemplo, se quisermos saber se o utilizador DB2ADMIN possui o prvilégio SELECT na tabela T2, e saber quem garantiu este privilégio, podemos correr a seguinte consulta:

```
SELECT grantor, grantee, selectauth
  FROM syscat.tabauth
 WHERE tabname = 'T2'

GRANTOR            GRANTEE              SELECTAUTH
-------------------------------------------------
ARFCHONG           DB2ADMIN                  Y
```

No exemplo acima, o utilizador ARFCHONG garantiu o privilégio SELECT ao utilizador DB2ADMIN.

10.7 Considerações sobre os privilégios do Grupo

Para tornar mais fácil a administração do DB2, reuna utilizadores em grupos, e garanta aos grupos os privilégios requeridos.

Quando são garantidos privilégios a grupos, aos membros do grupo são implicitamente garantidos os privilégios inerentes através da relação de associação ao grupo.

Quando um utilizador é removido de um grupo, este perde os privilégios implícitos ao grupo, mas continua a possuir privilégios que lhe foram explicitamente garantidos: Privilégios que foram explicitamente dados a um utilizador precisam de ser revogados explicitamente do utilizador.

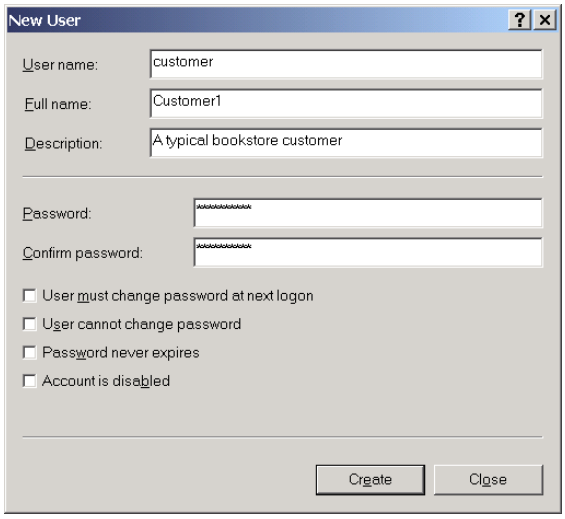
Quicklab #9 – Garantir e revogar privilégios ao utilizador

Objectivo

Até agora, estavamos a usar a conta da instância do administrador (SYSADM) para invocar todos os comandos da base de dados. Esta conta possui acesso total a todas as ferramentas, dados, e objectos da base de dados. Por isso, é muito importante salvaguardar esta conta para evitar perda de dados acidental ou deliberada. Na maioria dos casos, iremos criar uma conta de utilizador ou grupo diferente com permissões limitadas. Neste Quicklab, iremos criar uma nova conta de utilizador, e depois garantir privilégios específicos.

Procedimento

- Abra a consola de *Windows Computer Management* carregando com o botão direito no *My Computer* no desktop, e seleccionando o item *Manage*.
- Expanda a selecção *System Tools* na árvore da parte esquerda da janela e expanda a pasta *Local Users and Groups*. Botão direito na pasta *User* e seleccione o item *New User*.
- Na janela interactiva *New User*, insira a seguinte informação: no campo *User name,* insira "*customer*" e no campo *Full name*, insira "*Customer1*". No campo *Description,* insira "*A typical bookstore customer*". Nos campos *Password* e *Confirm password*, insira "*ibmdb2*". Retire o *checkmark* da opção *User must change password on next logon*, e clique no botão *Create* para criar um novo utilizador.



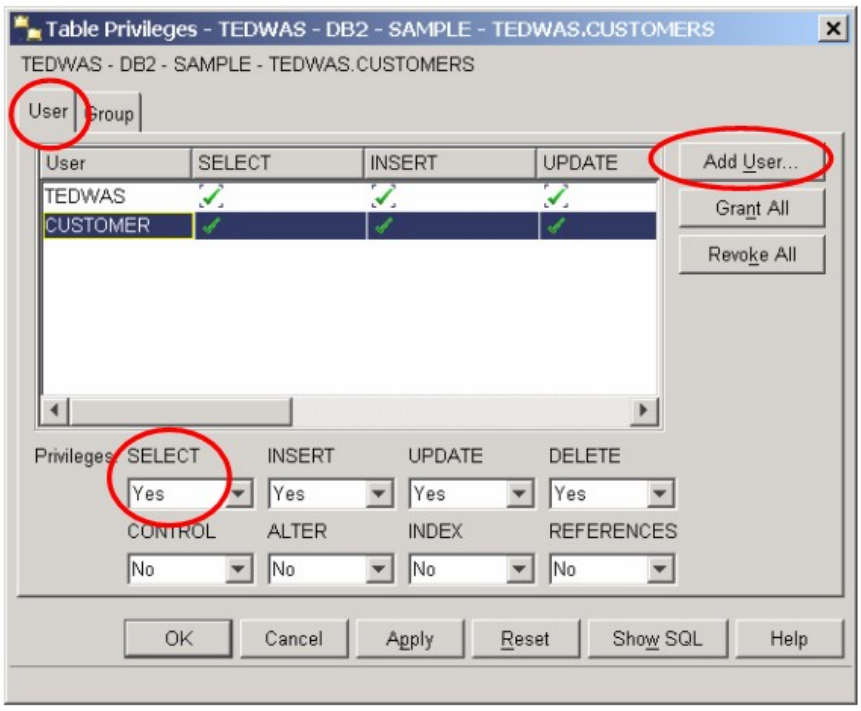
- Certifique-se que a vista avançada está a ser usada pelo *Control Center* do DB2. Para mudar para a vista avançada, seleccione o item *Customize Control Center* do menu *Tools* do Centro de Controlo. Seleccione a opção *Advanced* e clique no botão OK.

- Expanda a árvore de objectos do *Control Center* na parte esquerda para *All Databases* > *EXPRESS* > *Tables*.

- Garanta os privilégios requeridos ao utilizador recém criado. Da lista de tabelas na base de dados *EXPRESS*, botão direito na tabela *CUSTOMERS*, e seleccione o item *Privileges* para visualizar a janela interactiva *Table Privileges*.

- Clique no botão *Add User* e seleccione o utilizador *customer* recém criado. Clique no botão OK para fechar a caixa interactiva *Add User*.

- Neste ponto o utilizador *customer* foi adicionado á lista de utilizadores, mas ainda não possui qualquer privilegio. Para garantir os privilegios SELECT, INSERT, UPDATE, e DELETE ao utilizador, mude para *Yes* cada opção da caixa respectiva. Um cliente da Internet deveria ser apto a visualizar/adicionar/actualizar/remover os dados da sua conta. Nós não demos outras permissões ao utilizador porque não eram requeridas. Clique no botão *OK* para fechar a janela interactiva *Table Privileges* e aceite as alterações que realizou.



- Repita os passos 7-9 para as tabelas *BOOKS* e *SALES*. Para a tabela *BOOKS*, apenas garanta o privilégio SELECT porque o cliente não deverá modificar os dados do inventário da loja. Para a tabela SALES, apenas garanta os privilégios SELECT e INSERT. O cliente NÃO deve possuir os privilégios DELETE ou UPDATE porque apenas os empregados da loja deverão poder modificar as transacções nas vendas.

- Ligue-se à base de dados usando o user ID *costumer* criado acima. Tente executar o comando SELECT aos dados da tabela costumers. O que acontece? Tente os comandos DELETE ou UPDATE aos dados na tabela *SALES*. O que acontece?

Neste Quicklab, apenas criamos um utilizador; no entanto, a sua aplicação poderá conter diferentes tipos de utilizadores. Experimente criar outros utilizadores e atribua-lhes privilegios. Poderá também criar grupos de utilizadores e atribuir privilegios a esses grupos, ao invém de atribuir especificamente a cada utilizador individual.

11

Capítulo 11 – Cópias de Segurança e Recuperação

Neste capítulo falaremos de *logging* em bases de dados DB2, como criar uma cópia parcial ou completa da base de dados utilizando a utilidade BACKUP, e como recuperar os dados usando a ferramenta RESTORE.

Nota:

Para mais informações sobre *logging*, *backup* e *recovery*, veja este vídeo:
http://www.channeldb2.com/video/video/show?id=807741:Video:4282

11.1 *Logging de Bases de Dados*

Se está a trabalhar com um editor de texto, sempre que deseja assegurar que documento foi guardado, carrega no botão "Guardar". No mundo das bases de dados, um comando COMMIT faz exactamente o mesmo. Sempre que executamos o comando COMMIT estamos a garantir que quaisquer alterações nos dados são guardadas algures.

Do mesmo modo, quando se trabalha com documentos de texto, por vezes vemos no canto inferior direito uma breve mensagem: "guardando automaticamente". No mundo das bases de dados, isto também acontece, porque qualquer operação que realize sobre os dados, como UPDATE, INSERT ou DELETE, será guardada algures à medida que as realiza.

Esse "algures" nos parágrafos precedentes refere-se aos *logs* da base de dados. Os *logs* das bases de dados são guardados no disco e são usados para registar acções de transacções. Se houver uma falha do sistema ou da base de dados, os *logs* são usados para repor os dados.

Figura 11.1 mostra uma visão geral do objectivo dos *logs* nas bases de dados.

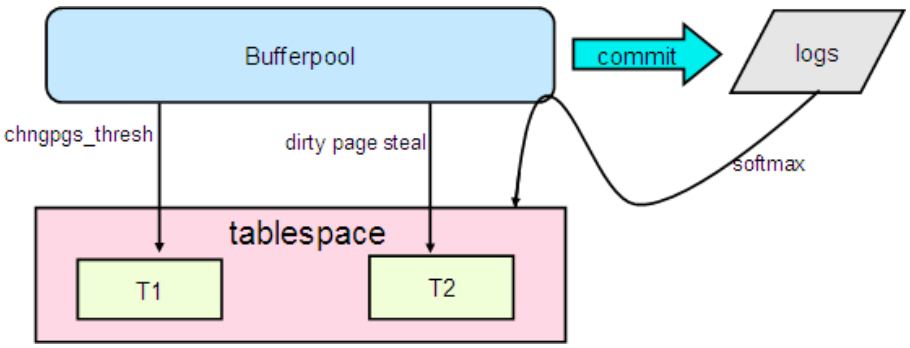


Figura 11.1 – *Logging* de bases de dados

Na Figura 11.1, podemos ver *tablespace* e *logs*. Ambos residem no disco, embora recomendamos que sejam alocadas em disco diferentes. Quando uma operação UPDATE é executada, por exemplo, as páginas para essa(s) linha(s) são trazidas para o *buffer pool* (memória). As alterações resultantes dessa operação serão tratadas no *buffer pool* e os valores novos e antigos são

guardados nos ficheiros de *log,* às vezes imediatamente, outras quando o *buffer* do *log* está cheio. Se um COMMIT vem depois de um UPDATE, o velho e novo valor será guardado nos ficheiros de *log* imediatamente. Este processo é repetido para muitas outras operações SQL executadas na base de dados. Só quando se juntam determinadas condições, como por exemplo, atingir a o número *threshold* de páginas de alterações especificado no parâmetro CHNGPGS_THRES, serão as páginas no *buffer pool* "exteriorizadas" ou escritas nos blocos de disco referentes ao *tablespace*. O parâmetro CHNGPGS_THRES indica a percentagem do *buffer pool* com páginas de "lixo", ou seja, páginas contendo alterações.

Do ponto de vista da eficiência, não faz sentido executar duas escrituras para cada operação de COMMIT: uma para escrever para os *logs*, e outra para escrever para o disco de *table space*; é por isto que a "exteriorização" dos dados para o disco de *table space* só ocorre quando parâmetros como CHNGPGS_THRES são atingidos.

11.2 Tipos de *logs*

Existem dois tipos de *logs*:

***Logs* primários**
Estes são pré-alocados e o número de *logs* primários disponíveis é determinado pelo parâmetro de db cfg LOGPRIMARY.

***Logs* secundários**
Estes são dinamicamente alocados quando necessário pelo DB2. O máximo número de *logs* secundários é definido pelo parâmetro db cfg LOGSECOND. Alocar dinamicamente é custoso; logo, para as operações do dia-a-dia, assegure-se que utiliza *logs* de alocação primária. Os ficheiros de *log* secundário são apagados quando todas as ligações à base de dados estão terminadas.

Logging infinito é uma possibilidade se definir o LOGSECOND para o valor -1; no entanto, esta não é uma opção recomendada, e pode ficar sem espaço no disco.

11.3 Tipos de *logging*

Existem dois tipos de *logging*: *logging* circular (por omissão) e *logging* de arquivo.

11.3.1 *Logging* circular

Figura 11.2 mostra como funciona o *logging* circular.

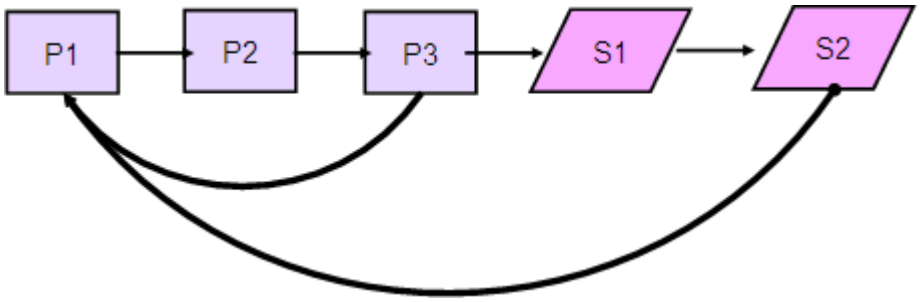


Figura 11.2 – Trabalhar com *logs* primários e secundários em modo circular

Na Figura 11.2 temos 3 *logs* primários, por isso podemos assumir que LOGPRIMARY foi definido como 3. Para simplificar, digamos que só está a ser executada uma transacção neste exemplo. À medida que esta é executada, o espaço começa a preencher o ficheiro de *log* P1, e depois o P2. Se um COMMIT ocorre e a informação é, mais tarde, exteriorizada para o disco, então P1 e P2 podem ser reescritos, porque a informação já não é necessária para recuperação de falhas/*crashes* (que serão discutidas mais à frente neste capítulo). Se, por outro lado, a transacção é tão grande que usa P1, P2, P3 e ainda necessita de mais espaço de *log* porque a transacção não foi submetida nem exteriorizada, então um *log* secundário (na figura, S1) será dinamicamente alocado. Se a transacção ainda assim continuar, mais *logs* secundários serão alocados até atingir o máximo de *logs* secundários estarem alocados. Se mais *logs* forem necessários, uma mensagem de erro indicando a condição de *log* totalmente preenchido será mostrada ao utilizador, e a transacção será desfeita.

11.3.2 *Log* de arquivo ou retenção

No *log* de arquivo, também conhecido como *log* de retenção, os ficheiros de *log* não são sobrescritos, mas mantidos *online* ou *offline.* Os *logs* de arquivo *online* são mantidos com *logs* activos que são ainda necessários para recuperação de falha. Os *logs* de arquivo *offline* são movidos para outro tipo armazenamento como *tapes*, e isto pode ser feito com rotinas USEREXIT. Para activar *logging* de arquivo configure o parâmetro db cfg LOGRETAIN para YES.

O *logging* de arquivo é normalmente usado em sistemas de produção, e como os *logs* são mantidos, isso permite que a base de dados recupere para o mais antigo dos ficheiros de *log* na maioria das situações. Com o *logging* de arquivo, um DBA pode recuperar dos erros causados pelos humanos. Por exemplo, se um utilizador do sistema inadvertidamente inicia a execução de uma transacção que dura dias, quando o problema é detectado mais tarde, o DBA pode restaurar o sistema até ao ponto antes do problema ser introduzido. Contudo, será necessário alguma manipulação manual para executar esta tarefa correctamente.

O *logging* de arquivo é necessário para recuperações *roll foward* e cópias de segurança *online.* A Figura 11.3 mostra o processo de *logging* de arquivo.

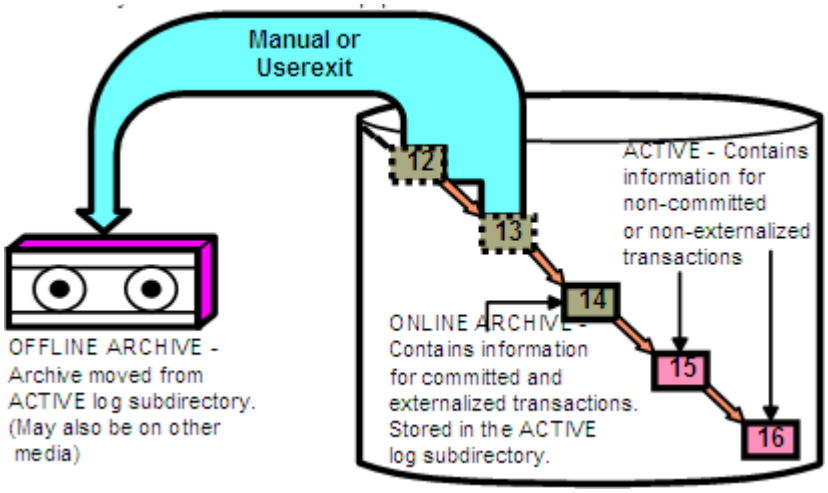


Figura 11.3 – *Logging* de arquivo

11.4 *Logging* da base de dados a partir do *Control Center*

Pode configurar o *logging* da base de dados desde o *Control Center* clicando no botão direito do rato na base de dados em questão, e escolher “Configure Database Logging”. A Figura 11.4 demonstra como fazê-lo.

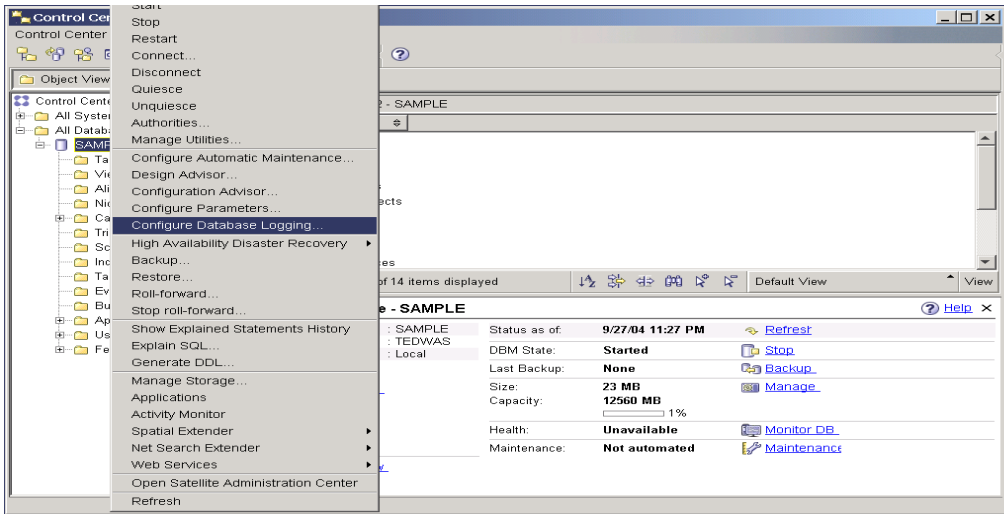


Figura 11.4 – Configurando o *logging* da base de dados desde o *Control Center*.

A Figura 11.5 mostra o Assistente de *Logging* da Base de Dados onde escolhe se deseja *logging* circular ou de arquivo.

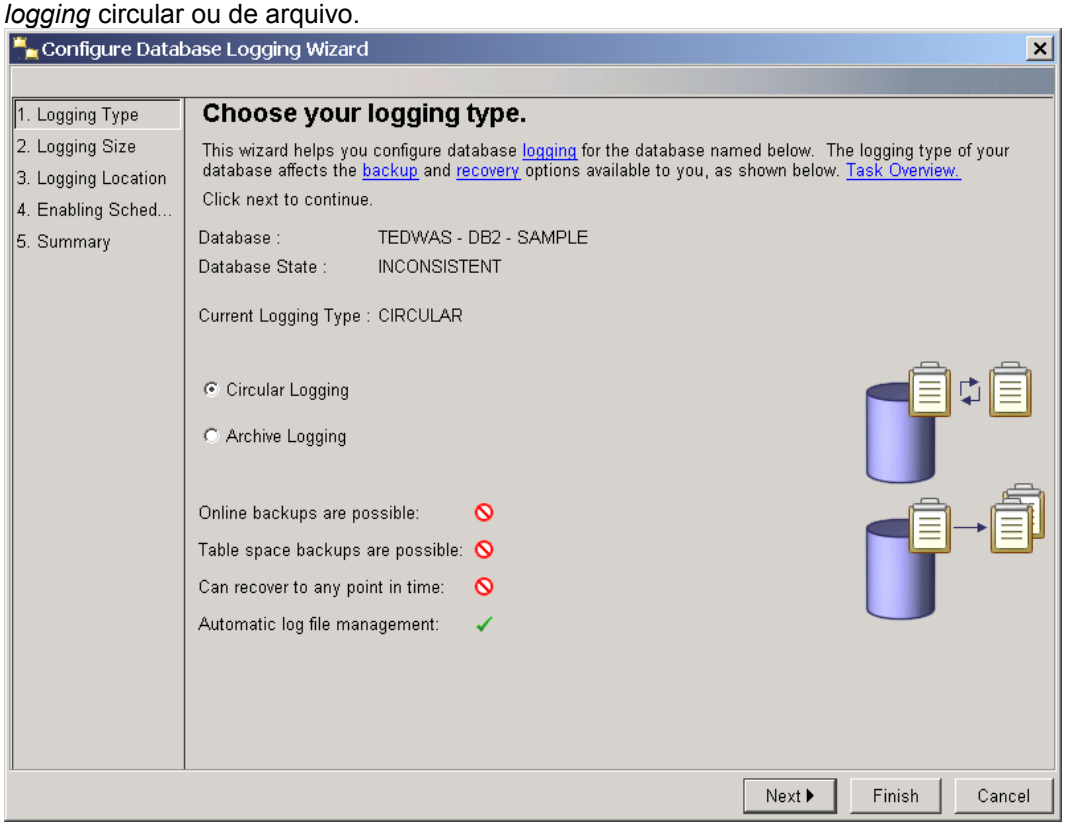


Figura 11.5 – Assistente de *Logging* da Base de Dados

11.5 Parâmetros de *Logging*

Existe um número de parâmetros DB CFG relacionados com o *logging*. A Tabela 11.1 lista os parâmetros principais.

Parâmetro	Descrição
logbufsz	A quantidade de memória para usar como *buffer* para o registo de *logs* antes de escrever estes em disco.
logfilsz	O tamanho de cada *log* de configuração, em números de páginas de 4KB.
logprimary	O número de *logs* primários do tamanho logfilsz que serão criados.
logsecond	O número de *logs* secundários que são criados e usados na recuperação, se necessária.
logpath/newlogpath	A localização em que são localizados os futuros *logs* activos e de arquivo.
mirrorlogpath	Para proteger os *logs* no caminho dos *logs* primários de *crashes* do disco ou de remoção acidental, pode especificar que um idêntico conjunto de *logs* seja mantido num caminho de *log* secundá-

	rio (mirror).
loghead	O nome do *log* que está actualmente activo.
userexit	Activa o programa USEREXIT para que copie os *logs offline.*
softmax	Custos limites de uma recuperação de falha (*crash).*
logretain	Activa o modo de *logging* de arquivo.
overflowlogpath	Similar à opção OVERFLOW LOG PATH do comando ROLLFORWARD; contudo, em vez de especificar a opção OVERFLOW LOG PATH para cada comando ROLLFOWARD pode configurar este parâmetro uma única vez.
blk_log_dsk_ful	Definido para prevenir a geração de erros de disco cheio quando o DB2 nao pode criar um novo ficheiro *log* no caminho de *log* activo. Em vez disso, o DB2 tentará criar um ficheiro de *log* a cada 5 minutos até obter sucesso. Desbloqueado, apenas SQL de leitura poderá continuar.
max_log	Percentagem máxima de espaço do *log* activo para transacção.
num_log_span	Número de ficheiros de *log* activos para 1 UOW activo.

Tabela 11.1 – Parâmetros de *logging*

11.6 Cópias de Segurança da base de dados

O comando BACKUP do DB2 permite guardar uma cópia da instância da base de dados à altura da execução do comando. A sintaxe mais simples para correr este comando é:

```
BACKUP DATABASE <nome_bd> [ TO <caminho> ]
```

A maioria dos comandos e utilitários podem correr *online* ou *offline. Online* implica que outros utilizadores possam estar conectados e a executar operações na base de dados enquanto executa o seu comando. *Offline* significa que mais nenhum utilizador está ligado à base de dados enquanto executa a sua operação. Para permitir uma operação *online*, adicione a palavra ONLINE à sintaxe do comando, senão, por omissão será assumida a opção *offline.*

Por exemplo, se deseja fazer cópias de segurança da base de dados **sample** para o caminho **C:\BACKUPS** pode executar o seguinte comando a partir da linha de comandos do Windows/Linux:

```
db2 BACKUP DB sample TO C:\BACKUPS
```

Note que a directoria C:\BACKUPS deve existir antes de proceder a execução do comando. Assegure também que não existem ligações à base de dados quando executa o comando, senão receberá uma mensagem de erro, uma vez que cópias de segurança *offline* não podem ser executadas quando existem ligações.

Para saber se existem ligações à base de dados numa determinada instância, execute o seguinte comando na linha de comandos Windows/Linux:

```
db2 list applications
```

Para forçar todas as ligações de uma instancia da base de dados, execute o seguinte comando na linha de comandos Windows/Linux:

```
db2 force applications all
```

Não deverá executar este último comando num ambiente de produção com muitos utilizadores, pois poderá receber muitas chamadas de colegas chateados! Note também que esse mesmo comando corre assincronamente. Isto significa que quando tentar correr o comando BACKUP logo a seguir, pode ainda não funcionar. Espere alguns segundos, e repita o comando BACKUP se receber um erro da primeira vez.

Depois de uma execução com sucesso do comando BACKUP, é criado um novo ficheiro contendo a imagem da cópia de segurança da base de dados. O nome do ficheiro segue a convenção abaixo representada na Figura 11.6.

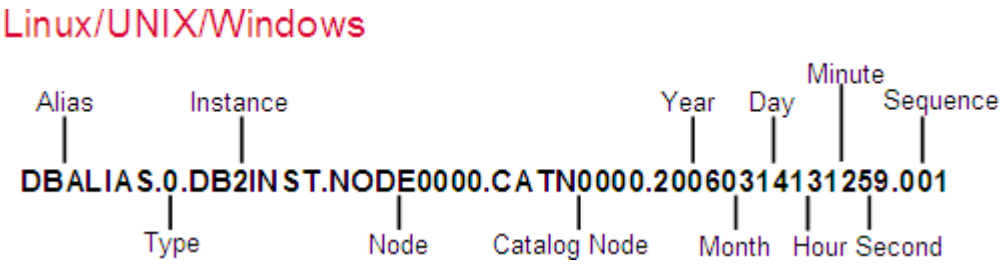


Figura 11.6 – Convenção do nome das imagens das cópias de segurança

Um tipo de “0” indica que se trata de uma cópia de segurança completa. Um tipo de “3”, significaria que é apenas uma cópia de segurança dos dados das tabelas. O *node* é fixado em NODE0000 para bases de dados não particionadas, que é o caso de todas as edições de DB2 excepto a DB2 Enterprise Edition com a característica DPF. O *catalog node* também é fixado como CATN0000. Veja os manuais de DB2 para mais detalhes.

Quando são guardadas muitas cópias de segurança e armazenadas na mesma directoria, o *timestamp* do nome serve para distinguir entre as diversas imagens guardadas. Como veremos na próxima secção, o comando RESTORE pode usar o *timestamp* para recuperar uma cópia de segurança específica.

Quicklab #10 – Agendando uma cópia de segurança

Objectivo

Apesar de DB2 ser capaz de automatizar várias actividades de manutenção da base de dados, às vezes quererá especificar quando determinada actividade ocorre. Neste Quicklab, criará uma tarefa agendada nocturna de cópias de segurança para a base de dados EXPRESS.

Procedimentos

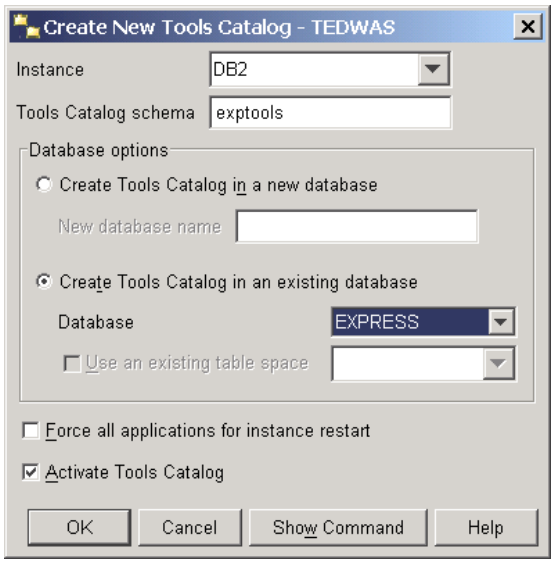
14. Desde a árvore de objecto do *Control Center*, navegue até *Control Center => All Databases*. Clique com o direito em *EXPRESS database* e depois seleccione o item *Backup*. Isto lança o *Backup Wizard* (Assistente de Cópias de Segurança).

15. A página de introdução do Assistente sumaria o estado actual da base de dados incluindo a data da última cópia de segurança e o método de *logging*. Clique em *Next.*

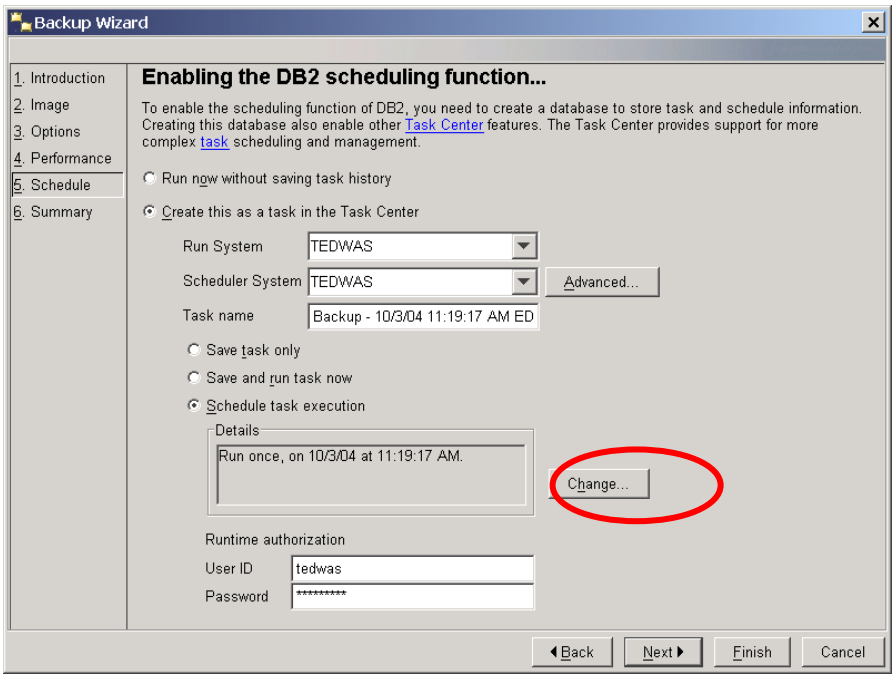
16. Na página *Image* do Assitente, seleccione o destino da imagem da cópia de segurança. Tipicamente irá escolher uma *drive* física diferente da actual *drive* onde está armazenada a base de dados. Por agora, crie uma nova directoria no sistema chamada `C:\db2backup`, e especifique que essa será a directoria de cópia de segurança. No Assitente, seleccione o item *File System* na lista *Media Type*. Clique no botão *Add*, seleccione a directoria que acabou de criar, e de seguida clique em *Ok*. Clique em *Next.*

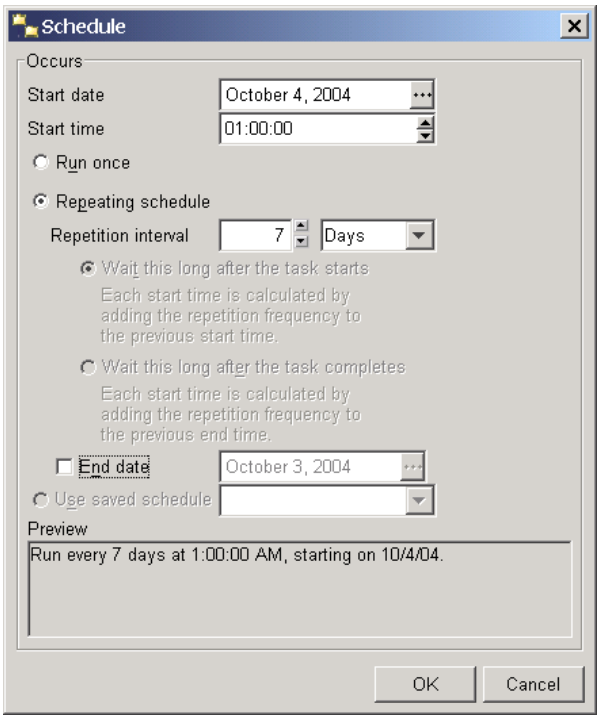
17. Pode explorar as páginas de *Options* e *Performance*, mas as suas opções por defeito são usualmente suficientes porque o DB2 automaticamente executa cópias de segurança optimizadas. Navegue pela página *Schedule* quando terminar a sua exploração.

18. Na página *Schedule*, se ainda não estiver activo o *Scheduler,* active-o agora mesmo. Seleccione o sistema para criar o catálogo de ferramentas *ON* e crie um novo catálogo de ferramentas. Especifique um esquema para o catálogo de ferramentas e escolha para criar-lo na base de dados EXPRESS existente. O catálogo de ferramentas retém metadados sobre todas as tarefas da *Schedule*. Clique *Ok* para continuar. Clique em *Next* para ir para a próxima página do Assistente assim que o catálogo tenha sido criado.



19. Na página de *Schedule*, escolha a opção *schedule for task execution*. Agende a cópia de segurança para correr todos os dias à 01h00. Clique em *Next.*





20. Na página *Summary*, pode rever as tarefas agendadas que serão criadas. Quando tiver revisto todas as tarefas clique em *Finish* para cria a tarefa.

21. Lance o *Task Center* (Centro de Tarefas) para ver ou alterar as recentemente criadas tarefas de cópias de segurança.

11.7 Recuperação da base de dados

Uma recuperação de base de dados implica restaurar a base de dados de uma cópia de segurança e/ou dos *logs*. Se restaurar a partir de uma cópia de segurança estará a restaurar a base de dados tal como existia no momento em que o comando BACKUP foi executado.

Se o *logging* de arquivo estava activo antes da cópia de segurança, pode restaurar não só a partir de uma imagem de cópia de segurança, assim como a partir de ficheiros de *log*. Como veremos na próxima secção, uma recuperação *roll-foward* permie que restaure a cópia de segurança e depois aplique (roll-foward) os *logs* até ao fim dos *logs*, ou até outra data específica.

Note que o termo recuperação é usado frequentemente nesta secção mas o comando usado para recuperação é o comando RESTORE.

11.7.1 Tipos de Recuperação

Existem três tipos de recuperação:

- ***Crash* ou recuperação de reinício**
  Assuma que está a trabalhar num computador, correndo importantes transacções para uma base de dados DB2 na mesma máquina. De repente fica sem corrente eléctrica, ou alguém lhe arranca, acidentalmente, o fio da tomada: que acontecerá à base de dados?

  A próxima vez que arrancar o seu computador, e arrancar o DB2, uma recuperação da falha (*crash*) será automaticamente iniciada. Numa recuperação de um *crash,* o DB2 automaticamente correrá o comando RESTART DATABASE e irá ler e refazer/desfazer as transacções baseando-se nos *logs* activos. Quando este comando terminar, está garantido que a base de dados está num estado consistente, ou seja, o que quer que haja sido submetido será guardado, e o que não haja sido submetido será desfeito.

- **Recuperação de versão ou imagem**
  Este tipo de recuperação implica que está a ser restaurado apenas da imagem de cópia de segurança; logo, a sua base de dados será posta no estado aquando da execução do comando BACKUP. Qualquer transacção efectuada depois da cópia de segurança será perdida.

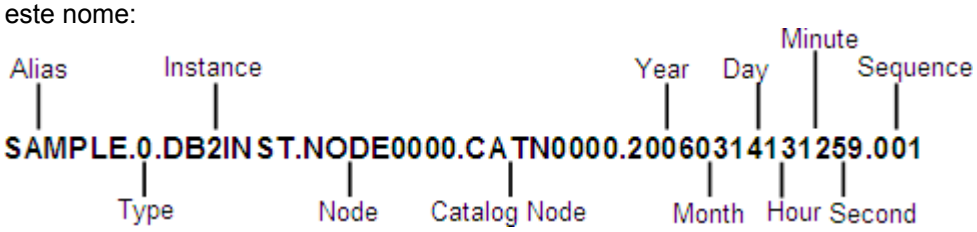
- **Recuperação *Roll-forward***
  Com este tipo de recuperação, não recuperará apenas da imagem de cópia de segurança, assim como também correrá o comando ROLLFOWARD para aplicar os *logs* por cima da cópia de segurança para que assim possa recuperar para uma data específica. Este tipo de recuperação minimiza as perdas de dados.

11.7.2 O comando RESTORE DATABASE

Use o comando RESTORE para recuperar a base de dados a partir de uma cópia de segurança. De seguida mostramos a sintaxe mais simples para executar tal operação:

```
RESTORE DATABASE <nome_db> [from <caminho>] [taken at <timestamp>]
```

Por exmplo, se tem uma imagem de cópia de segurança da base de dados *sample* com este nome:



Poderia executar o seguinte comando:

```
RESTORE DB sample FROM <caminho> TAKEN AT 20060314131259
```

11.8 Outras operações de BACKUP e RESTORE

A seguir listam-se algumas das coisas que se podem fazer também com os comandos BACKUP e RESTORE. Encorajámo-lo para que releia os manuais DB2 para mais detalhes.

- Efectue cópias de segurança a uma instância de 32-bit e recupere-a para uma de 64-bit
- Recupere sobre uma base de dados já existente
- Use uma recuperação redireccionada quando está a restaurar um sistema onde existe um diferente número de discos do que foi especificado na imagem da cópia de segurança.
- Faça cópias de segurança ou recupere apenas da *table space*, em vez de usar a base de dados inteira
- Cópias de segurança incrementais e *delta* são permitidas; cópias *delta* registam apenas alterações da cópia de segurança para a próxima, enquanto que as cópias incrementais registam todas as alterações e acumulam-nas em cada imagem da cópia de segurança
- Copiar de segurança desde uma *flash copy* (requerido hardware correcto)
- Recuperar tabelas que foram apagadas (se a opção estava activa para tabela dada)
- **Cópias de segurança de uma plataforam (ex. Windows) e recuperação para outra plataforma (ex. Linux) não é possível. Use *db2look* e *db2move* para situações como esta.**

12

Capítulo 12 – Tarefas de Manutenção

Este capítulo descreve algumas tarefas requeridas para conseguir uma boa manutenção da base de dados. O objectivo global do DB2 é automatizar a maior parte destas tarefas. A edição DB2 Express-C, como todas as outras edições DB2, incluem estas capacidades automatizadas. Esta capacidade de gestão própria traz grandes benefícios às pequenas e médias empresas que não podem contratar um Administrador de Base de Dados (DBA), a tempo inteiro, para gerir o servidor de dados. Por outro lado, se um DBA for contratado, ele ou ela terão mais tempo disponível para levar a cabo actividades avançadas, que darão mais valor às bases da empresa.

Nota:

Para mais informações sobre tarefas de manutenção, veja este vídeo:
http://www.channeldb2.com/video/video/show?id=807741:Video:4302

12.1 REORG, RUNSTATS, REBIND

Estas são as três principais tarefas de manutenção em DB2, tal como está representado na Figura 12.1: REORG, RUNSTATS e REBIND.

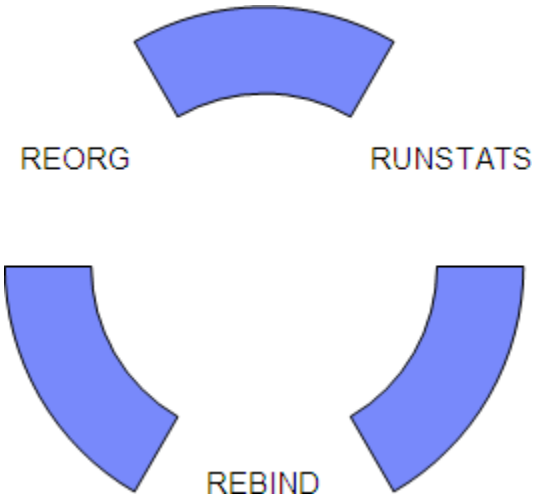


Figura 12.1 – Tarefas de Manutenção: REORG, RUNSTATS, REBIND

A Figura 12.1 mostra que as tarefas de manutenção são executadas num estilo circular. Se uma REORG é executada, é recomendado que também se execute uma RUNSTATS, seguida de uma REBIND. Após algum tempo, as tabelas numa base de dados serão modificadas devido a operações de UPDATE, DELET e INSERT. Nessa altura o ciclo começará de novo com um REORG.

12.1.1 O comando REORG

Com o tempo, à medida que se vai executando operações de INSERT, UPDATE e DELETE na sua base de dados, os seus dados começam a ficar cada vez mais fragmentados através das páginas da base de dados. O comando REORG recupera espaço desperdiçado e reorganiza os dados para tornar as procuras mais eficientes. As tabelas que são frequentemente modificadas vão beneficiar do REORG. Pode reorganizar (REORG) os índices, tal como as tabelas, e pode usar o REORG quer *online* quer *offline*.

O REORG *offline* é mais rápido e mais eficiente, mas não permite o acesso à tabela, enquanto que o REORG *online* permite o acesso à tabela, mas consome muitos recursos do sistema; isto resulta melhor para tabelas pequenas.

Sintaxe:

```
REORG TABLE <tablename>
```

Exemplo:

```
REORG TABLE employee
```

O comando REORGCHK pode ser usado antes do REORG para determinar se uma tabela ou um índice necessitam ser reorganizados.

12.1.2 O comando RUNSTATS

O *DB2 Optimizer* é "o cérebro" do DB2. Ele procura os caminhos de acesso mais eficientes para localizar e devolver os dados. O *optimizer* é "*system cost-aware*", e utiliza estatísticas da base de dados que são armazenadas em tabelas de catálogo, para maximizar a performance da base de dados. Por exemplo, as tabelas de catálogo têm estatísticas sobre quantas colunas, ou quantas linhas, estão presentes numa tabela, quantos e que tipos de índices estão disponíveis para determinada tabela, etc.

Estas estatísticas não são actualizadas dinamicamente. Isso não acontece, pois não desejaria que o DB2 actualizasse as estatísticas a cada operação efectuada na base de dados; afectaria negativamente a performance da base de dados. Ao contrário, o DB2 disponibiliza o comando RUNSTATS para actualizar estas estatísticas. É essencial para manter a base de dados actualizada. O *DB2 optimizer* pode fazer mudanças radicais no caminho de acesso se pensar que uma tabela tem 1 linha versus 1 milhão de linhas. Quando as estatísticas da base de dados estão actualizadas, o DB2 escolhe o melhor plano de acesso aos dados. A frequência da captura de estatísticas deve ser determinada pela frequência com que os dados se alteram na tabela.

Sintaxe:

```
RUNSTATS ON TABLE <schema.table>
```

Exemplo:

```
RUNSTATS ON TABLE myschema.employee
```

12.1.3 BIND / REBIND

Após executar com sucesso o comando RUNSTATS, nem todas consultas irão utilizar as últimas estatísticas. Os planos de acesso de SQL estático são determinados quando executar um comando BIND, e as estatísticas usadas na altura podem não ser as mesmas que as actuais. A Figura 12.2 ajuda a esclarecer esta ideia.

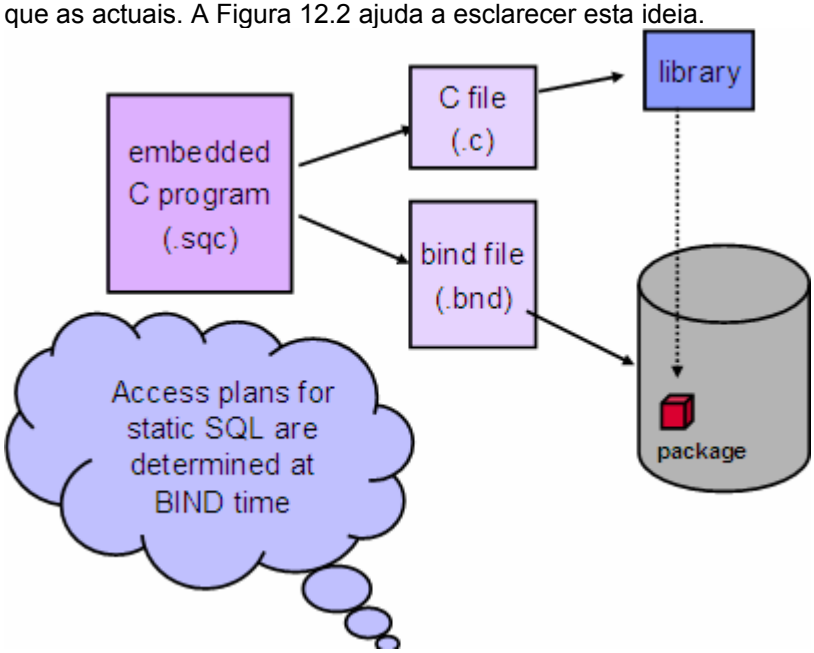


Figura 12.2 – Processo de *bind* de SQL estático

Na Figura 12.2 um programa em *embedded* C (armazenado com a extensão "sqc") é pré-compilado. Após a pré-compilação, dois ficheiros são gerados, um ficheiro ".c" contendo o código C com todo o SQL comentado; e um ficheiro ".bnd" contendo toda as *queries* SQL. O ficheiro com a extensão ".c" é compilado, normalmente com um compilador C, criando uma "biblioteca" como mostra a figura no canto superior direito. O ficheiro ".bnd" gera um *package* que é armazenado na base de dados. *Binding* é equivalente a compilar consultas SQL onde o melhor plano de acesso é determinado com base em estatísticas. O comando *db2rbind* pode ser usado para REBIND todas as *packages* existentes usem as últimas estatísticas.

Sintaxe:

```
db2rbind database_alias -l <ficheirolog>
```

Exemplo:
Para fazer o REBIND de todas as packages da base de dados *sample* e guardar o *log* de saída no ficheirolog.txt utiliza-se este comando:

```
db2rbind sample -l ficheirolog.txt
```

12.1.4 Tarefas de Manutenção do *Control Center*

Desde o *Control Center* pode-se reorganizar (REORG) e correr estatísticas (RUNSTATS). (Figura 12.3)

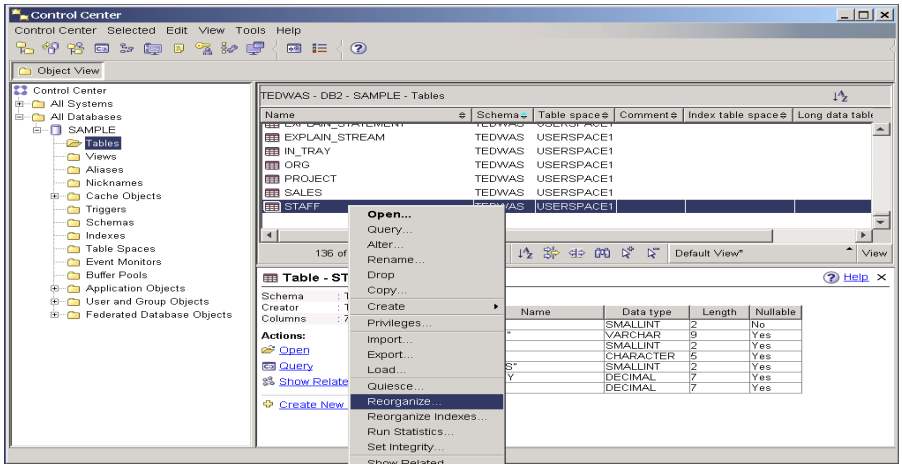


Figura 12.3 – REORG e RUNSTATS desde o *Control Center*

Escolheu a tabela onde quer operar, clique com o botão direito e escolha *Reorganize* (REORG) ou *Run Statistics* (RUNSTATS).

A vista operacional da base de dados

Quando selecciona a base de dados, a vista operacional da base de dados no canto inferior direito do *Control Center* providenciará informação sobre a base de dados como por exemplo, o tamanho, quando foi feita a última cópida de segurança (*Backup*), etc. Esta vista permite-lhe identificar rapidamente requisitos de manutenção para a sua base de dados. A Figura 12.4 ajudará.

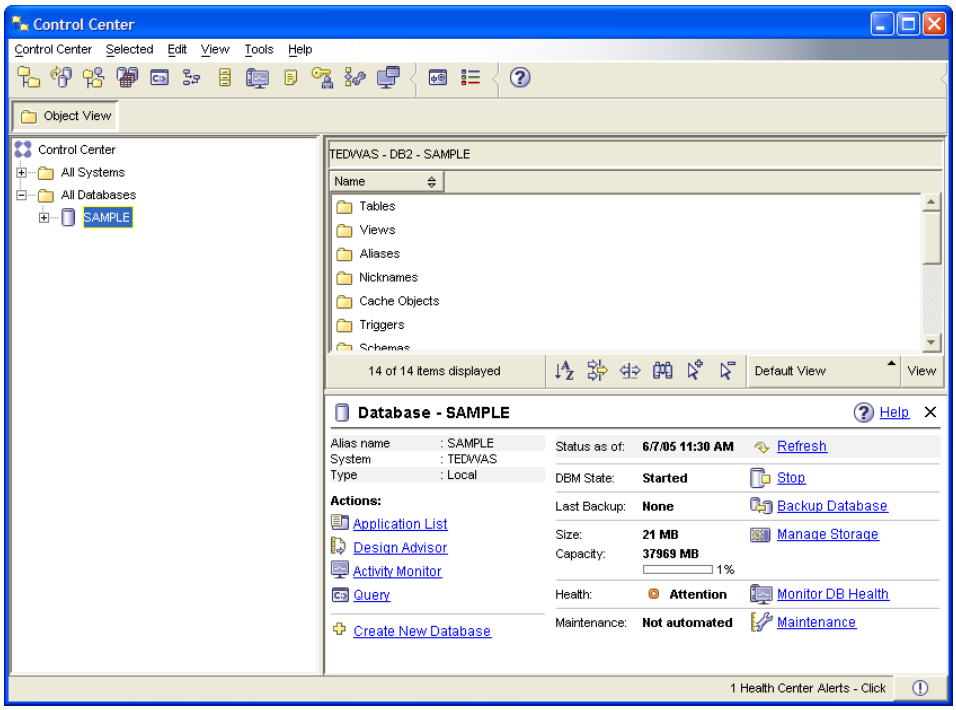


Figura 12.4 – A vista operacional da base de dados desde o *Control Center*

12.2 Opções de Manutenção

Existem três maneira de executar as tarefas de manutenção:

1. Manutenção Manual
   As actividades de manutenção são executadas manualmente pelo administrador, consoante as necessidades surgem.

2. Criar *scripts* para executar a manutenção
   Pode criar *scripts* com comandos de manutenção, e agendá-los para execução.

3. Manutenção Automática
   O DB2 automaticamente toma conta das tarefas de manutenção por si (REORG, RUNSTATS, BACKUP)

Nesta secção vamos concentrar-nos na manutenção automática.

A manutenção automática consiste em:

- O utilizador define uma *maintenance window* onde as tarefas podem ser executadas com mínimas rupturas. Por exemplo, se o sistema tem a pouca actividade agendada para Domingo das 02:00am às 04:00am, este intervalo de tempo servirá como *maintenance window*.
- Existem duas *maintenance Windows*: uma para operações *online*, e outra para operações *offline*.

- O DB2 executará automaticamente as operações de manutenção, só quando necessário durante a *maintenance window.*

Desde o *Control Center*, poderá lançar o Assistente de Configuração da Manutenção Automática, como mostra a Figura 12.5.

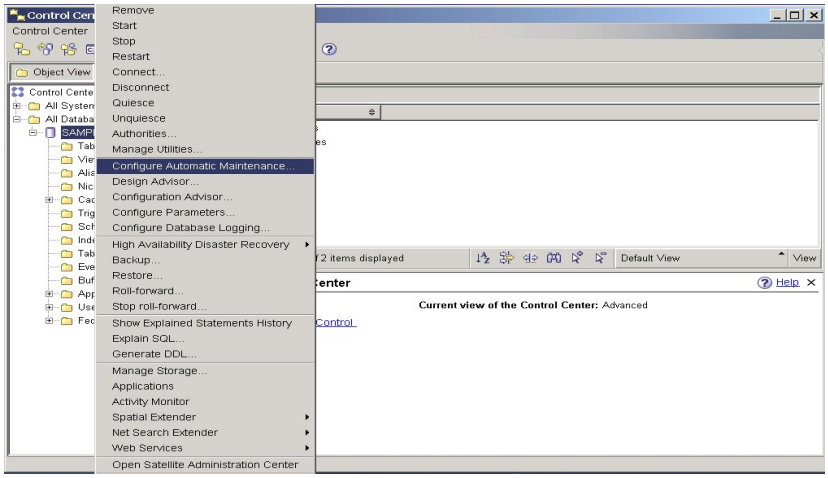


Figura 12.5 –Lançando o Assistente de Configuração da Manutenção Automática

A Figura 12.6 mostra o Assistente de Configuração da Manutenção Automática.

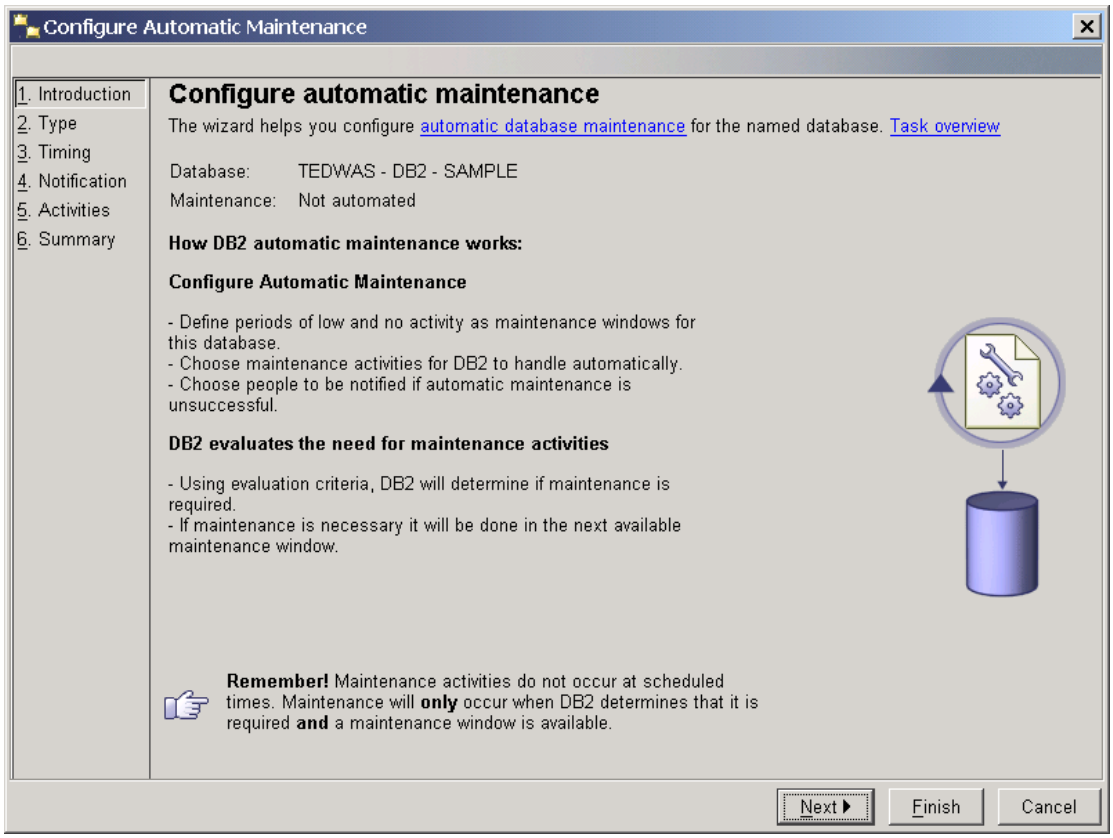


Figura 12.6 – O Assistente de Configuração da Manutenção Automática.

Quicklab #11 – Configurando a Manutenção Automática

Objectivo

Neste Quicklab, em poucos passos, irá configurar manutenção automática na base de dados DB2 SAMPLE.

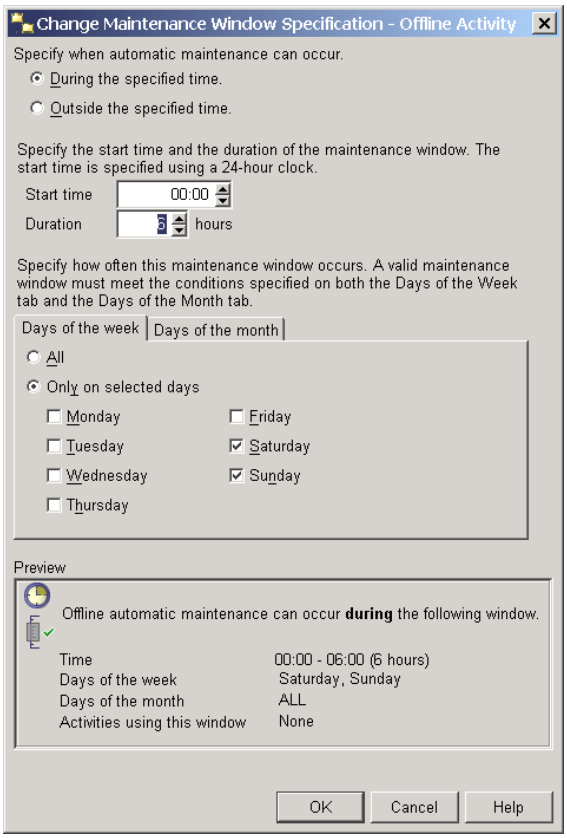
Procedimento

- Desde a árvore de objectos do *Control Center*, clique com o botão direito na base de dados SAMPLE e seleccione o item *Configure Automatic Maintenance* do menu. Isto lança o Assistente de Configuração da Manutenção Automática.

- A página *Introduction* do Assistente (*wizard*) mostra as configurações actuais da manutenção automática. Se criou a base deados com a opção de manutenção automática, então a esta já se encontra configurada. Pode utilizar o Assistente para reconfigurar opções da manutenção automática. Clique no botão *Next* para mover para a próxima página do Assistente.

- A página *Type* do Assistente pede-lhe para escolher entre desactivar a manutenção automatic, ou alterar as suas configurações da manutenção automática. Seleccione a opção para alterar a configuração actual. Clique *Next*.

- A página *Timing* do Assistente pede-lhe para especificar a *maintenance window*. Configure a *Offline window* para todos os Sábados e Domingos à noite da meia-noite até às 06h00am como mostra a figura. Clique no botão *Change* além do painel de pré-visualização da *offline maintenance window*-Após especificar as informações requeridas, clique Ok para voltar ao Assistente. Mantenha a *online window* como está. Clique *Next*.



- Na página *Notification* do Assistente, pode definer um contacto para o caso da manutenção automática falhar. Avance este passo por agora. Clique *Next*.

- Na página *Activities* do Assistente, pode escolher automatização individual ou não automatizar certas actividades, assim como escolher ser notificado de certas actividades. Neste exemplo, assegure que todas as *Automate checkboxes* estão *checked* e as *Notify checkboxes* and the *Notify checkboxes* estão *unchecked*. Clique *Next*.

- Antes de avançar para a próxima página do Assistente, deve configurar a localização da cópia de segurança da base de dados. Idealmente, quer armazenar as cópias de segurança em unidades físicas diferentes para o caso de haver uma falha no disco. Na página *Activities*, seleccione a opção *Backup database*, e em seguida clique no botão *Configure Settings*.

- No separador *Backup Criteria* da janela de diálogo *Configure Settings*, escolha a opção *Balance Database Recoverability with Performance*. No separador *Backup Location*, seleccione a localização da cópia de segurança existente e clique no botão *Change*. Especifique uma localização diferentes para executar a cópia de segurança (assegure que existe espaço suficiente em disco). No separador *Backup Mode*, assegure que está seleccionado o *Offline Backup*. Clique *OK* para fechar o separador *Backup Criteria*. Clique *Next*.

- A página *Summary* do Assistente de Configuração da Manutenção Automática contém o sumário das opções que seleccionou. Clique *Finish* para aceitar e implementar as alterações.

13

Capítulo 13 - Concorrência e *Locking*

Neste capítulo discute-se como permitir que múltiplos utilizadores tenham acesso à mesma base de dados ao mesmo tempo sem interferirem uns com os outros, mantendo as suas operações consistentes. Será discutido os conceitos de transacções, concorrência e *locking*.

> **Nota**:
>
> Para mais informações acerca de concorrência e *locking*, veja este video:
> http://www.channeldb2.com/video/video/show?id=807741:Video:4322

13.1 Transacções

Uma transacção ou unidade de trabalho consiste em um ou mais comandos SQL que, quando executados, devem ser considerados como um único comando; isto é, se um dos comandos falhar, toda a transacção falha, e todas as indicações executadas até o momento da falha são desfeitas. Uma transacção termina com o comando COMMIT, que também significa o começo de uma nova transacção. Na Figura 13.1 pode-se ver um exemplo de uma transacção.

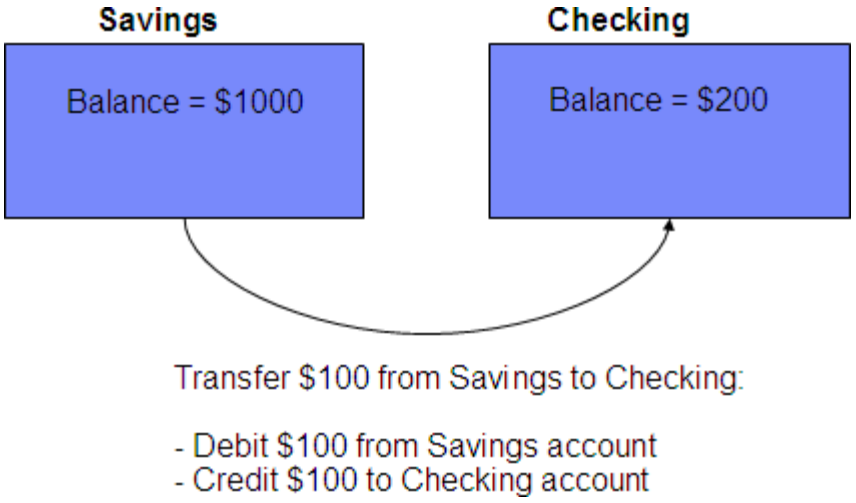


Figura 13.1 – Exemplo de uma transacção

Na Figura 13.1, por exemplo, pretende-se transferir 100 dólares da conta *Savings* para a conta *Checking.* A seguinte sequência de eventos é necessária para realizar esta tarefa:

Debita-se $100 da conta *Savings*
Credita-se $100 para a conta *Checking*

Se a sequência de eventos acima descrita não for tratada como uma única unidade de trabalho, transacção, imagine o que aconteceria se uma falha eléctrica ocorresse após o débito da conta *Savings* e imediatamente antes de se creditar $100 na conta *Checking.* O titular da conta *Savings* perderia $100!

13.2 Concorrência

A concorrência implica que diversos utilizadores possam trabalhar ao mesmo tempo nos mesmos objectos da base de dados. O DB2 foi projectado como uma base de dados multi-utilizador. O acesso aos dados deve ser coordenado correctamente e transparente usando um mecanismo para assegurar a integridade e a consistência dos dados. Considere a Figura 13.2 como um exemplo.

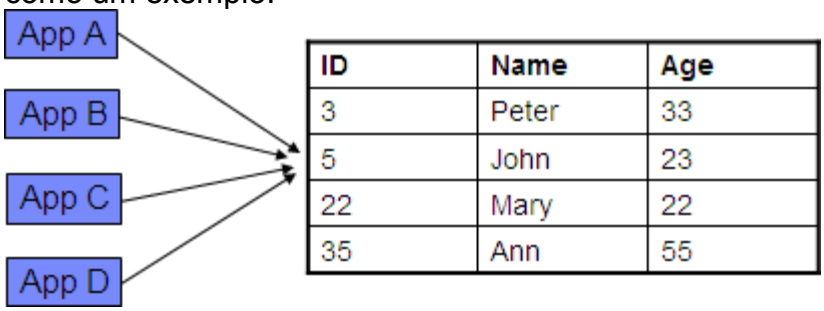


Figura 13.2 – Exemplo de concorrência, e da necessidade de um controlo de concorrência.

Na Figura 13.2, há quatro aplicações, App A, App B, App C, e App D que estão a tentar aceder à mesma linha (linha 2) de uma tabela. Sem nenhum controlo de concorrência, todas as aplicações podiam executar operações sobre a mesma linha. Supondo que todas as aplicações estão a actualizar a coluna “Age” na linha 2 com valores diferentes, a aplicação que faz a última actualização será provavelmente a vencedora nesta situação. Deve ser óbvio neste exemplo que a utilização de algum controlo de concorrência é necessário para garantir resultados consistentes. Este controlo de concorrência é baseado na utilização de *locks*.

Os conceitos de *locking* e concorrência andam em conjunto. Fazer *lock* pára temporariamente a execução de outras aplicações até que uma outra aplicação termine. Quantos mais *locks* um sistema tenha, menos concorrência é possível. Por outro lado, quantos menos *locks* um sistema tenha, mais concorrência será possível.

Os *locks* são adquiridos automaticamente conforme seja necessário para executar uma transacção e são libertados quando a transacção termina (usando o comando COMMIT ou o comando ROLLBACK). Os *locks* podem ser adquiridos em tabelas ou em linhas. Há dois tipos básicos de *locks*:

- *Locks* partilhados (S *locks*) - adquiridos quando uma aplicação quer ler e impedir que outras actualizem a mesma linha
- *Locks* exclusivos (X *locks*) - adquiridos quando uma aplicação actualiza, insere, ou elimina uma linha

Considere agora Figura 13.3, que é idêntica à Figura 13.2, mas agora com um *lock*.

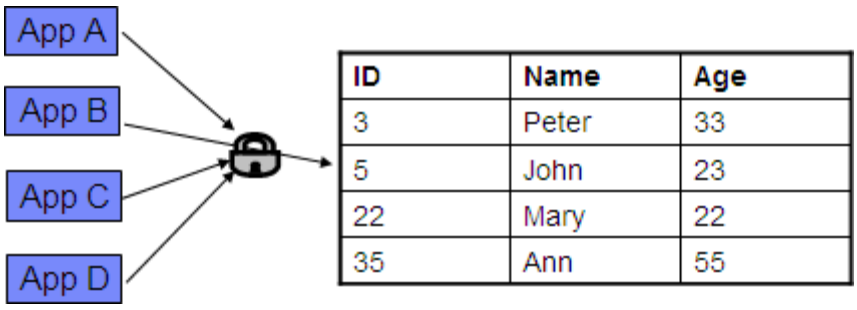


Figura 13.3 – Um exemplo de concorrência e a necessidade do uso de *locks*.

Por exemplo, na figura 13.2, se o App B for a primeira a aceder à linha 2, e está a executar um UPDATE, a App B é detentora do X *lock* na linha. Quando a App A, App C e a App D tentarem aceder à mesma linha, não poderão fazer UPDATE por causa do *lock* exclusivo. Este tipo de controlo permite a consistência e a integridade dos dados.

13.3 Problemas sem controlo de concorrência

Sem alguma forma de controlo de concorrência, podem acontecer os seguintes problemas

1. Lost Update
2. Uncommitted Read
3. Non-repeatable Read
4. Phantom Read

13.3.1 *Lost Update*

A perca de uma actualização é um problema semelhante ao que foi anteriormente explicado nesta secção. A aplicação que executa a última actualização, será a vencedora e todas as alterações anteriores serão perdidas.

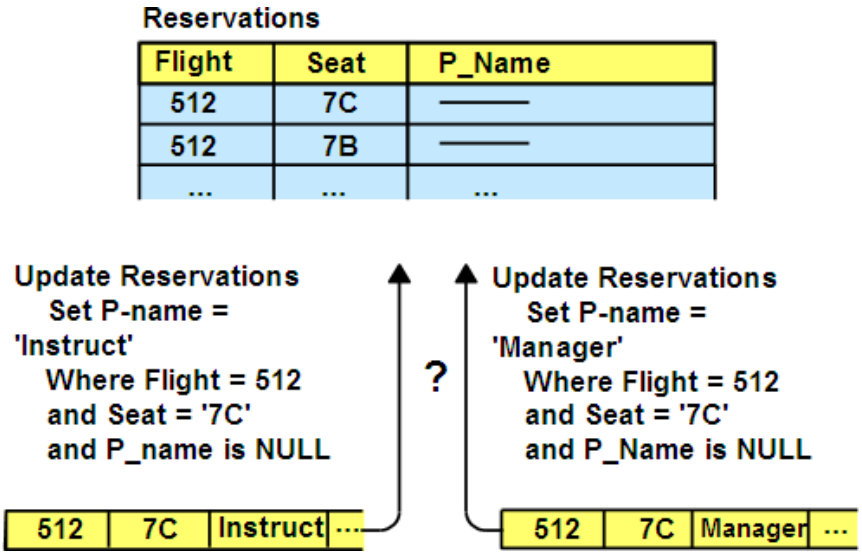


Figura 13.4 – *Lost Update*

Na Figura 13.4 existem dois pedidos a tentar actualizar a mesma linha. O que está à esquerda é o pedido App1, e o da direita é pedido App2. A sequência de eventos é a seguinte:

- App1 actualiza a linha
- App2 actualiza a mesma linha
- App1 faz COMMIT
- App2 faz COMMIT

A actualização do App1 é perdido quando App2 fazer a sua actualização, daí o termo "Lost Update" (perca de actualização).

13.3.2 *Uncommitted read*

Uma *Uncommitted Read*, ou leitura suja permite que uma aplicação leia informações que não tenham sido guardadas e, consequentemente não são precisas.

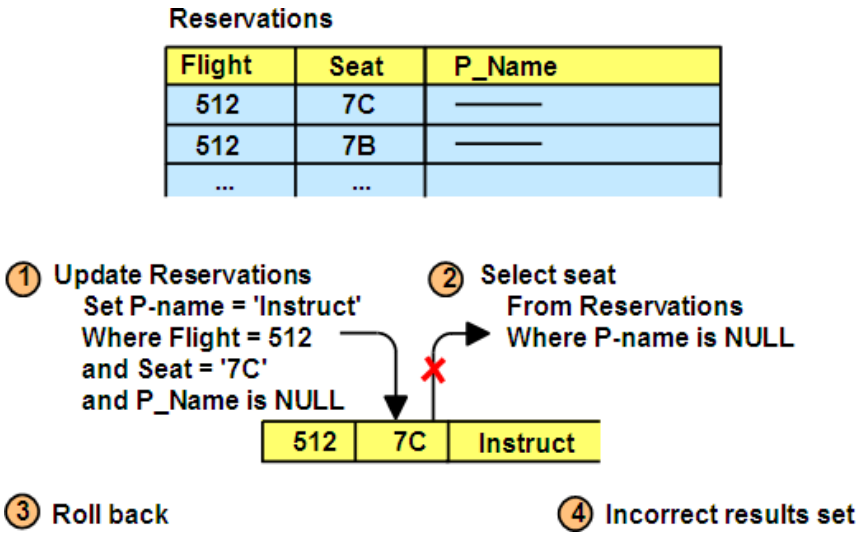


Figura 13.5 – *Uncommitted Read*

A Figura 13.5 segue esta sequência de eventos:

1. App1 actualiza uma linha
2. App2 lê o novo valor da nova linha
3. App1 volta atrás nas mudanças que fez na linha

App2 está a ler dados que ainda não foram gravados, logo são dados inválidos, e é por isso que este problema é chamado de "uncommitted read" (leitura de dados ainda não gravados).

13.3.3 *Non-repeatable read*

A leitura não-repetitiva implica que não se pode obter o mesmo resultado após executar a mesma leitura na mesma operação.

FLIGHT	SEAT	NAME	DESTINATION	ORIGIN
512	7B	____	DENVER	DALLAS
....				
....				
814	8A	____	SAN JOSE	DENVER
....				
134	1C	____	HONOLULU	SAN JOSE
....				....

Figura 13.6 – *Non-repeatable Read*

Na Figura 13.6, considere-se que alguém está a tentar reservar um voo de Dallas para Honolulu. A sequência de eventos é:

1. App1 abre um cursor (*result set)* obtendo o que se pode ver na figura 13.6
2. App2 elimina uma linha do *result set* (por exemplo, a linha com o destino "San Jose")
3. App2 faz COMMIT
4. App1 fecha e reabre o *result set*

Neste caso, uma vez que App1 não iria obter os mesmos dados numa segunda leitura, não pode reproduzir os dados; é por isso que este problema é chamado de "*non-repeatable read*".

13.3.4 *Phantom read*

O problema da leitura fantasma é semelhante ao *non-repeatable read*, a diferença é que pesquisas posteriores, pode-se obter mais linhas, em vez de menos linhas. A Figura 13,7 mostra um exemplo deste problema.

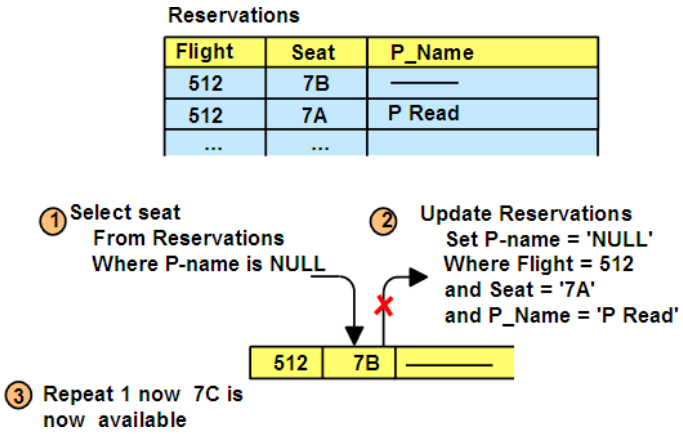


Figura 13.7 – *Phantom read*

Figura 13.7 mostra a seguinte sequência de eventos:

1. App1 abre um *result set*
2. App2 adiciona uma linha à base de dados que apareceria no mesmo *reult set*
3. App2 faz COMMIT
4. App1 fecha e reabre o *result set*

Neste caso, a App1 não iria obter os mesmos dados de uma segunda leitura, iria obter mais linhas, por isso é que este problema é chamado de "phantom read" (leitura fantasma).

13.4 Níveis de isolamento

Pode-se considerar os níveis de isolamento como protocolos fechados onde, em função do nível de isolamento escolhido, uma aplicação pode obter diferentes comportamentos para o *lock* de base de dados.

O DB2 fornece diferentes níveis de protecção para isolar os dados:

1. Uncommitted Read (UR)
2. Cursor Stability (CS)
3. Read Stability (RS)
4. Repeatable Read (RR)

13.4.1 *Uncommitted read*

A protecção *uncommitted read* é também conhecida como leitura suja (*dirty read*). É o nível mais baixo de isolamento, e oferece maior grau de concorrência. Nenhum *lock* de linha é obtido para operações de leitura, a não ser

que outra aplicação tente eliminar ou alterar uma tabela; e as operações de actualização agem como se estivessem a usar o nível de isolamento *cursor stability*.

Problemas que ainda são possíveis de acontecer com este nível de isolamento:

- Uncommitted read
- Non-repeatable read
- Phantom read

Problemas evitados com este nível de isolamento:

- Loss of update

13.4.2 *Cursor stability*

Cursor stability é o nível de isolamento por omissão. Oferece um grau mínimo de *locking*. Basicamente, com este nível de isolamento, uma aplicação que esteja a trabalhar numa linha de um *result set* adquire o *lock*. Se a linha é só para ler, o *lock* é mantido até que uma nova linha seja obtida ou a transacção terminada. Se a linha é actualizada, o *lock* é mantido até a transacção terminar.

Problemas que ainda são possíveis de acontecer com este nível de isolamento:

- Non-repeatable read
- Phantom read

Problemas evitados com este nível de isolamento:

- Loss of update
- Uncommitted read

13.4.3 *Read stability*

Com *Read stability*, todas as linhas que uma aplicação pede durante uma transacção adquirem *lock*. Para um dado cursor (*result set*), é adquirido o lock para todas as linhas que constituem o *result set*. Por exemplo, se se tem uma tabela com 10000 filas e a *query* devolve 10 linhas, só estas 10 linhas estão sob o *lock*. A *Read stability* usa um método moderado de *locking*.

Problemas que ainda são possíveis de acontecer com este nível de isolamento:

- Phantom read

Problemas evitados com este nível de isolamento:

- Loss of update
- Uncommitted read
- Non-repeatable read

13.4.4 *Repeatable read*

Repeatable read é o maior nível de isolamento. Este oferece o maior grau de *locking* e menor concorrência. Os *locks* são mantidos em todas as linhas processadas para construir o *result set* ; isto é, as linhas que poderão aparecer no *result set* final podem ter adquirido *lock*. Até a transacção estar completa nenhuma outra aplicação pode actualizar, eliminar ou inserir uma linha que possa afectar o *result set*.
O *Repeatable read* garante que uma consulta efectuada por uma aplicação mais de uma vez numa transacção dará o mesmo resultado.

Problemas que ainda são possíveis de acontecer com este nível de isolamento:

- nenhum

Problemas evitados com este nível de isolamento:

- Loss of update
- Uncommitted read
- Non-repeatable read
- Phantom read

13.4.5 Comparação de níveis de isolamento

A Figura 13.8 compara os diferentes níveis de isolamento para a recolha de informação. Na figura, vemos que o nível isolamento *uncommitted read* (UR) não adquire nenhum *lock*. O nível de isolamento *cursor stability* (CS) tem adquirido um *lock* para a linha 1 quando está a extrair informação desta, mas liberta-o logo que extrai a linha 2, e assim por diante. Para o isolamento *read stability* (RS) ou o *repeatable read* (RR), qualquer linha que for alvo de extracção de informação irá adquirir *lock*, e o *lock* só é libertado no fim da transacção (momento em que se faz COMMIT).

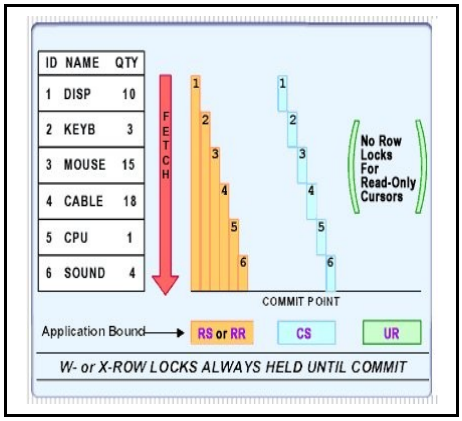


Figura 13.8 – Comparação do nível de isolamento para uma pesquisa.

13.4.6 Seleccionar nível de isolamento

Níveis de isolamento podem ser especificados a muitos níveis:

1. Sessão (aplicação)
2. Conexão
3. Declaração (statement)

O nível de isolamento é normalmente definido durante a sessão ou ao nível da aplicação. Se não for definido nenhum nível de isolamento na aplicação, será usado o nível de isolamento padrão, *cursor stability*. Por exemplo, a Tabela 13.1 mostra os possíveis níveis de isolamento para um programa .NET ou JDBC. E como estas propriedades, quando usadas, correspondem a um nível de isolamento DB2.

DB2	.NET	JDBC
Uncommitted Read (UR)	ReadUncommit-ted	TRANSACTION_READ_UNCOMMITTED
Cursor Stability (CS)	ReadCommitted	TRANSACTION_READ_COMMITTED
Read Stability (RS)	RepeatableRead	TRANSACTION_REPEATABLE_READ

Repeatable Read (RR)	Serializable	TRANSACTION_SERIALIZABLE

Tabela 13.1 - Comparação da terminologia para os níveis de isolamento

Uma declaração para especificar o nível de isolamento pode ser definida usando a clausula WITH {nível de isolamento}. Por exemplo:

```
SELECT ... WITH {UR | CS | RS | RR}
```

Cenário exemplo:
Um pedido deve obter uma contagem "grosseira" de quantas linhas estão numa tabela. O desempenho é de extrema importância. O nível de isolamento *cursor stability* é necessário, excepto para uma declaração SQL:

```
SELECT COUNT(*) FROM tab1 WITH UR
```

Para SQL embebido, o nível de isolamento é decidido em tempo de interpretação, para SQL dinâmico, o nível é fixado em tempo de execução.

A escolha do nível de isolamento a usar depende da aplicação. Se a aplicação não precisa de contagens exactas como no exemplo acima, então deve ser escolhido o isolamento UR. Se a aplicação requer maior controlo sobre os dados com que trabalha, deve ser escolhido o isolamento RR.

13.5 *Lock escalation*

Cada *lock* feito pelo DB2 consome alguma memória. Quando o optimizador pensa que é melhor ter um *lock* em toda a tabela, em vez de múltiplos *locks* de linhas fechaduras, ocorre a escalonação do *lock*.
A Figura 13.9 ilustra isso.

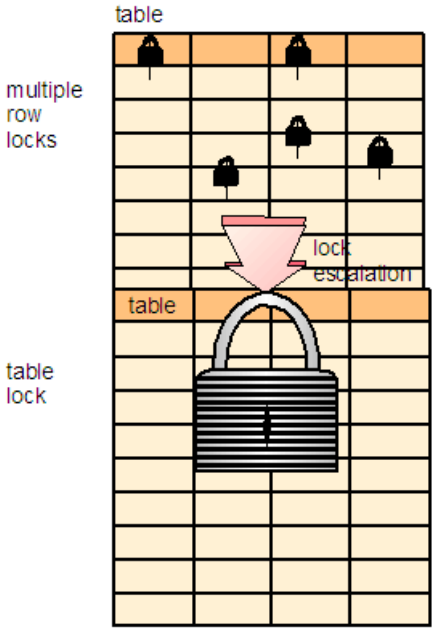


Figura 13.9 – Escalonamento de um *lock*

Existem dois parâmetros principais de configurações de bases de dados relacionados com o escalonamento de *locks*:

1. **LOCKLIST** – A quantidade de memória (em páginas 4k) que é reservada para gerir *locks* para todas as aplicações ligadas. O padrão é cinquenta páginas de 4K (200 K) no Windows

2. **MAXLOCKS** – Percentagem máxima de toda a lista de *locks* que uma única aplicação pode usar. O padrão é de 22%.

Portanto, se os valores padrão são utilizados, a escalonação de *locks* ocorre quando uma aplicação necessita mais de 44K de memória para *locks* (200 K * 22% = 44K). Se, com essas configurações, o escalonamento de *locks* ocorrer frequentemente, deve-se aumentar o valor de LOCKLIST e MAXLOCKS. O escalonamento de *locks* não é bom para o desempenho, uma vez que reduz concorrência. O ficheiro de diagnóstico do DB2 (db2diag.log, que normalmente está localizado no directório C: \ Program Files \ IBM \ SQLLIB \ DB2) pode ser usado para determinar se um escalonamento de *locks* está a ocorrer.

13.6 Monitorização de locks

Pode-se acompanhar o uso de *locks* utilizando a aplicação de *DB2 Lock Snapshot*. Para ligar o *Lock Snapshot*, executa-se este comando:

```
UPDATE MONITOR SWITCHES USING LOCK ON
```

Depois de estar ligado (ON), as informações de monitorização serão colectadas. Para obter um relatório dos *locks* num determinado momento, executa-se este comando:

```
GET SNAPSHOT FOR LOCKS FOR APPLICATION AGENT ID <handle>
```

A Figura 13.10 mostra o relatório de *locks* de uma aplicação.

```
Application Lock Snapshot

Snapshot timestamp = 11-05-2002 00:09:08.672586

Application handle = 9
Application ID = *LOCAL.DB2.00B9C5050843
Sequence number = 0001
Application name = db2bp.exe
Authorization ID = ADMINISTRATOR
Application status = UOW Waiting
Status change time = Not Collected
Application code page = 1252
Locks held = 4
Total wait time (ms) = 0

List Of Locks
Lock Name = 0x05000700048001000000000052
Lock Attributes = 0x00000000
Release Flags = 0x40000000
Lock Count = 255
Hold Count = 0
Lock Object Name = 98308
Object Type = Row
Tablespace Name = TEST4K
Table Schema = ADMINISTRATOR
Table Name = T2
```

Figura 13.10 – Relatório de Locks para uma Aplicação

13.7 Espera de *lock*

Quando duas ou mais aplicações precisam de efectuar uma operação sobre o mesmo objecto, uma delas pode ter de esperar para obter o *lock* necessário. Por omissão, um pedido vai esperar indefinidamente. O tempo que um pedido aguarda para obter *lock* é controlado pelo parâmetro de configuração LOCKTIMEOUT. O valor por omissão deste parâmetro é -1 (espera infinita).

O registo CURRENT LOCK TIMEOUT pode ser usado para definir a espera pelo *lock* de um determinada conexão. Por defeito, este registo é definido com o valor de LOCKTIMEOUT. Usa-se a declaração SET LOCK TIMEOUT para alterar o seu valor. Assim que o valor deste registo é definido por uma ligação, ele irá ser sempre o mesmo em toda a transacção.
Exemplo:

```
SET LOCK TIMEOUT=WAIT n
```

13.8 Causas e detecção de *deadlocks*

Um *deadlock* ocorre quando duas ou mais aplicações ligadas à mesma base de dados aguardam indefinidamente por um recurso. A espera nunca é terminada, porque cada pedido é possuidor de um recurso que o outro precisa. Os d*eadlocks* são, a maior parte das vezes, um problema de design da aplicação. A Figura 13.11 ilustra um *deadlock*.

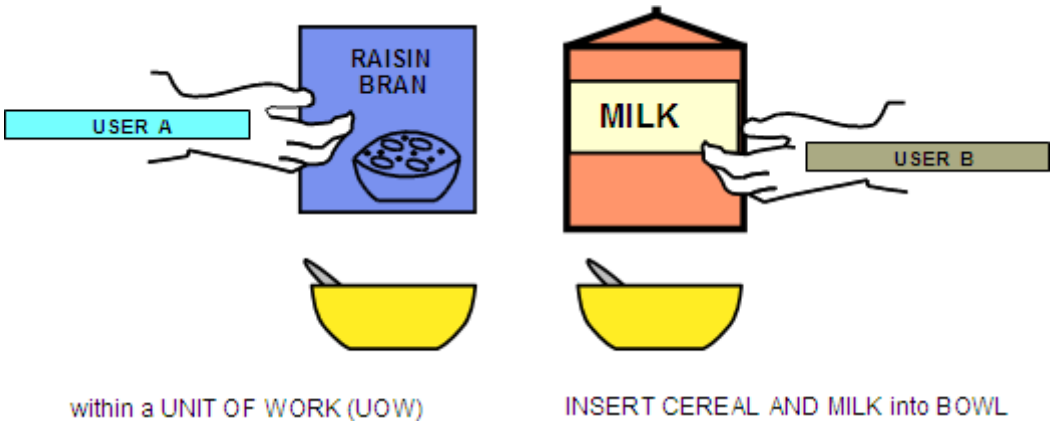


Figura 13.11 – Cenário de um *deadlock*

Na Figura 13.11, o utilizador A está a segurar os cereais e não vai largar até receber o leite. Por outro lado, o utilizador B está a segurar o leite, e não vai largar até receber os cereais. Portanto, temos uma situação de *deadlock*.

Para simular uma situação de *deadlock* no DB2, siga estes passos:

1. Abrir duas consolas DB2 (que vamos chamar "CLP1" e "CLP2". Respectivamente), que representam duas conexões diferentes à base de dados
2. Através de CLP1 corra os comandos:

```
db2 connect to sample
db2 +c update employee set firstnme = 'Mary'
where empno = '000050'
```

Primeiro liga-se à base de dados SAMPLE e, em seguida, executa-se uma declaração de actualização sobre a linha com "empno = 50000" na tabela EMPLOYEE. A opção "+c" indica que não se quer que a consola de DB2 faça automaticamente o COMMIT da declaração. Faz-se isto de propósito, para que se fique com os *locks*.

3. Através de CLP2 corra os comandos:

```
db2 connect to sample
db2 +c update employee set firstnme = 'Tom' where
empno = '000030'
```

Na consola CLP2, que representa a segunda aplicação, também estamos a fazer uma ligação à base de dados SAMPLE, mas agora estamos a actualizar uma outra linha da tabela EMPLOYEE.

4. Da consola CLP1 execute:

```
db2 +c select firstnme from employee where empno =
'000030'
```

Depois de carregar Enter para executar o SELECT anterior, o SELECT parece ficar preso. Na verdade ele não está preso, mas à espera da libertação do *lock* exclusivo, que foi adquirido por CLP2 sobre esta linha no ponto 3. Neste momento, se o LOCKTIMEOUT tem o seu valor padrão de -1, a aplicação CLP1 irá esperar indefinidamente.

5. Da consola CLP2 execute:

```
db2 +c select firstnme from employee where empno =
'000050'
```

Ao executar o SELECT anterior, estamos a criar um *deadlock*. Este SELECT irá também parecer que fica preso, ficando à espera da libertação do *lock* exclusivo que foi adquirido por CLP1 sobre esta linha no ponto 3.

No cenário *deadlock* acima descrito, o DB2 irá verificar a configuração do parâmetro DLCHKTIME. Este parâmetro tem definido o intervalo de tempo para verificar se há *deadlocks*. Por exemplo, se este parâmetro está configurado para 10 segundos, o DB2 irá verificar a cada 10 segundos se um *deadlock* ocorreu. Se, de facto, um *deadlock* aconteceu, o DB2 irá utilizar um algoritmo interno para determinar qual das duas operações deve ser desfeita, e qual deve continuar.

Se estivermos a enfrentar vários *deadlocks*, deveremos reexaminar as nossas transacções e re-estrutura-las se possível.

13.9 As melhores práticas de concorrência e locking

Aqui estão algumas dicas a seguir para permitir a melhor concorrência possível:

1. Ter transacções o mais curtas possível. Isto pode ser conseguido através da execução frequente da declaração COMMIT (o mesmo para operações só de leitura (*read-only*)), quando a sua aplicação lógica o permitir.

2. Faça *log* da informação de uma transacção apenas quando necessário.

3. Investigar rapidamente os dados usando:

```
ALTER TABLE ACTIVATE NOT LOGGED INITIALLY WITH EMPTY
TABLE
```

4. Fazer modificações dos dados em *batches*/grupos. Por exemplo:

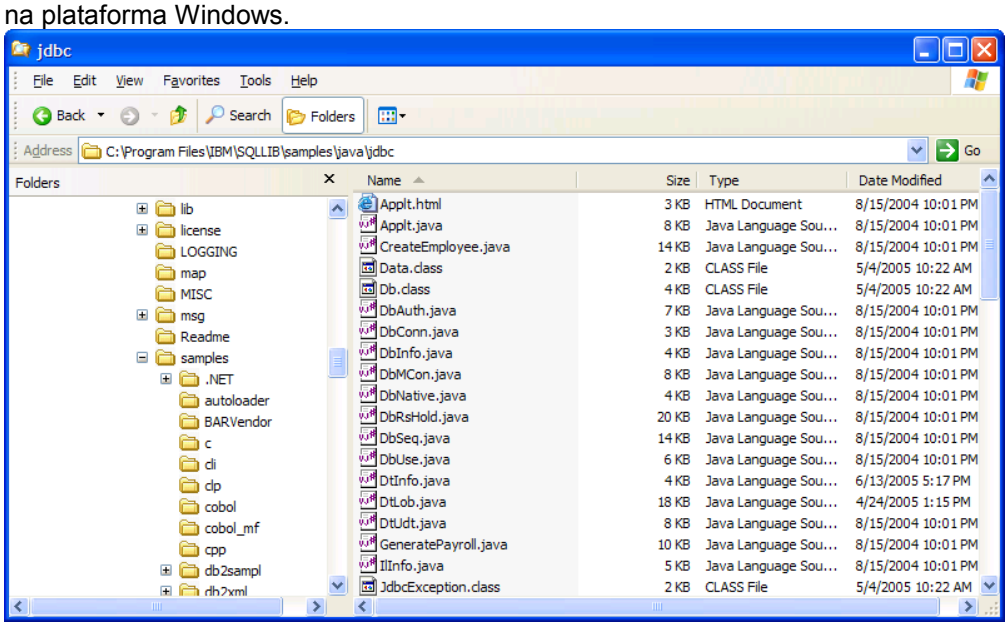
```
DELETE FROM (
   SELECT * FROM tedwas.t1 WHERE c1 = … FETCH FIRST
   3000 ROWS ONLY)
```

5. Usar funcionalidades de concorrência nas ferramentas de movimentação de dados do DB2.

6. Definir o parâmetro LOCKTIMEOUT (os tempos sugeridos são entre 30-120 segundos). Não deixar o valor padrão de -1. Também se pode usar *locks timeout* para a sessão.
7. Não ir buscar mais dados do que é necessário. Por exemplo, use a cláusula FETCH FIRST n ROWS ONLY em declarações SELECT.

Parte III – Aprender DB2: Desenvolvimento de Aplicações

Na parte III deste livro, discutiremos com detalhe objectos de aplicação de base de dados como: *stored procedures*, *user-defined functions* (UDFs), e *triggers*. Note que pode aprender a programar em diferentes linguagens com o DB2 como servidor de dados analisando as aplicações-exemplo integradas na instalação do servidor DB2 na directoria SQLLIB\samples. A figura que se segue ilustra programas-exemplo em Java fornecidos pelo DB2 na plataforma Windows.



Programas-exemplo em Java integrados no DB2

14

Capítulo 14 – SQL PL *Stored Procedures*

Neste capítulo iremos discutir *stored procedures*. Um *stored procedure* é um objecto de aplicação de base de dados que pode encapsular comandos SQL. Manter parte da lógica da aplicação na base de dados permite um melhor desempenho, e assim a quantidade de tráfego da rede entre a aplicação e a base de dados é reduzida consideravelmente. Os *stored procedures* também fornecem uma posição central para armazenar o código, para que outras aplicações possam reutilizar os mesmos procedimentos.

Os DB2 *stored procedures* podem ser desenvolvidos com SQL PL, C/C++, Java, Cobol, linguagens suportadas pela CRL (Common Language Runtime), e OLE. Neste capítulo, iremos discutir os procedimentos SQL PL pela sua popularidade e simplicidade.

A Figura 14.1 ilustra o funcionamento de *stored procedures*

.

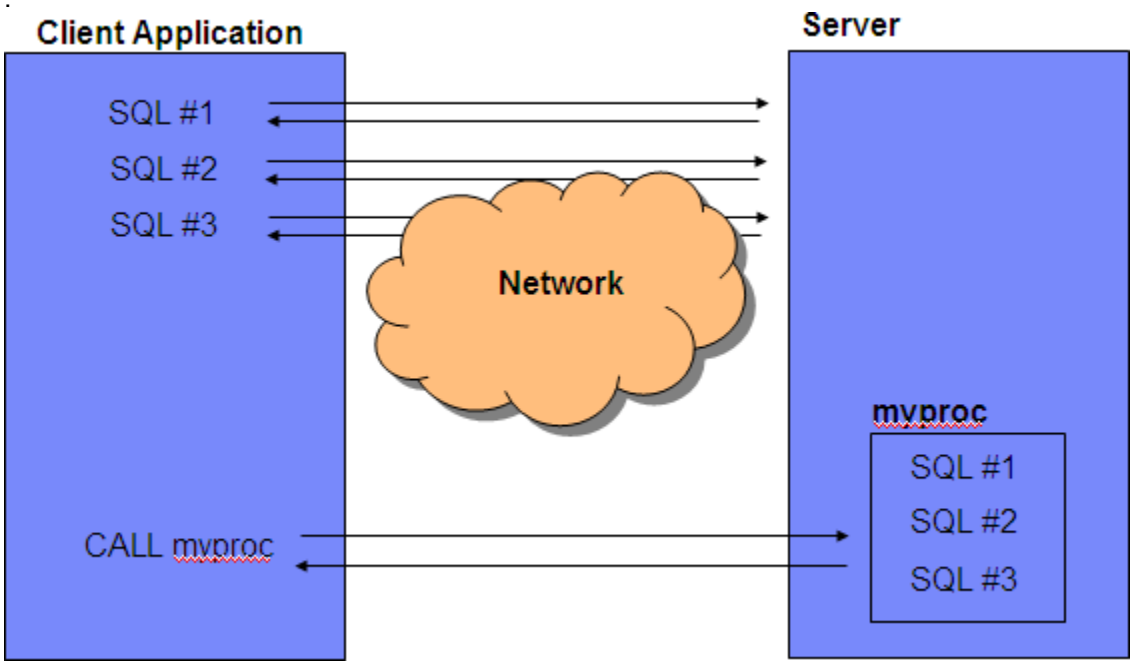


Figura 14.1 – Redução do tráfego da rede com *stored procedures*

No canto superior esquerdo da figura, podemos ver comandos SQL a serem executados sequencialmente. Cada comando SQL é enviado do cliente para o servidor, e este retorna o resultado para o cliente. Se forem executados muitos comandos SQL desta forma, o tráfego da rede aumenta. Por outro lado, no canto inferior direito, podemos ver um *stored procedure* chamado "myproc" guardado no servidor, que contém o mesmo SQL; o cliente (lado esquerdo) invoca o procedimento com comando CALL. Este segundo método de invocação é mais eficiente, pois só um comando atravessa a rede, e um resultado é retornado para o cliente.

Stored procedures podem também ser úteis na segurança da base de dados. Por exemplo, podemos garantir o acesso dos utilizadores a tabelas ou *views* apenas através de *stored procedures*, o que ajuda a proteger o servidor e a garantir que os utilizadores só têm acesso à informação que devem. Isto é possível porque os utilizadores não necessitam de privilégios explícitos nas tabelas ou vistas que acedem atráves dos *stored procedures*; apenas precisam de possuir o privilégio de invocação do *stored procedure*.

Nota:
Para mais informação sobre *stored procedures* SQL PL, veja este vídeo:
http://www.channeldb2.com/video/video/show?id=807741:Video:4343

14.1 O IBM Data Studio

O IBM Data Studio é um programa interactivo que ajuda a desenvolver e gerir aplicações de base de dados ao longo do ciclo de vida da gestão de dados. Algumas das operações que podem ser realizadas com o Data Studio são:

- Criar, alterar, e eliminar objectos da base de dados DB2 (com análise de impacto)
- Explorar e editar dados – relacionais e XML
- Criar interactivamente comandos SQL e Xquery
- Optimizar consultas (“queries”) utilizando o Visual Explain
- Desenvolver, corrigir, e distribuir “stored procedures” tanto em SQL como em Java™
- Desenvolver “user defined functions” (UDFs)
- Desenvolver aplicações SQLJ
- Desenvolver consultas e rotinas para aplicações pureXML
- Executar tarefas de transferência de dados
- Colaborar e partilhar projectos com membros de equipa
- Criar rapidamente *Web Services* SOAP e REST
- Descobrir relações entre objectos da base de dados com modelos de dados físicos (“diagramming”)
- Visualizar a distribuição dos dados nas tabelas

O IBM Data Studio é baseado na plataforma Eclipse. É um pacote independente (i.e. não é uma parte do pacote de instalação DB2), mas também é gratuito. O pacote Data Studio pode ser descarregado da tab “Download” no web site ibm.com/db2/express. A Figura 14.2 mostra o IBM Data Studio.

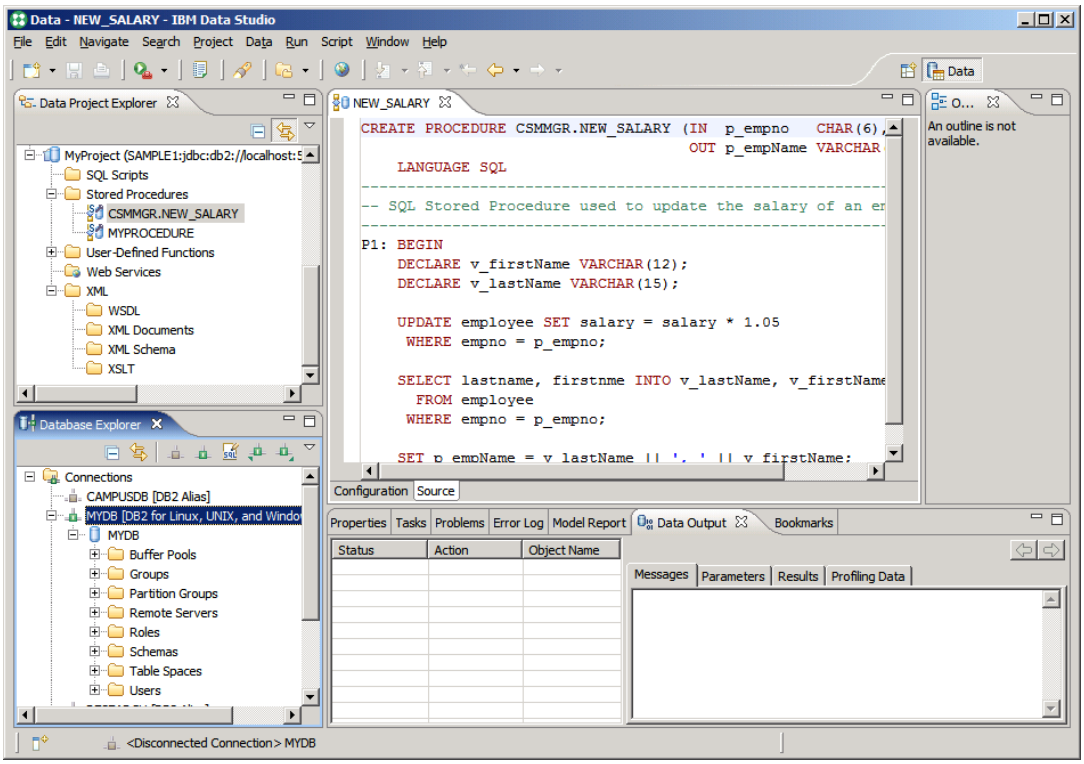


Figura 14.2 – O IBM Data Studio

14.1.2 Criar um "stored procedure" no Data Studio

Para criar um "stored procedure" Java ou SQL PL no Data Studio, siga os passos seguintes. Note que os "stored procedures" escritos noutras linguagens não podem ser criados no Data Studio.

Passo 1: Crie um projecto Data Studio
No menu Data Studio, escolha *File -> New -> Project* e escolha *Data Development Project*. Este passo é ilustrado na Figura 14.3

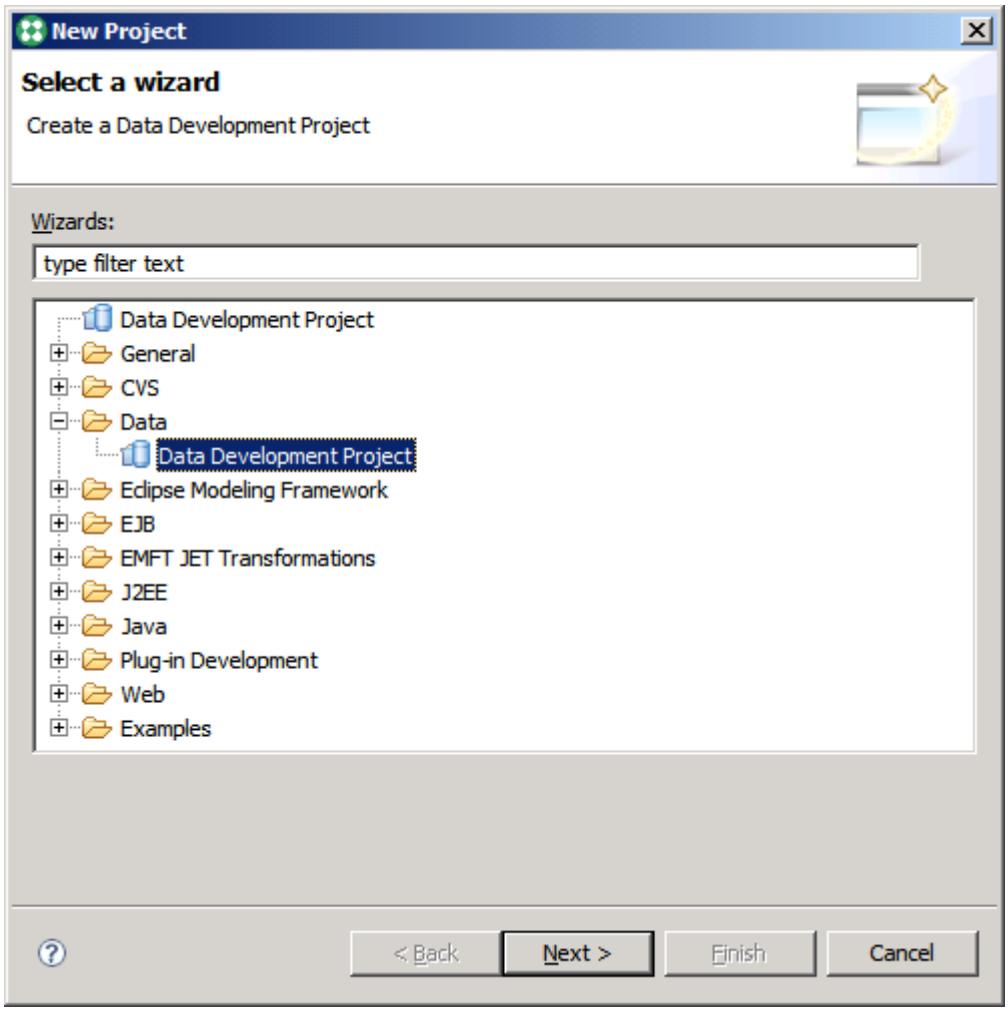


Figura 14.3 – O projecto de desenvolvimento de dados

Siga os passos do assistente – *wizard* - para inserir o nome do projecto, indicar a base de dados para ligar-se como parte do projecto e especificar o directório JDK (o fornecido por omissão é normalmente correcto).

Passo 2: Criar o "stored procedure"
Quando o projecto é criado, o lado esquerdo da perspectiva de dados irá mostrar o projecto. Na Figura 14.4 podemos ver o projecto "myProject" criado e expandido.

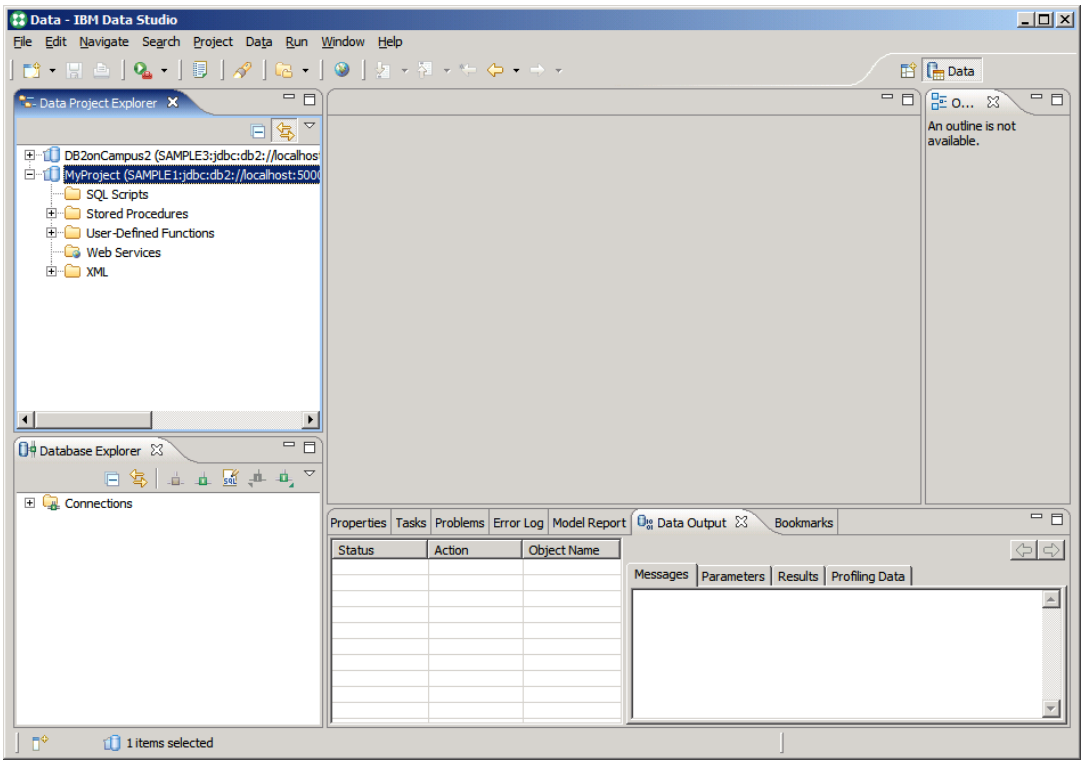


Figura 14.4 – O projecto "myProject"

A Figura 14.4 mostra as diferentes pastas do projecto. Para criar um "stored procedure", clique com o botão direito do rato na pasta "Stored Procedures" e escolha *New -> Stored Procedure*. Complete a informação do *wizard* "New Stored Procedure": projecto associado ao "procedure"; nome e linguagem do "procedure" (note que apenas as linguagens SQL PL e Java são suportadas pelo IBM Data Studio), e os comandos SQL do "procedure". Por omissão, o Data Studio fornece um exemplo de um comando SQL. Neste ponto, pode finalizar com um clique em *Finish.* O "stored procedure" é criado com o código "template" e os comandos SQL dados anteriomente como exemplo. (Figura 14.5)

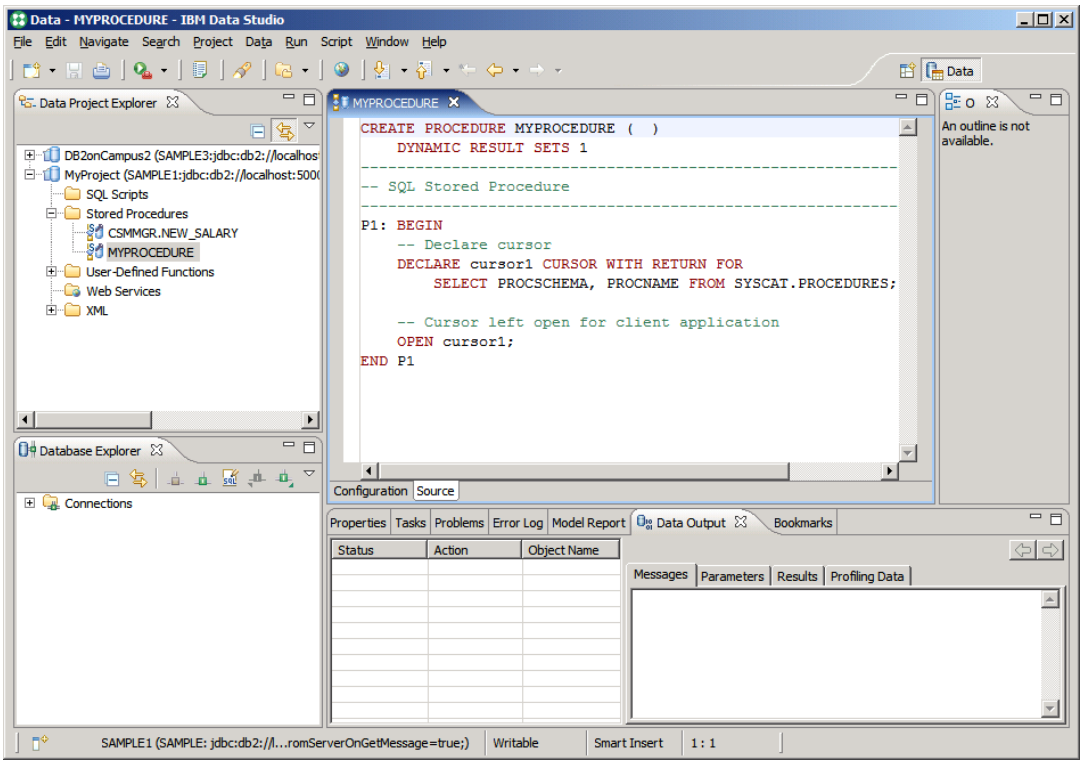


Figura 14.5 – Exemplo de um "stored procedure"

Na Figura 14.5, o código do "stored procedure" exemplo "MYPROCEDURE" foi gerado. Podemos substituir qualquer parte do código pelo nosso código. Para simplificar, iremos continuar a utilizar neste livro o "stored procedure" exemplo como se fosse escrito por nós.

Passo 3: Compilar e distribuir um "stored procedure"
Uma vez criado o "stored procedure", podemos compilar e distribuir o "stored procedure" através de um clique com o botão direito neste no painel da esquerda, e escolher "Deploy". (Figura 14.6)

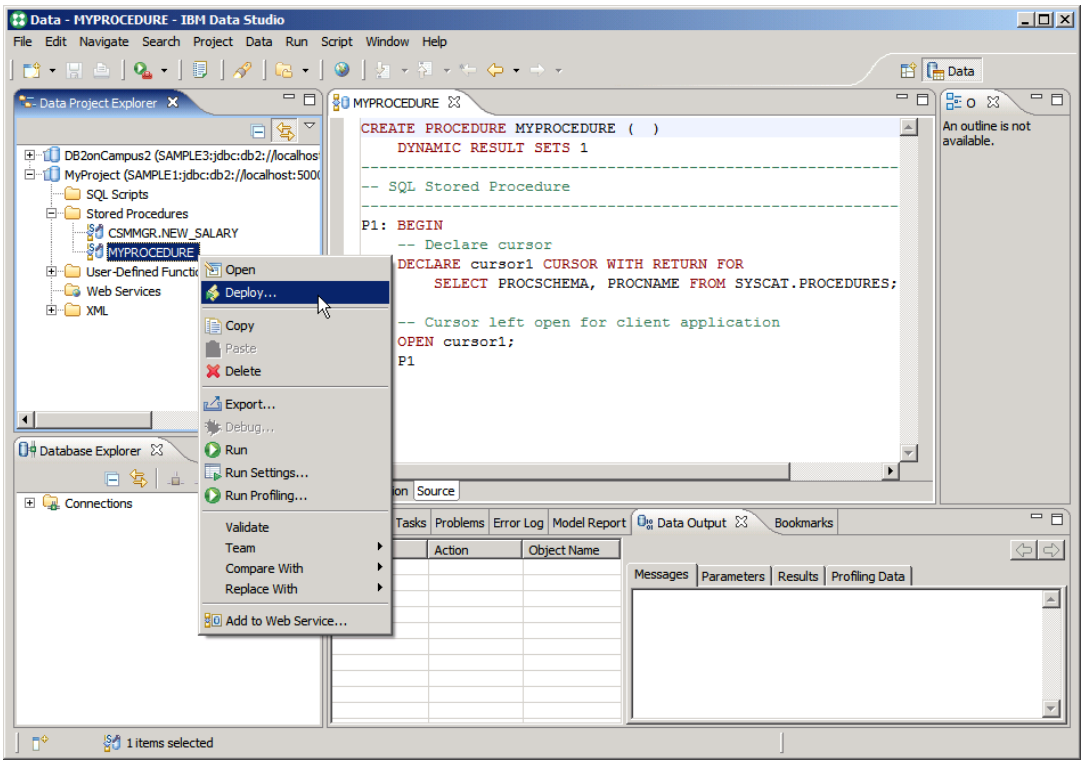


Figura 14.6 – Distribuir um "stored procedure"

Passo 4: Executar um "stored procedure"
Assim que o "stored procedure" é distribuído, podemos executá-lo com um clique no botão direito do rato neste e escolher "Run". Os resultados aparecem na tab "Results" no canto inferior direito da janela.
Para executar um "stored procedure" do Command Windows ou do Command Editor, podemos usar `CALL <procedure name>`

14.2 Introdução a *stored procedures* SQL PL

Os *stored procedures* SQL Procedural Language (SQL PL) são facéis de criar e aprender. Possuem um melhor desempenho em DB2. *Stored procedures* SQL PL (ou simplesmente *SQL stored procedures*) são o foco deste capítulo.

14.2.1 Estrutura de um *stored procedure*

Sintaxe básica de um *stored procedure*:

```
CREATE PROCEDURE proc_name [( {optional parameters} )]
                    [optional procedure attributes]      <statement>
```

Onde `<statement>` é um único comando, ou um conjunto de comandos agrupados por `BEGIN [ATOMIC] ... END`

14.2.2 Atributos opcionais de *stored procedures*

Os tópicos seguintes descrevem alguns atributos opcionais:

- `LANGUAGE SQL`
  Este atributo indica a linguagem que o *stored procedure* irá usar. O valor default é LANGUAGE SQL. Para outras linguagens, como Java ou C utilize LANGUAGE JAVA ou LANGUAGE C, respectivamente.

- `RESULT SETS <n>`
  Necessário se o *stored procedure* retornar n resultados.

- `SPECIFIC my_unique_name`
  Este é um nome único que pode ser dado a um *procedure*. Um *stored procedure* pode ser *overloaded* isto é, vários *stored procedures* podem ter o mesmo nome, mas diferentes números de paramêtros. Usando a palavra-chave (*keyword*) SPECIFIC podemos fornecer um nome único a cada um destes *stored procedures*, o que facilita a gestão de *stored procedures*. Por exemplo, para remover um *stored procedure* utilizando a palavra-chave SPECIFIC, pode invocar o comando: `DROP SPECIFIC PROCEDURE.` Se a palavra-chave SPECIFIC não for utilizada é necessário o comando DROP PROCEDURE e o nome do *procedure* com os paramêtros para que o DB2 saiba qual dos *procedures overloaded* queremos eliminar.

14.2.3 Paramêtros

Existem três tipos de parâmetros num *stored procedure* SQL PL:

- IN - Input parameter (parâmetro de entrada)
- OUT - Output parameter (parâmetro de saída)
- INOUT - Input and Output parameter (parâmetro de entrada e saída)

Por exemplo:

```
CREATE PROCEDURE proc(IN p1 INT, OUT p2 INT, INOUT p3 INT)
```

Quando se invoca o procedimento é necessário fornecer todos os parâmetros no comando CALL. Por exemplo, para invocar o *stored procedure* acima teremos de especificar:

```
CALL proc (10,?,4)
```

O sinal de interrogação (?) é utilizado no parâmetro OUT no comando CALL.

Eis mais um exemplo de um *stored procedure* com parâmetros que pode testar:

```
CREATE PROCEDURE P2 ( IN    v_p1 INT,
                      INOUT v_p2 INT,
                      OUT   v_p3 INT)
LANGUAGE SQL
SPECIFIC myP2
```

```
BEGIN
    -- my second SQL procedure
    SET v_p2 = v_p2 + v_p1;
    SET v_p3 = v_p1;
END
```

Para invocar o procedimento no Command Editor utilize:

```
call P2 (3, 4, ?)
```

14.2.4 Comentários num *stored procedure* SQL PL

Existem duas maneiras de especificar comentários num *stored procedure* SQL PL:

- Utilizando dois hifens. Por exemplo:
  `-- This is an SQL-style comment`
- Utilizando um formato semelhante ao da linguagem C. Por exemplo:
  `/* This is a C-style coment */`

14.2.5 Comandos compostos

Um comando composto num *stored procedure* é um comando que consiste em várias instruções procedimentais e comandos SQL encapsulados pelas palavras-chave BEGIN e END. Quando a palavra-chave ATOMIC segue a BEGIN, o comando composto é tratado como uma unidade átomica, isto é, o comando composto é bem sucedido quando todas as instruções ou comandos também o são. Se um dos comandos não o for, então as instruções são desfeitas. A Figura 14.7 ilustra a estrutura de um comando composto.

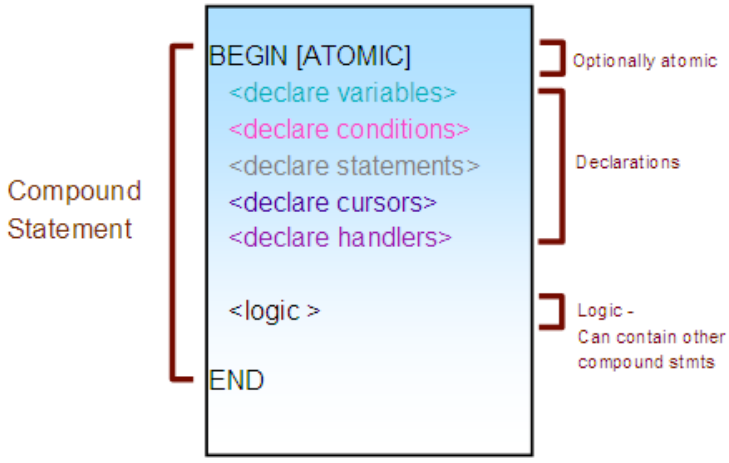


Figura 14.7 – Comandos compostos

14.2.6 Declaração de variáveis

Para declarar uma variável, utilize o comando DECLARE:

```
DECLARE var_name <data type> [DEFAULT value];
```

Alguns exemplos:

```
  DECLARE temp1 SMALLINT DEFAULT 0;
  DECLARE temp2 INTEGER DEFAULT 10;
  DECLARE temp3 DECIMAL(10,2) DEFAULT 100.10;
  DECLARE temp4 REAL DEFAULT 10.1;
  DECLARE temp5 DOUBLE DEFAULT 10000.1001;
  DECLARE temp6 BIGINT DEFAULT 10000;
  DECLARE temp7 CHAR(10) DEFAULT 'yes';
  DECLARE temp8 VARCHAR(10) DEFAULT 'hello';
  DECLARE temp9 DATE DEFAULT '1998-12-25';
  DECLARE temp10 TIME DEFAULT '1:50 PM';
  DECLARE temp11 TIMESTAMP DEFAULT '2001-01-05-12.00.00';
  DECLARE temp12 CLOB(2G);
  DECLARE temp13 BLOB(2G);
```

14.2.7 Comandos de atribuição

Para atribuir um valor a uma variável, utilize o comando SET. Por exemplo:

```
SET total = 100;
```

O comando acima é equivalente a

```
VALUES(100) INTO total;
```

Adicionalmente, qualquer variável pode ser atribuída a NULL:

```
SET total = NULL;
```

No exemplo a seguir, um erro será gerado se as consultas SQL devolveram mais do que um resultado:

```
SET total = (select sum(c1) from T1);
SET first_val = (select c1 from T1 fetch first 1 row only)
```

Também podemos associar variáveis de acordo com propriedades externas da base de dados:

```
SET sch = CURRENT SCHEMA;
```

14.3 Cursores

Um cursor é um conjunto de resultados que contém o resultado de um comando SELECT. Explica-se de seguida a sintaxe para declarar, abrir, extrair, e fechar um cursor:

```
DECLARE <cursor name> CURSOR [WITH RETURN <return target>]
      <SELECT statement>;
OPEN  <cursor name>;
FETCH <cursor name> INTO <variables>;
CLOSE <cursor name>;
```

Quando um cursor é declarado, a claúsula WITH RETURN pode ser utilizada com estes valores:

- CLIENT: o conjunto de resultados é retornado para aplicação do cliente
- CALLER: o conjunto de resultados é retornado para o cliente ou para o *stored procedure* que o invocou.

Segue-se um exemplo de um *stored procedure* que utiliza um cursor:

```
CREATE PROCEDURE set()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
DECLARE cur CURSOR WITH RETURN TO CLIENT
        FOR SELECT name, dept, job
         FROM staff
         WHERE salary > 20000;
OPEN cur;
END
```

14.4 Controlo de fluxo

Tal como outras linguagens, a linguagem SQL PL possui vários comandos que podem ser usados para controlar o fluxo lógico. Em baixo listamos alguns dos comandos de controlo de fluxo suportados:

CASE (selecciona um caminho de execução (procura simples))
IF
FOR (executa o corpo de cada linha da tabela)
WHILE
ITERATE (força a próxima iteração. Semelhante ao CONTINUE em C)
LEAVE (sai de um bloco ou ciclo. "Structured Go to")
LOOP (ciclo infinito)
REPEAT
GOTO
RETURN
CALL (invocação de um procedimento)

14.5 Invocar *stored procedures*

Os próximos exemplos mostram como utilizar o comando CALL para invocar *stored procedures* utilizando linguagens de programação diferentes.

Exemplo: Invocar um *stored procedure* de uma aplicação CLI/ODBC

```
    SQLCHAR *stmt = (SQLCHAR *)
    "CALL MEDIAN_RESULT_SET( ? )" ;
      SQLDOUBLE   sal = 20000.0;  /* Bound to parameter marker in
    stmt */
      SQLINTEGER  salind  = 0;    /* Indicator variable for sal */
```

```
      sqlrc = SQLPrepare(hstmt, stmt, SQL_NTS);
      sqlrc = SQLBindParameter(hstmt, 1, SQL_PARAM_OUTPUT,
        SQL_C_DOUBLE, SQL_DOUBLE, 0, 0, &sal, 0, &salind);
      SQLExecute(hstmt);


if (salind == SQL_NULL_DATA)
    printf("Median Salary = NULL\n");
  else
    printf("Median Salary = %.2f\n\n", sal );

/* Get first result set */
    sqlrc = StmtResultPrint(hstmt);
/* Check for another result set */
    sqlrc = SQLMoreResults(hstmt);
  if (sqlrc == SQL_SUCCESS)  {
    /* There is another result set */
    sqlrc = StmtResultPrint(hstmt);
  }
```

Para mais detalhes, veja o ficheiro exemplo DB2: sqllib/samples/sqlproc/rsultset.c

Exemplo: Invocar um s*tored procedure* de uma aplicação VB.NET

```
Try
  ' Create a DB2Command to run the stored procedure
  Dim procName As String = "TRUNC_DEMO"
  Dim cmd As DB2Command = conn.CreateCommand()
  Dim parm As DB2Parameter

  cmd.CommandType = CommandType.StoredProcedure
  cmd.CommandText = procName

  ' Register the output parameters for the DB2Command
  parm           = cmd.Parameters.Add("v_lastname", DB2Type.Var-
Char)
  parm.Direction = ParameterDirection.Output
  parm           = cmd.Parameters.Add("v_msg", DB2Type.VarChar)
  parm.Direction = ParameterDirection.Output

  ' Call the stored procedure
  Dim reader As DB2DataReader = cmd.ExecuteReader

Catch myException As DB2Exception
  DB2ExceptionHandler(myException)
Catch
  UnhandledExceptionHandler()
End Try
```

Exemplo: Invocar um *stored procedure* de uma aplicação Java

```
try
{
  // Connect to sample database
  String url = "jdbc:db2:sample";
  con = DriverManager.getConnection(url);


  CallableStatement cs = con.prepareCall("CALL trunc_demo(?, ?)");

  // register the output parameters
  callStmt.registerOutParameter(1, Types.VARCHAR);
  callStmt.registerOutParameter(2, Types.VARCHAR);

  cs.execute();
  con.close();
}
catch (Exception e)
{
  /* exception handling logic goes here */
}
```

14.6 Erros e *handlers* de condições

No DB2, as palavras-chave SQLCODE e SQLSTATE são utilizadas para determinar o sucesso ou erro na execução de um comando SQL. Estas palavras-chave precisam de ser declaradas explicitamente na parte mais exterior do procedimento da seguinte forma:

```
DECLARE SQLSTATE CHAR(5);
DECLARE SQLCODE INT;
```

O DB2 irá automaticamente atribuir valores às palavras-chave acima após cada operação SQL. Para o SQLCODE, os valores podem ser:
1. = 0, bem sucessido
2. > 0, bem sucessido com aviso
3. < 0, mal sucessido
4. = 100, sem dados. (i.e.: o comando FETCH não retornou dados)

Para o SQLSATE, os valores podem ser:
- sucesso: SQLSTATE '00000'
- não encontrado: SQLSTATE '02000'
- aviso: SQLSTATE '01XXX'
- excepção: restantes valores

O SQLCODE é específico do motor da base de dados, e é mais detalhado do que o SQLSTATE. O SQLSTATE é standard entre motores de bases de dados mas é geralmente muito genérico. Vários SQLCODEs podem corresponder a um SQLSTATE.

Uma condição pode surgir com qualquer comando SQL e corresponde a um SQLSTATE. Por exemplo, uma condição específica como SQLSTATE '01004' surge quando o valor é

truncado durante uma operação SQL. Para testar esta condição podemos atribuir nomes, em vez de SQLSTATE '01004'. No próximo exemplo em particular, o nome “trunc” é associado à condição SQLSTATE '01004'.

```
DECLARE trunc CONDITION FOR SQLSTATE '01004'
```

Outras condições gerais pré-definidas são:

- SQLWARNING
- SQLEXCEPTION
- NOT FOUND

Handling de condições

Para resolver uma condição, podemos criar um *handler* de condição que especifíca:

1. a condição a resolver
2. onde retoma a execução (baseado no tipo de *handler*: CONTINUE, EXIT ou UNDO)
3. as acções a realizar para tratar a condição. As acções podem ser quaisquer comandos, incluíndo estruturas de controlo.

Se uma condição SQLEXCEPTION é gerada, e não existe *handler* para esta, o procedimento termina e retorna um erro ao cliente.

Tipos de *handlers*
Existem três tipos de *handlers*:

CONTINUE – Este *handler* é utilizado para indicar que depois de uma excepção ser gerada, o fluxo irá continuar (CONTINUE) para o próximo comando depois do comando que gerou a condição.
EXIT – Este *handler* é utilizado para indicar que, depois de uma excepção ser gerada, o fluxo irá para o final do procedimento.
UNDO – Este *handler* é utilizado para indicar que depois de uma excepção ser gerada, o fluxo irá para o final do procedimento, e irá desfazer os comandos realizados.

A Figura 14.8 ilustra os diferentes *handlers* de condição e o seu comportamento.

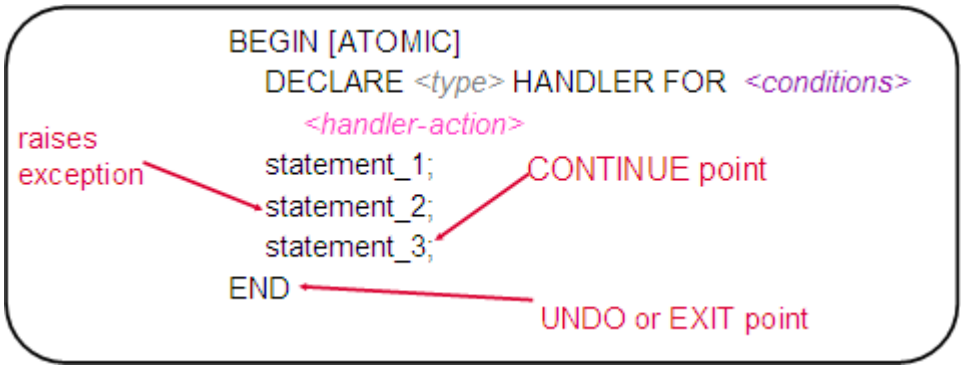


Figura 14.8 – Tipos de *handlers* de condição

14.7 SQL dinâmico

Em SQL dinâmico, em oposição ao SQL estático, o comando SQL não é totalmente conhecido até ao momento de execução. Por exemplo se `col1` e `tabname` são variáveis neste comando, então estamos no campo do SQL dinâmico:

```
'SELECT ' || col1 || ' FROM ' || tabname;
```

SQL dinâmico também é recomendado para DDL para evitar problemas de dependência. É também necessário para implementar recursividade.

SQL dinâmico pode ser executado através de duas vias:

- Utilizando o comando EXECUTE IMMEDIATE – ideal para uma execução singular de SQL
- Utilizando o comando PREPARE com o comando EXECUTE – ideal para multípla execução de SQL

O exemplo de fragmento de código que se segue fornece um exemplo de SQL Dinâmico utilizando as duas vias. Este exemplo assume que uma tabela T2 foi criada com a seguinte definição:

```
CREATE TABLE T2 (c1 INT, c2 INT)

CREATE PROCEDURE dyn1 (IN value1 INT, IN value2 INT)
SPECIFIC dyn1
BEGIN
DECLARE stmt varchar(255);
DECLARE st STATEMENT;

SET stmt = 'INSERT INTO T2 VALUES (?, ?)';

PREPARE st FROM stmt;

EXECUTE st USING value1, value1;
EXECUTE st USING value2, value2;

SET stmt = INSERT INTO T2 VALUES (9,9)';
EXECUTE IMMEDIATE stmt;
END
```

15

Capítulo 15 – Inline SQL PL, *Triggers*, e UDFs

Neste capítulo iremos discutir inline SQL PL (Linguagem de programação para SQL embebido – "inline") e outros objectos de aplicaçãos para base de dados como *user-defined functions* (UDFs) e *triggers.*

15.1 Inline SQL PL

No Capítulo 14 discutimos a criação de *stored procedures* usando a linguagem SQL PL. A linguagem SQL PL também pode ser utilizada "inline" significando que se sustém sem a criação de um *stored procedure*. A SQL PL usada em UDFs e *triggers* é também inline porque o código é adicionado inline com o código UDF/*trigger*, e é SQL dinâmico no estado natural. Inline SQL PL suporta apenas um subconjunto de todos os comandos SQL PL. Apresenta-se de seguida os keywords dos inline SQL PL suportados:

```
DECLARE <variable>
SET
CASE
FOR
GET DIAGNOSTICS
GOTO
IF
RETURN
SIGNAL
WHILE
ITERATE
LEAVE
```

Lista dos *keywords* inline SQL PL não suportados:

```
ALLOCATE CURSOR
ASSOCIATE LOCATORS
DECLARE <cursor>
DECLARE ...HANDLER
PREPARE
EXECUTE
EXECUTE IMMEDIATE
LOOP
REPEAT
RESIGNAL
CALL
COMMIT/ROLLBACK
```

Apresenta-se um exemplo de comando complexo SQL usando inline SQL PL. Se quiser experimentar, pode inseri-lo num ficheiro script, assegurando-se que cria as seguintes tabelas:

```
CREATE TABLE T1 (c1 INT)
CREATE TABLE T3 (c1 INT)
```

```
BEGIN ATOMIC
  DECLARE cnt      INT DEFAULT 0;
  DECLARE sumevens INT DEFAULT 0;
  DECLARE err_msg  VARCHAR(1000) DEFAULT '';
  WHILE (cnt < 100) DO
    IF mod(cnt,2) = 0 THEN
       SET sumevens = sumevens + cnt;
    END IF;
    SET cnt=cnt+1;
  END WHILE;
  INSERT INTO T3 values (sumevens);
  SET cnt = (SELECT 0 FROM SYSIBM.SYSDUMMY1);
  FOR cur1 AS SELECT * FROM T1 DO
    IF cur1.c1 > 100 THEN
      SET cnt = cnt + 1;
    END IF;
  END FOR;

  SET err_msg = 'Rows with values > 100 is:' || char(cnt);
  SIGNAL SQLSTATE '80000' SET MESSAGE_TEXT = err_msg;
END!
```

Se salvou o inline SQL PL em cima num ficheiro script chamado “myScript.txt” pode executa-lo da seguinte maneira:

```
db2 -td! –vf myScript.txt
```

15.2 *Triggers*

Triggers são objectos da base de dados associados a uma tabela que definem operações a serem executadas quando uma operação INSERT, UPDATE, ou DELETE é executada numa tabela. Eles são activados (ou “disparados”) automaticamente. As operações que provocam os “disparos” dos *triggers* chamam-se comandos SQL *triggering*.

15.2.1 Tipos de *triggers*

Existem três tipos de *triggers*: *before triggers*, *after triggers* e *instead of triggers*.

Before triggers
Os *before triggers* são activados antes de uma linha ser inserida, actualizada ou eliminada. As operações feitas por este *trigger* não podem activar outros *triggers* (logo operações INSERT, UPDATE, e DELETE não são permitidas)

Um exemplo de um *before trigger* simples é ilustrado na Figura 15.1.

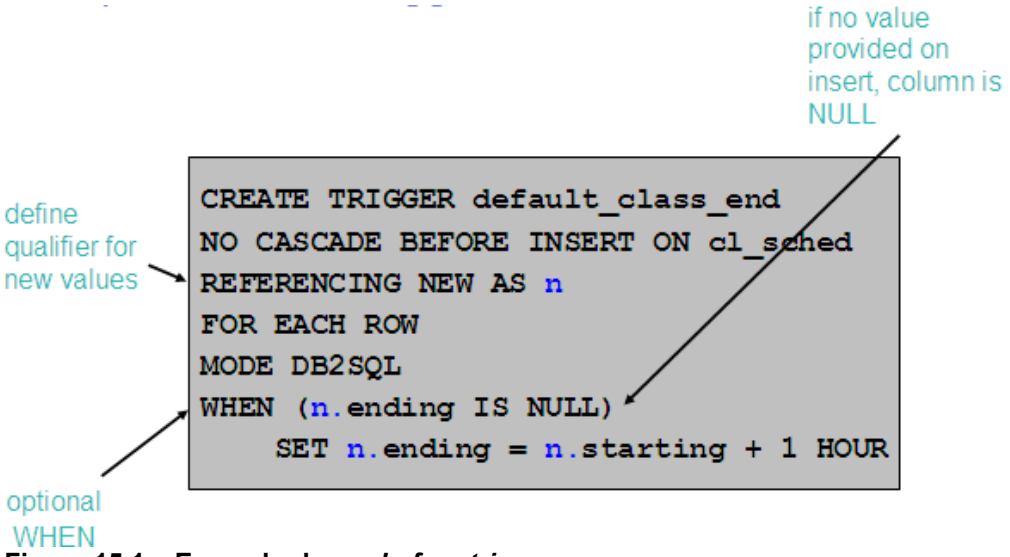


Figura 15.1 – Exemplo de um *before trigger*

Na Figura 15.1 o *trigger* "default_class_end" irá ser activado antes de uma operação INSERT SQL ser efectuada na tabela "cl_sched". Esta tabela é parte da base de dados SAMPLE, logo pode criar e testar este *trigger* quando estiver ligado a esta base de dados. A variável "n" na definição do *trigger* irá representar o novo valor na INSERT, que é, o valor a ser inserido. O *trigger* irá validar o conteúdo que está a ser inserido na tabela. Se a coluna "ending" não tiver valor durante uma inserção, o *trigger* irá assegurar que tem o valor da coluna "starting" acrescentado de uma hora.

Os seguintes comandos mostram como testar o *trigger*.

```
C:\Program Files\IBM\SQLLIB\BIN>db2 insert into cl_sched
(class_code, day, starting) values ('abc',1,current time)
DB20000I  The SQL command completed successfully.

C:\Program Files\IBM\SQLLIB\BIN>db2 select * from cl_sched

CLASS_CODE DAY    STARTING ENDING
---------- ------ -------- --------
042:BF          4 12:10:00 14:00:00
553:MJA         1 10:30:00 11:00:00
543:CWM         3 09:10:00 10:30:00
778:RES         2 12:10:00 14:00:00
044:HD          3 17:12:30 18:00:00
abc             1 11:06:53 12:06:53

6 record(s) selected.
```

O *trigger* "validate_sched" mostrado em baixo extende a funcionalidade do *trigger* "default_class_end" anteriormente descrito. Novamente, pode criá-lo e testar sobre a base de dados SAMPLE.

```
CREATE TRIGGER validate_sched
NO CASCADE BEFORE INSERT ON cl_sched
REFERENCING NEW AS n
FOR EACH ROW
MODE DB2SQL
BEGIN ATOMIC
-- supply default value for ending time if null
IF (n.ending IS NULL) THEN
    SET n.ending = n.starting + 1 HOUR;
END IF;

-- ensure that class does not end beyond 9pm
IF (n.ending > '21:00') THEN
    SIGNAL SQLSTATE '80000'
    SET MESSAGE_TEXT='class ending time is beyond 9pm';
ELSEIF (n.DAY=1 or n.DAY=7) THEN
    SIGNAL SQLSTATE '80001'
        SET MESSAGE_TEXT='class cannot be scheduled on a weekend';
END IF;
END
```

After triggers

After triggers são activados depois que o comando SQL responsável pela sua activação tenha terminado a sua execução com sucesso. As operações realizadas por este *trigger* podem activar outros *triggers* (até um limite de 16 camadas de execução). *After triggers* suportam operações INSERT, UPDATE e DELETE. Em baixo pode ver um exemplo de um *after trigger*.

```
CREATE TRIGGER audit_emp_sal
AFTER UPDATE OF salary ON employee
REFERENCING OLD AS o NEW AS n
FOR EACH ROW
MODE DB2SQL
  INSERT INTO audit VALUES (
    CURRENT TIMESTAMP, ' Employee ' || o.empno || ' salary changed
from ' || CHAR(o.salary) || ' to ' || CHAR(n.salary) || ' by ' ||
USER)
```

Neste exemplo, o *trigger* "audit_emp_sal" é usado para realizar *auditing* na coluna "salary" da tabela "employee". Quando alguém modifica esta coluna, o *trigger* irá ser activado para escrever a informação sobre a modificação feita ao salário noutra tabela chamada "audit". A linha "OLD as o NEW as n" indica que o prefixo "n" irá ser usado para representar o novo valor que será recebido do comando UPDATE. Portanto, "o.salary" representa o valor existente ou antigo, e o "n.salary" representa o valor actualizado para a coluna salary.

Instead of triggers

Instead of triggers são definidos em *views*. A lógica definida no *trigger* é executada em vez do comando SQL que o activou. Por exemplo, ao realizar uma operação de actualização (update) numa *view*, o *instead of trigger* poderá ser activado para realizar a actualização nas tabelas originais que formam a *view*.

Os *triggers* não podem ser criados através do IBM Data Studio. Eles podem ser criados através do Control Center ou através das ferramentas de linha de Comandos (Command Window, Command Line Processor, ou o Command Editor).

Quicklab #12 – Criar *triggers* no Control Center

Objectivo

Triggers são objectos da base de dados utilizados para realizar alguma lógica aplicacional quando ocorre uma operação que modifica dados numa tabela. Neste Quicklab, irá criar um *trigger* utilizando no Control Center. Este *trigger* irá manter um registo com todas as modificações efectuadas na tabela SALES, utilizado para efeitos de auditoria. Irá registar o nome do utilizador que efectuou a alteração, assim como a hora em que a operação foi efectuada.

Procedimento

1. Abra o Control Center.

2. Para este Quicklab, irá precisar de criar tabelas adicionais para ser usadas para registar as alterações. Crie um tabela com as seguintes características:

   Nome da tabela: `saleslog`

   Primeira coluna:
   Nome: userid
   Tipo de dados: VARCHAR(128)
   Outros atributos: NOT NULL

   Segunda coluna
   Nome: daytime
   Tipo de dados: TIMESTAMP
   Outros atributos: NOT NULL

   Pista: Crie esta tabela usando o comando CREATE TABLE no Command Editor, ou utilize o *wizard Create Table* no Control Center.

3. No Control Center, expanda a pasta da base de dados *EXPRESS*. Botão direito na pasta *Triggers* e seleccione a opção *Create*. A janela interactiva *Create Trigger* aparece.

4. Preencha a seguinte informação na janela:

   Esquema do *trigger*: Nome do utilizador que está a ser utilizado (deverá ser a opção default)

   Nome do *trigger*: *audit_sales*

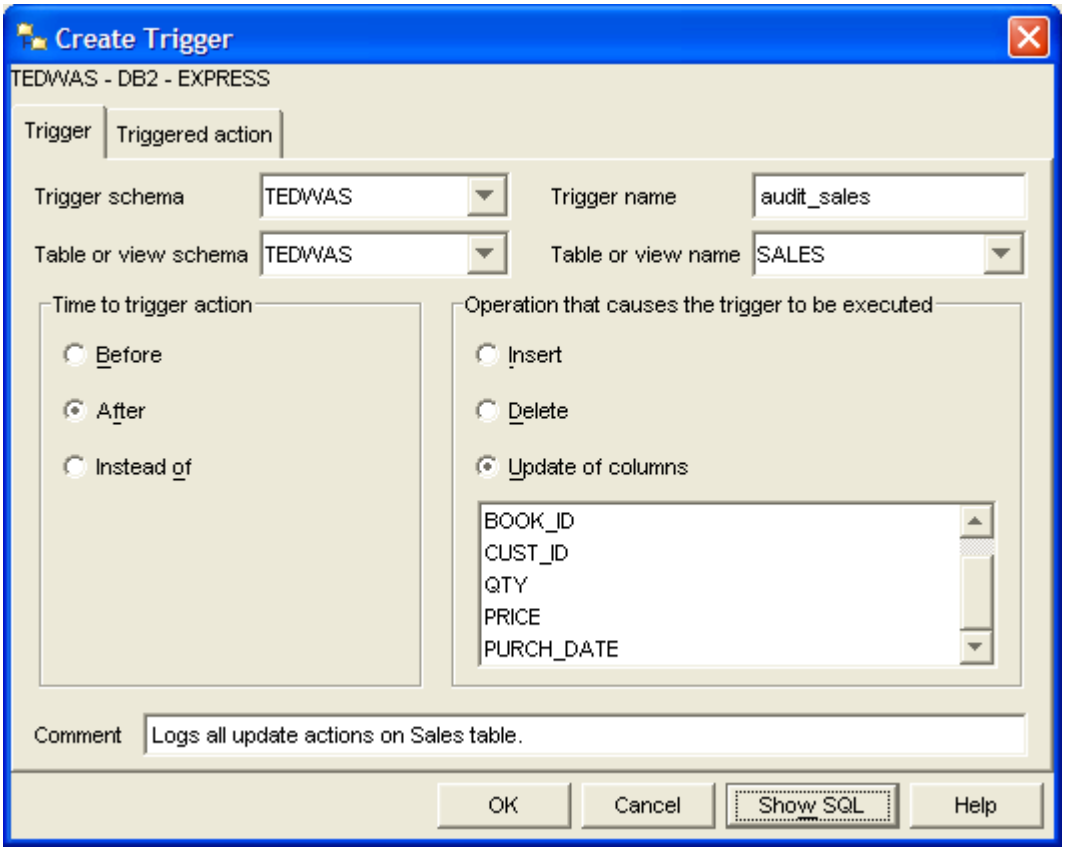
Esquema da tabela ou *view*: Nome de utilizador que está a ser utilizado (deverá ser a opção default)

Nome da tabela ou *view*: *SALES*

Tempo para a acção do *trigger*: *After*

Operações que provocam a execução do *trigger*: *Update of columns* (não especifique nenhumas colunas porque queremos que o *trigger* seja "disparado" quando alguma for actualizada.

Comentário: *Registar todas as acções na tabela Sales.*

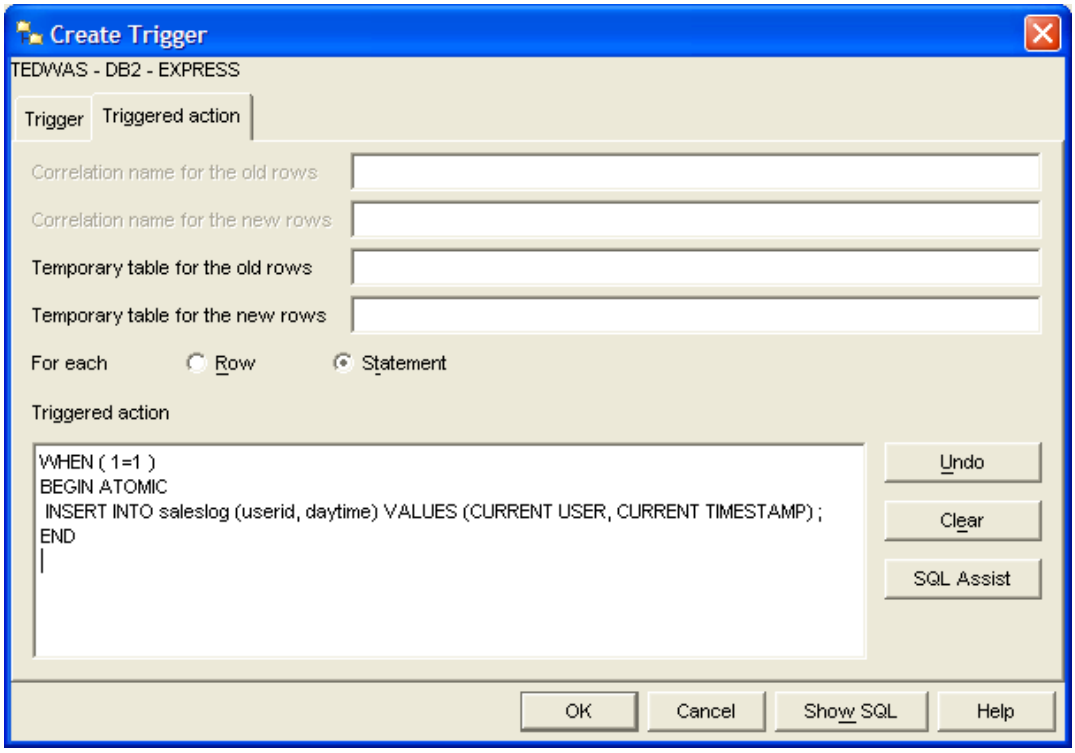


5. No separador *Triggered action*, seleccione a opção *For Each* ***STATEMENT***. Utilize o código seguinte para a acção do *trigger*:

```
WHEN ( 1=1 )
BEGIN ATOMIC
```

```
  INSERT INTO saleslog (userid, daytime) VALUES (CURRENT
USER, CURRENT TIMESTAMP);
END
```

(Nota: O comando do trigger inicia quando o comando que activou o trigger estiver finalizado. Um row trigger especifica que a acção do trigger irá ser executada cada vez que o comando SQL que activou o trigger afecta uma linha.)



Clique no botão OK para criar o *trigger*.

6. Agora deverá ser capaz de ver o *trigger* na pasta *Triggers* no Control Center.

7. Consulte a tabela *saleslog* para assegurar que esta não contém dados . Apague todas as linhas que a tabela possa conter (`DELETE FROM saleslog`).

8. Tente actualizar um registo na tabela SALES. (Ideia: utilize o Command Editor ou o SQL Assist Wizard).

9. Verifique o contéudo da tabela *saleslog* novamente. Quantas linhas existem?

15.3 *User-defined functions* (UDFs)

Uma *user-defined function* (UDF) é um objecto aplicacional para base de dados que transforma um conjunto de valores de dados de entrada num conjunto de valores de saída. Por exemplo, uma função pode receber medições em polegadas, e retornar o valor em centimetros.

O DB2 suporta a criação de funções que usam SQL PL, C/C++, Java, CLR (Commom Language Runtime), e OLE (Object Linking and Embedding). Neste livro, são salientadas as funções SQL PL pela sua simplicidade, popularidade, e desempenho.

Existem quatro tipos de funções: funções escalares, funções de tabelas, funções de linha e funções de coluna. Neste capítulo vamo-nos focar apenas em funções escalares.

15.3.1 Funções escalares

Funções escalares retornam um único valor. Funções escalares não incluem comandos SQL que podem mudar o estado da base de dados; isto é, os comandos INSERT, UPDATE e DELETE não são permitidos. Algumas funções escalares *built-in* são SUM(), AVG(), DIGITS(), COALESCE(), e SUBSTR().

O DB2 permite a criação de UDFs personalizadas, onde se pode encapsular lógica aplicacional utilizada frequentemente. Por exemplo, considere uma migração da sua aplicação de Oracle para DB2. Na sua aplicação, existem várias invocações da função NVL(), específica da plataforma Oracle. A função equivalente em DB2 chama-se COALESCE. Em vez de alterar todo o código e renomear todas as ocorrências de NVL para COALESCE, pode simplesmente criar uma nova UDF, de nome NVL, que invoca a função COALESCE. Isto é exemplificado abaixo.

```
CREATE FUNCTION NVL (p_var1 VARCHAR(30),
                     p_var2 VARCHAR(30))
SPECIFIC nvlvarchar30
RETURNS VARCHAR(30)
RETURN COALESCE(p_var1, p_var2)
```

A função COALESCE retorna o primeiro argumento não nulo.

Em baixo pode-se ver outro exemplo de uma função escalar. A função chama-se “deptname” e retorna o número do departamento de um empregado através do número de série do empregado.

```
CREATE FUNCTION deptname(p_empid VARCHAR(6))
RETURNS VARCHAR(30)
SPECIFIC deptname
BEGIN ATOMIC
  DECLARE v_department_name VARCHAR(30);
  DECLARE v_err VARCHAR(70);
  SET v_department_name = (
```

```
    SELECT d.deptname FROM department d, employee e
      WHERE e.workdept=d.deptno AND e.empno= p_empid);
  SET v_err = 'Error: employee ' || p_empid || ' was not found';
  IF v_department_name IS NULL THEN
    SIGNAL SQLSTATE '80000' SET MESSAGE_TEXT=v_err;
  END IF;
RETURN v_department_name;
END
```

Para testar a função, invoque na linha de Comandos ou na *shell* Linux/UNIX:

```
db2 "values (deptname ('000300'))"
```

Invocar UDFs escalares
UDFs escalares podem ser invocadas em comandos SQL sempre que um valor escalar é esperado, ou na claúsula VALUES. Em baixo estão dois exemplos para invocar a função escalar COALESCE:

```
SELECT DEPTNAME, COALESCE(MGRNO, 'ABSENT') FROM DEPARTMENT

VALUES COALESCE('A','B')
```

15.3.2 Funções tabelares

Funções tabelares retornam uma tabela de resultados. Estas funções podem ser invocadas da cláusula FROM de uma *query*. Ao contrário de funções escalares, as funções tabelares podem mudar o estado da base de dados; por isso, comandos INSERT, UPDATE e DELETE são permitidos. Algumas funções *built-in* que fazem parte do DB2 são SNAPSHOT_DYN_SQL( ) e MQREADALL( ). Funções tabelares são semelhantes a *views*, mas porque permitem modificar dados (através dos comandos INSERT, UPDATE e DELETE), permitem maior funcionalidade. Geralmente estas funções são utilizadas para devolver uma tabela de resultados e manter um registo para auditoria.

Segue-se um exemplo de uma **função tabelar** que enumera um conjunto de empregados de um departamento:

```
CREATE FUNCTION getEnumEmployee(p_dept VARCHAR(3))
RETURNS TABLE
  (empno CHAR(6),
   lastname VARCHAR(15),
   firstnme VARCHAR(12))
SPECIFIC getEnumEmployee
RETURN
  SELECT e.empno, e.lastname, e.firstnme
    FROM employee e
    WHERE e.workdept=p_dept
```

Para testar a função anterior, tente:

```
db2 "SELECT * FROM table(getEnumEmployee('D11')) AS t"
```

Invocar UDFs tabelares

Uma UDF tabelar tem de ser invocada na claúsula FROM de um comando SQL. A função TABLE() tem de ser utilizada e apelidada. A Figura 15.2 mostra um exemplo de como invocar a função "getEnumEmployee" que testamos anteriormente.

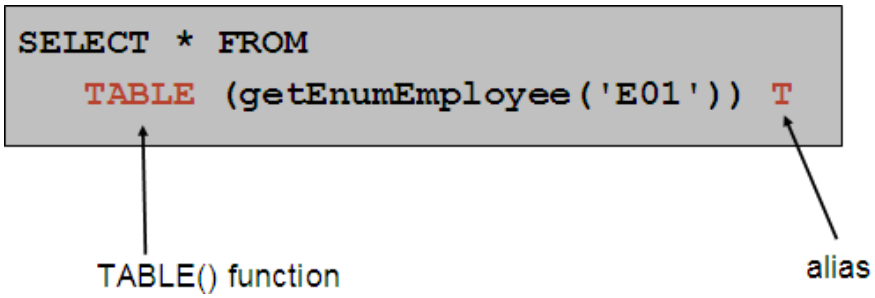


Figura 15.2 – Invocar uma função tabelar

Quicklab #13 – Criar uma UDF com o IBM Data Studio

Objectivo

Neste Quicklab, iremos criar uma UDF escalar no IBM Data Studio.
Ganhará mais experiência com o Data Studio, assim como se tornará mais familiar com a linguagem SQL PL para UDFs.

Procedimento

- Abra o IBM Data Studio (Dica: Está disponivel através do menu Iniciar).
- Na janela Data Project Explorer, escolha o projecto que criou no Quicklab anterior e seleccione *Abrir Projecto*.
- Clique com o botão direito do rato na pasta *User-Defined Functions*. Seleccione o item *New*.
  Seleccione o item *SQL User-Defined Function*. Podia alternativamente seleccionar o item *User-Defined Function using Wizard* se quiser ser guiado durante o processo usando um GUI *wizard*.
- O modo *Editor* deverá abrir com o esqueleto da função. Modifique o código da seguinte maneira:

```
CREATE FUNCTION booktitle(p_bid INTEGER)
RETURNS VARCHAR(300)
-----------------------------------------------------------
SQL UDF (Scalar)
-----------------------------------------------------------
SPECIFIC booktitle
F1: BEGIN ATOMIC
DECLARE v_book_title VARCHAR(300);
DECLARE v_err VARCHAR(70);
SET v_book_title = (SELECT title FROM books WHERE p_bid =
book_id);
SET v_err = 'Error: The book with ID ' || CHAR(p_bid) || '
was not found.';
IF v_book_title IS NULL THEN SIGNAL SQLSTATE '80000' SET MES-
SAGE_TEXT=v_err;
END IF;
RETURN v_book_title;
END
```

- Botão direito na função e escolha *Deploy* para “produzir” a função.

- Execute a função clicando no botão *Run* da barra de ferramentas.

- Desde o momento em que a função aceita um parâmetro de entrada, uma janela interactiva aparece a perguntar os valores do parâmetro.

  Introduza o valor: 80002

  Qual é o resultado?

  Tente novamente com o valor: 1002

  O que acontece desta vez? (Pista: Analise a secção *Messages* do *Output*). Look in the *Messages* section of the *Output* view).

- Feche o IBM Data Studio quando acabar.

16

Capítulo 16 – DB2 pureXML

Neste capítulo iremos discutir pureXML, a nova tecnologia fornecida pelo DB2 9 para suportar armazenamento de XML nativo. A maioria dos exemplos e dos conceitos discutidos neste capítulo foram retirados do IBM Redbook: *DB2 9: pureXML overview and fast start*. Verifique a secção Recursos para mais informações. A Figura 16.1 mostra a parte do mundo DB2 que iremos discutir neste capítulo.

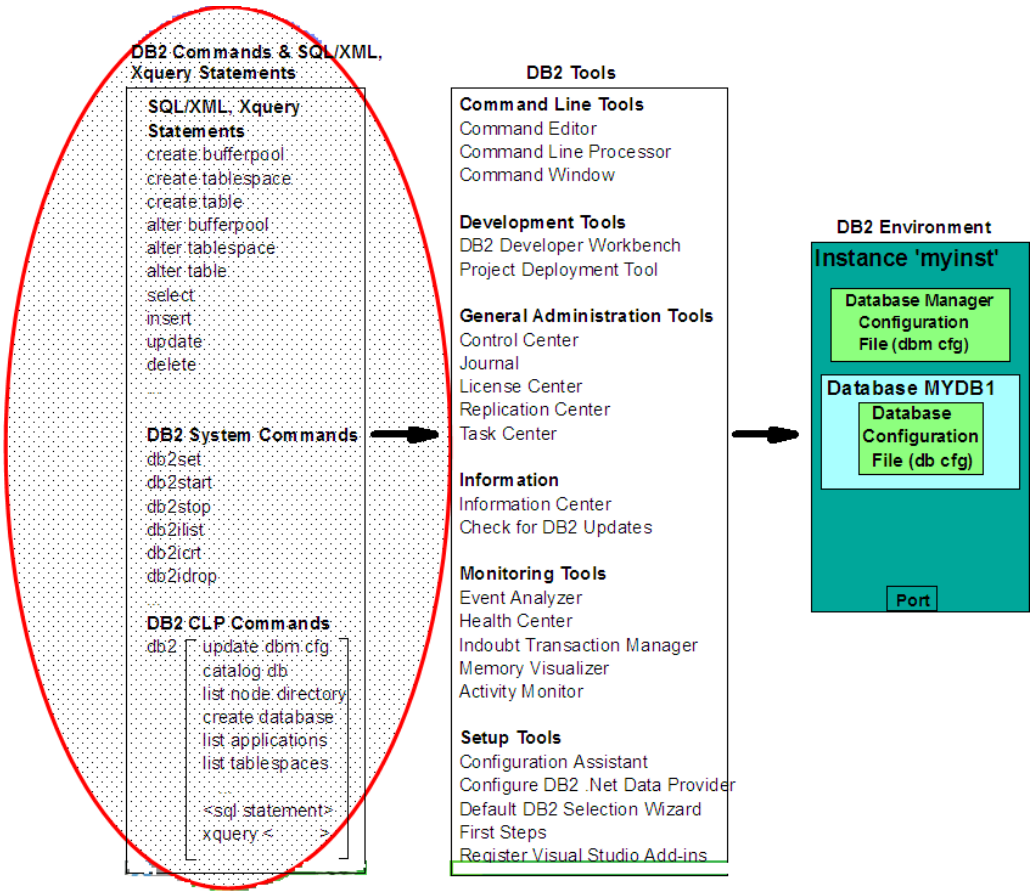


Figura 16.1 – The DB2 Big Picture: comandos DB2, SQL/XML e XQuery

16.1 Utilizar XML com base de dados

Os documentos XML podem ser guardados em ficheiros de texto, repositórios XML, ou base de dados. Existem duas razões principais para muitas empresas guardarem XML em base de dados:

- Gerir grandes quantidades de dados XML é uma tarefa para base de dados. XML são dados como qualquer outro tipo de dados, apenas num formato diferente. As mesmas razões para guardar dados relacionais em base de dados aplicam-se a dados XML: as base de dados fornecem procuras eficientes, suporte robusto para dados permanentes, *backup* e restauro, suporte de transacção, performance e escalabilidade.
- Integração: através do armazenamento de documentos XML e relacionais, pode integrar os novos dados XML com os dados relacionais existentes, e combinar SQL com Xpath ou Xquery numa só consulta. Além disso, dados relacionais podem ser publicados como XML, e vice versa. Através da integração, as base de dados podem melhorar o suporte a aplicações WEB, SOA, e *Web Services*.

16.2 Base de dados XML

Existem dois tipos de base de dados para guardar dados XML:

- Base de dados XML-enabled
- Base de dados de XML nativo

16.2.1 Base de dados XML-enabled

Uma base de dados XML-enabled utiliza um modelo relacional para o núcleo do modelo de armazenamento de dados. Isto requer uma organização entre o modelo de dados XML (hierárquico) e o modelo relacional de dados, ou então guardar dados XML como *character large object*. Este procedimento pode ser considerado uma tecnologia "antiga", mas ainda é utilizada por muitos "vendedores" de base de dados. A Figura 16.2 explica com mais detalhe as duas opções para as base de dados XML-enabled.

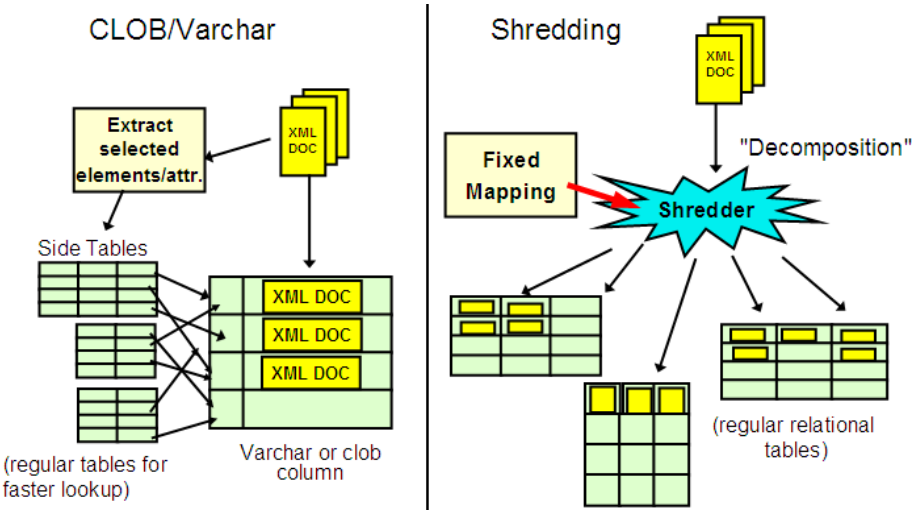


Figura 16.2 – As duas opções para guardar XML em base de dados XML-enabled

O lado esquerdo da Figura 16.2 mostra o metódo "CLOB e varchar" para guardar documentos XML na base de dados. Neste método, um documento XML é guardado como uma *unparsed string* numa coluna CLOB ou varchar na base de dados. Se o documento XML é guardado como uma *string*, quando quisermos aceder a uma parte do documento

XML, o programa irá procurar *string*, e realizar o *parsing* para encontrar o que queremos. Este método não é muito flexível.

A outra opção para base de dados XML-enabled chama-se de *shredding* ou decomposição e é ilustrada no lado direito da Figura 16.2. Através deste método, um documento XML inteiro é decomposto em partes mais pequenas que são guardadas em tabelas. Este método não é bom para flexibilidade: uma alteração no documento XML não é facilmente propagada nas tabelas correspondentes e mais tabelas poderão ter de ser criadas. Este método também não é eficiente: se precisarmos de construir o documento XML original, precisaremos de executar uma operação SQL dispendiosa, que ficará tanto mais *cara* quantas mais tabelas estiverem interligadas.

16.2.2 Base de dados XML nativo

Base de dados de XML nativo utilizam um modelo de dados XML hierárquico para guardar e processar o XML internamente. O formato de armazenamento é o mesmo do formato de processamento: não há mapeamento para o modelo relacional, e os documentos XML não guardados como imagens. Quando comandos Xpath ou Xquery são utilizados, eles são processados nativamente pelo motor, e não são convertidos para SQL. Esta é a razão pela qual estas bases de dados são conhecidas como base de dados XML "nativo". O DB2 após a versão 9 é actualmente o único servidor de dados comercial a fornecer este serviço.

16.3 XML em DB2

A Figura 16.3 em baixo mostra como os dados relacionais e hierárquicos (documentos XML) são guardados em DB2 a partir da versão 9. Na figura, assuma que a tabela dept é definida `dept` como:

```
CREATE TABLE dept (deptID CHAR(8),…, deptdoc XML);
```

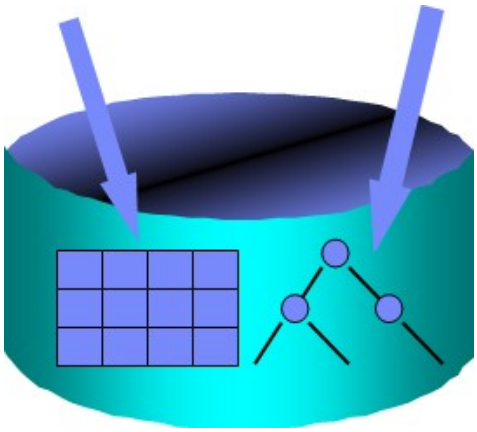


Figura 16.3 – XML em DB2

Note que a definição da tabela utiliza um novo tipo de dados, XML, para a coluna `deptdoc` . A seta esquerda na figura indica a coluna relacional `deptID` guardada no formato

relacional (tabelas), enquanto que a coluna XML `deptdoc` é guardada num formato hierquico segmentado.

A Figura 16.4 ilustra que a partir do DB2 9, existem quatro maneiras de aceder a dados:

- Utilizar SQL para aceder a dados relacionais
- Utilizar SQL com extensões XML para aceder a dados XML
- Utilizar XQuery para aceder a dados XML
- Utilizar XQuery para aceder a dados relacionais

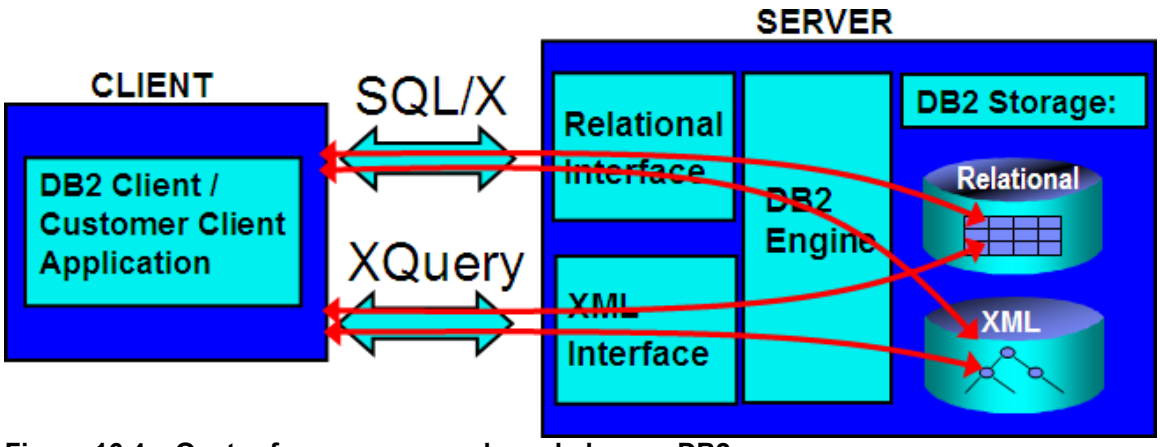


Figura 16.4 – Quatro formas para aceder a dados em DB2

Portanto, enquanto o utilizador SQL pode ver o DB2 como uma base de dados de primeira categoria que também suporta XML, uma pessoa XML verá o DB2 como repositório XML de primeira categoria que também suporta SQL.

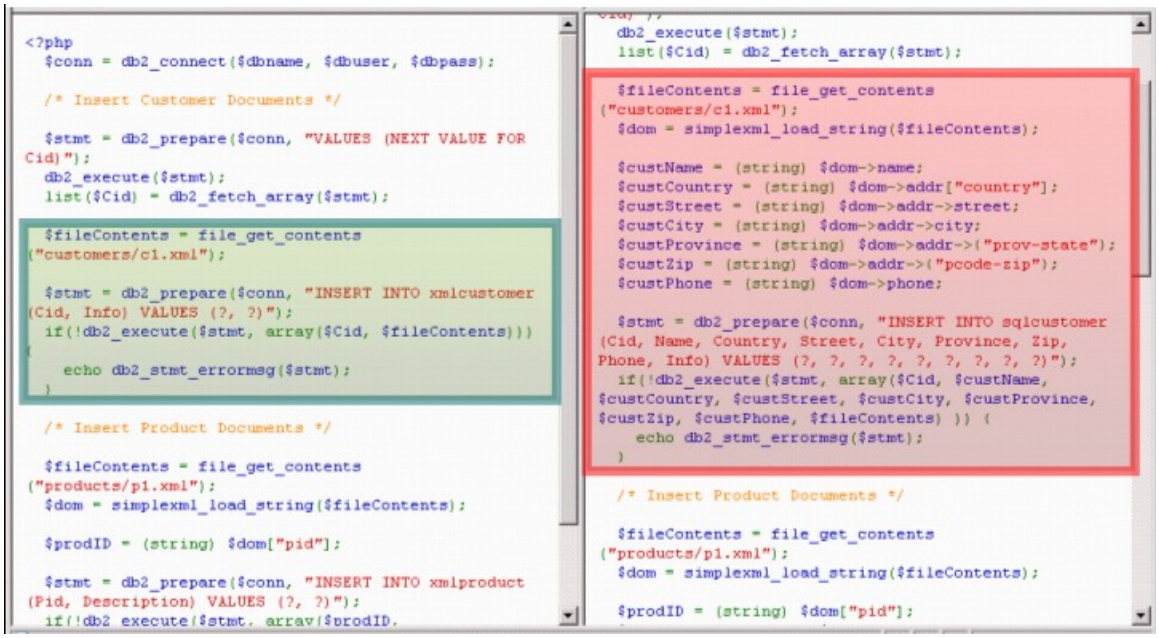
Note que a IBM utiliza o termo **pureXML** em vez de *XML nativo* (*native XML*) para descrever esta tecnologia. Enquanto outros vendedores ainda utilizam tecnologias antigas de CLOB/varchar ou decomposição para guardar documentos XML, eles chamam a essas tecnologias antigas de *native XML*. Para evitar confusão, a IBM decidiu usar o novo termo pureXML, e registar este nome para que mais nenhum vendedor de base de dados ou XML possa utilizar este termo para denotar outra tecnologia diferente.

16.3.1 Vantagens da tecnologia pureXML

Utilizar a tecnologia pureXML traz muitas vantagens.

1. Pode perfeitamente tirar partido do investimento num sistema relacional, se os documentos XML estiverem guardados em colunas de tabelas utilizando o novo tipo de dados XML.

2. Pode reduzir a complexidade do código. Por exemplo, a Figura 16.5 ilustra um script PHP escrito utizando pureXML e outro sem utilizar pureXML. Com pureXML (a caixa verde) as linhas de código são reduzidas. Isto não representa apenas que o código é menos complexo, mas a eficiência global é melhorada porque há menos linhas para processar e gerir neste código.



```
<?php
  $conn = db2_connect($dbname, $dbuser, $dbpass);

  /* Insert Customer Documents */

  $stmt = db2_prepare($conn, "VALUES (NEXT VALUE FOR
Cid)");
  db2_execute($stmt);
  list($Cid) = db2_fetch_array($stmt);

  $fileContents = file_get_contents
("customers/c1.xml");

  $stmt = db2_prepare($conn, "INSERT INTO xmlcustomer
(Cid, Info) VALUES (?, ?)");
  if(!db2_execute($stmt, array($Cid, $fileContents)))
{
    echo db2_stmt_errormsg($stmt);
  }

  /* Insert Product Documents */

  $fileContents = file_get_contents
("products/p1.xml");
  $dom = simplexml_load_string($fileContents);

  $prodID = (string) $dom["pid"];

  $stmt = db2_prepare($conn, "INSERT INTO xmlproduct
(Pid, Description) VALUES (?, ?)");
  if(!db2_execute($stmt, array($prodID,
```

```
  db2_execute($stmt);
  list($Cid) = db2_fetch_array($stmt);

  $fileContents = file_get_contents
("customers/c1.xml");
  $dom = simplexml_load_string($fileContents);

  $custName = (string) $dom->name;
  $custCountry = (string) $dom->addr["country"];
  $custStreet = (string) $dom->addr->street;
  $custCity = (string) $dom->addr->city;
  $custProvince = (string) $dom->addr->("prov-state");
  $custZip = (string) $dom->addr->("pcode-zip");
  $custPhone = (string) $dom->phone;

  $stmt = db2_prepare($conn, "INSERT INTO sqlcustomer
(Cid, Name, Country, Street, City, Province, Zip,
Phone, Info) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)");
  if(!db2_execute($stmt, array($Cid, $custName,
$custCountry, $custStreet, $custCity, $custProvince,
$custZip, $custPhone, $fileContents) )) {
    echo db2_stmt_errormsg($stmt);
  }

  /* Insert Product Documents */

  $fileContents = file_get_contents
("products/p1.xml");
  $dom = simplexml_load_string($fileContents);

  $prodID = (string) $dom["pid"];
```

Figura 16.5 – Complexidade de código com e sem pureXML

3. Alterações no esquema são mais facéis utilizando XML e a tecnologia pureXML. A Figura 16.6 ilustra um exemplo deste aumento de flexibilidade. Na figura, assuma que possui uma base de dados que consiste nas tabelas “Employee” e “Department”. Tipicamente, com uma base de dados não-XML, se o gestor lhe pedisse para gerir não apenas um número de telefone por empregado (telefone de casa), mas também um número de telefone secundário (número de telemóvel), poderia adicionar uma coluna extra na tabela “Employee” e guardar o número de telemóvel nessa coluna. No entanto, este método seria contra as regras de normalização das base de dados relacionais. Se quiser preservar estas regras, teria de criar uma nova tabela “Phone”, e mover toda a informação sobre os números de telefone para esta tabela. Poderia então adicionar os números de telemóvel para esta tabela. A criação desta nova tabela “Phone” ficaria cara, não apenas pela grande quantidade de informação pré-existente que precisaria de ser movida, mas também porque todo o SQL na aplicação teria de ser actualizado para apontar para esta nova tabela.

   Em vez disso, no lado esquerdo da figura, mostramos como poderiamos conseguir isso com XML. Se a empregada “Christine” também possuir um número de telemóvel, uma nova *tag* pode ser adicionada para inserir essa informação. Se o empregado “Michael” não possuir um número de telemóvel, não adicionamos nada.

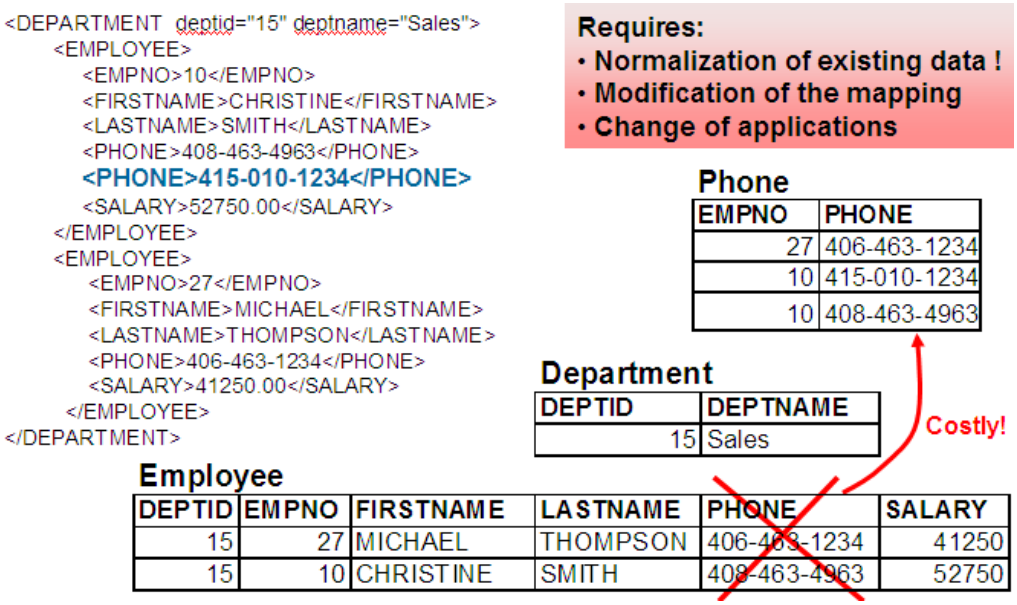


Figura 16.6 – Aumento de flexibilidade de dados utilizando XML

4. Pode melhorar a eficiência da sua aplicação XML. Testes de eficiência utilizando tecnologia pureXML mostraram grandes melhorias na eficiência de várias aplicações XML. A Figura 16.7 mostra os resultados dos testes para uma companhia que mudou de tecnologias antigas para pureXML. Os valores da coluna do meio são os resultados utilizando métodos antigos para trabalhar com XML, e a terceira coluna mostra os resultados utilizando pureXML.

Task	Other relational DB	DB2 with pureXML
Development of search and retrieval business processes	CLOB: 8 hrs Shred: 2 hrs	30 min.
Relative lines of I/O code	100	35 (65% reduction)
Add field to schema	1 week	5 min.
Queries	24 - 36 hrs	20 sec - 10 min

Figura 16.7 – Aumento de eficiência com a tecnologia pureXML

16.3.2 Conceitos Básicos de XPath

Xpath é uma linguagem que pode ser usada para fazer consultas a documentos XML. A Figura 16.8 mostra um exemplo de um documento XML, e a Figura 16.9 ilustra o mesmo documento representado num formato *parsed-hierarchical* (árvore hierárquica). Utilizaremos o formato *parsed-hierarchical* para explicar como o XPath funciona.

```
<dept bldg="101">
   <employee id="901">
        <name>John Doe</name>
            <phone>408 555 1212</phone>
            <office>344</office>
      </employee>
      <employee id="902">
            <name>Peter Pan</name>
            <phone>408 555 9918</phone>
            <office>216</office>
      </employee>
</dept>
```

Figura 16.8 – Exemplo de um documento XML

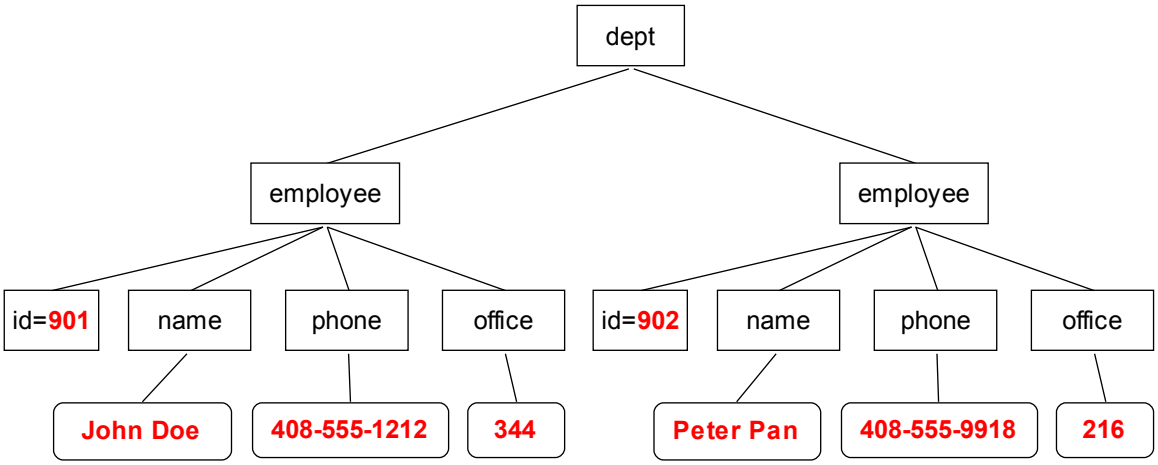


Figura 16.9 – Representação *parsed-hierarchical* do documento XML da Fig 16.8

Uma maneira rápida de aprender XPath é compará-lo ao comando `change directory (cd)` em MS-DOS ou Linux/UNIX. Com o `cd` comando pode fazer uma travessia na árvore de directorias como se segue:

```
cd /directory1/directory2/…
```

Similarmente, em XPath utilizam-se as barras para percorrer os vários elementos dentro do documento XML. Por exemplo, utilizando XPath com o documento na Figura 16.9 poderiamos obter o nome de todos os empregados com a seguinte expressão:

```
/dept/employee/name
```

Expressões XPath

Expressões XPath são caminhos completos para especificar elementos e/ou atributos. O sinal "@" é usado para especificar um atributo. Para obter apenas o valor (o texto do nodo) de um elemento, utilize a função *text()*. A Tabela 16.1 mostra consultas XPath e os resultados correspondentes utilizando o documento XML da Figura 16.9.

XPath	Resultado
/dept/@bldg	**101**
/dept/employee/@id	**901** **902**
/dept/employee/name	**<name>Peter Pan</name>** **<name>John Doe</name>**
/dept/employee/name/text()	**Peter Pan** **John Doe**

Tabela 16.1 – Exemplos de expressões XPath

***Wildcards* em XPath**
Existem dois *wildcards* principais em XPath:

- "*" corresponde a qualquer nome de *tag*
- "//" corresponde ao(s) nodo(s) "descendent-or-self"

A Tabela 16.2 fornece mais exemplos utilizando o documento XML da Figura 16.9

XPath	Resultado
/dept/employee/*/text()	**John Doe** **408 555 1212** **344** **Peter Pan** **408 555 9918** **216**
/dept/*/@id	**901** **902**
//name/text()	**Peter Pan** **John Doe**
/dept**//**phone	**<phone>408 555 1212</phone>** **<phone>408 555 9918</phone>**

Tabela 16.2 – Exemplos de *wildcards* em XPath

Predicados XPath
Predicados estão rodeados por parênteses rectos [ ]. Como analogia, podemos pensar neles como o equivalente à cláusula WHERE em SQL. Por exemplo [@id="902"] pode ser lida como: "WHERE atributo id é igual a 902". Podem haver mútiplos predicados numa só expressão XPath. Para especificar um predicado posicional, utilize [n] que significa o n-ésimo filho a ser seleccionado. Por exemplo, employee[2] significa que o segundo empregado deverá ser seleccionado. A Tabela 16.4 fornece mais exemplos.

XPath	Resultado
/dept/employee[@id="902"]/name	**<name>Peter Pan</name>**
/dept[@bldg="101"]/employee[office >"300"]/name	**<name>John Doe</name>**
//employee[office="344" OR office="216"]/@id	**901** **902**
/dept/employee[2]/@id	**902**

Tabela 16.3 – Exemplos de predicados XPath

XPath: o eixo *parent*
Similarmente ao MS-DOS ou Linux/UNIX, pode usar um "." (ponto" para indicar na expressão a que se está a referir ao contexto corrente, e ".." (ponto ponto) para se referir ao contexto do pai.
A Tabela 16.4 fornece mais exemplos.

XPath	Resultado
/dept/employee/name[../@id="902"]	**<name>Peter Pan</name>**
/dept/employee/office[.>"300"]	**<office>344</office>**
/dept/employee[office > "300"]/office	**<office>344</office>**
/dept/employee[name="John Doe"]/../@bldg	**101**
/dept/employee/name[.="John Doe"]/../../@bldg	**101**

Tabela 16.4 – Eixo *parent* do XPath

16.3.3 XQuery

Xquery é uma linguagem de consulta criada para XML. Xquery suporta expressões de caminhos para navegar na estrutura hierárquica do XML. De facto, XPath é um subconjunto de Xquery; portanto, tudo o que aprendemos anteriomente sobre XPath também se aplica a Xquery. Xquery suporta dados com tipo ou sem tipo. Em Xquery não existem valores nulos porque os documentos XML omitem dados desconhecidos ou em falta. Xquery retorna sequências de dados XML.
É importante notar que as expressões Xquery e XPath são *case sensitive*.

Xquery suporta a expressão FLWOR. Se usarmos SQL como analogia, é o equivalente à expressão SELECT-FROM-WHERE. A próxima secção descreve a FLWOR com mais detalhe.

XQuery: expressão FLWOR

FLWOR significa:

- FOR: itera sobre uma sequência, atribui uma variável aos elementos

- LET: atribui uma sequência a uma variável
- WHERE: foltra elementos da iteração
- ORDER: ordena os elementos da intereção
- RETURN: constroí os resultados da consulta

É uma expressão que permite a manipulação de documentos XML, retornando outra expressão. Por exemplo, assuma que possui uma tabela com esta definição

```
CREATE TABLE dept(deptID CHAR(8),deptdoc XML);
```

E o documento XML seguinte é inserido na coluna deptdoc:

```
<dept bldg=”101”>
  <employee id=”901”>
       <name>John Doe</name>
       <phone>408 555 1212</phone>
       <office>344</office>
   </employee>
   <employee id=”902”>
       <name>Peter Pan</name>
       <phone>408 555 9918</phone>
       <office>216</office>
    </employee>
</dept>
```

Então o seguinte comando Xquery utilizando a expressão FLWOR poderia ser executado:

```
xquery
for $d in db2-fn:xmlcolumn('dept.deptdoc')/dept
let $emp := $d//employee/name
where $d/@bldg > 95
order by $d/@bldg
return
   <EmpList>
   {$d/@bldg, $emp}
   </EmpList>
```

Que iria retornar o seguinte:

```
<EmpList bldg="101">
  <name>
      John Doe
  </name>
  <name>
      Peter Pan
  </name>
</EmpList>
```

16.3.4 Inserir documentos XML

Inserir documentos XML numa base de dados DB2 pode ser realizado com o comando INSERT SQL, ou a utilidade IMPORT. Xquery não pode ser utilizado com essa finalidade pois o processo de inserção ainda não está definido no standard.

Vamos examinar o seguinte *script*, que pode ser executado através do DB2 Command Windows ou da *shell* do Linux com este comando:

```
db2 -tvf table_creation.txt
```

table_creation.txt

```
-- (1)
drop database mydb
;

-- (2)
create database mydb using codeset UTF-8 territory US
;

-- (3)
connect to mydb
;

-- (4)
create table items (
  id          int primary key not null,
  brandname   varchar(30),
  itemname    varchar(30),
  sku         int,
  srp         decimal(7,2),
  comments    xml
);

-- (5)
create table clients(
  id          int primary key not null,
  name        varchar(50),
  status      varchar(10),
  contact     xml
);

-- (6)
insert into clients values (77, 'John Smith', 'Gold',
   '<addr>111 Main St., Dallas, TX, 00112</addr>')
;

-- (7)
IMPORT FROM "D:\Raul\clients.del" of del xml from "D:\Raul" INSERT
INTO CLIENTS (ID, NAME, STATUS, CONTACT)
```

```
;

-- (8)
IMPORT FROM "D:\Raul\items.del" of del xml from "D:\Raul" INSERT INTO
ITEMS (ID, BRANDNAME, ITEMNAME, SKU, SRP, COMMENTS)
;
```

Note que este script e os ficheiros relacionados são fornecidos no ficheiro zip `expressc_book_quicklabs.zip` que acompanham este livro. De seguida descrevemos cada linha do script.

1. Remove a base de dados "mydb". Isto é normalmente feito em ficheiros *script* criados para limpar a aplicação. Se "mydb" não existir, irá receber uma mensagem de erro, mas sem consequências.
2. Cria a base de dados "mydb" utilizando o codeset UTF-8. Uma base de dados UNICODE é necessária para suportar pureXML na versão DB2 9.1, por isso este passo é necessário para criar a base de dados como sendo UNICODE.
3. Liga-se à base de dados recém-criada "mydb". Isto é necessário para criar objectos dentro da base de dados.
4. Cria a tabela "items". Note que a última coluna da tabela (a coluna "comments") está definida como uma coluna XML com o novo tipo de dados XML.
5. Cria a tabela "clients". Note que a última coluna da tabela (a coluna "contact") também está definida como sendo do tipo XML.
6. Utilizando o comando SQL INSERT, podemos inserir um documento XML na coluna XML. No comando INSERT passamos o documento XML como uma *string* rodeada de plicas.
7. Utilizando o comando IMPORT, pode inserir ou importar vários documentos XML assim como dados relacionais para a base de dados. Em (7) está a importar os dados do ficheiro clients.del (um ficheiro delimited ascii), e também a indicar onde os dados XML referenciados pelo ficheiro clients.del está localizado (neste exemplo, em D:\Raul).

Iremos olhar com mais cuidado para o ficheiro clients.del, mas antes, veremos o conteúdo da directória D:\Raul (Figura 16.10).

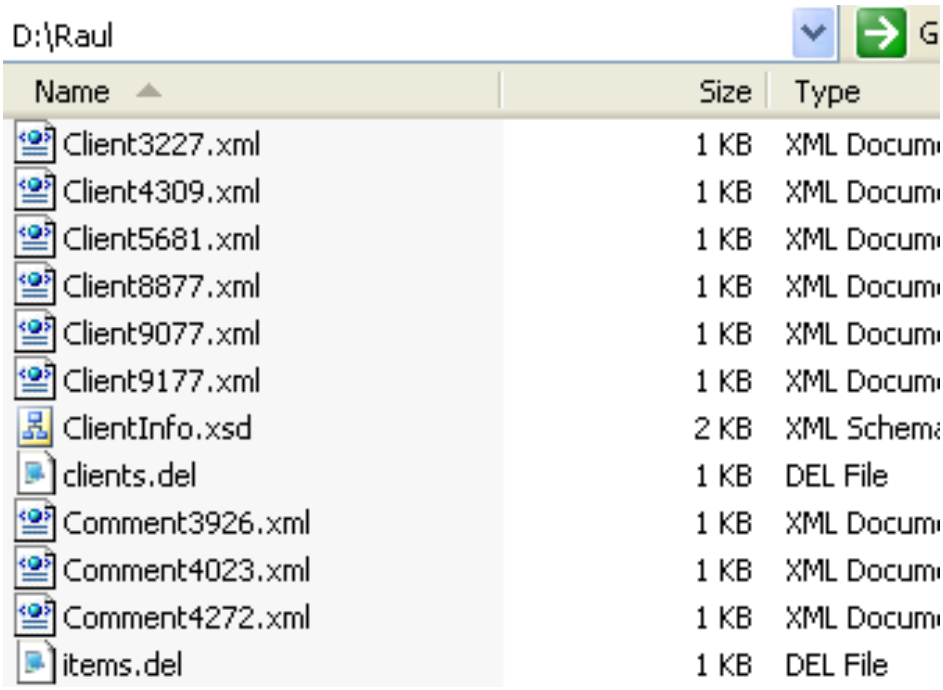


Figura 16.10 – Conteúdo da directória D:\Raul com os documentos XML

Este é o conteúdo do ficheiro clients.del

clients.del

```
3227,Ella Kimpton,Gold,<XDS FIL='Client3227.xml' />,
8877,Chris Bontempo,Gold,<XDS FIL='Client8877.xml'/>,
9077,Lisa Hansen,Silver,<XDS FIL='Client9077.xml' />
9177,Rita Gomez,Standard,<XDS FIL='Client9177.xml'/>,
5681,Paula Lipenski,Standard,<XDS FIL='Client5681.xml' />,
4309,Tina Wang,Standard,<XDS FIL='Client4309.xml'/>
```

No ficheiro clients.del, "`XDS FIL=`" é utilizado para apontar para um ficheiro XML específico.

A Figura 16.11 mostra o Control Center depois da execução do script anterior.

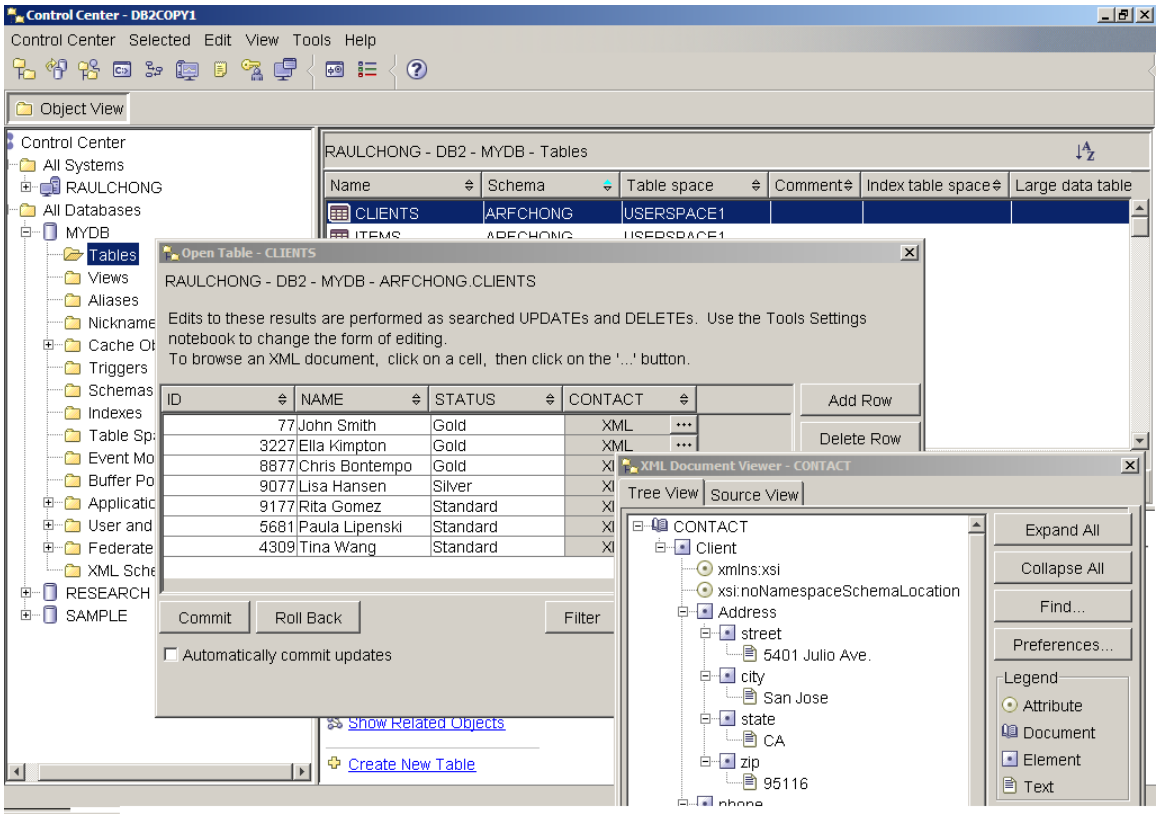


Figura 16.11 – O Control Center após a execução de table_creation.txt

Note que na figura, mostramos o conteúdo da tabela CLIENTS. A última coluna "Contact" é uma coluna XML. Quando clica no botão com três pontos, aparecerá outra janela com o conteúdo do documento XML. Isto é mostrado no canto inferior direito da figura 16.11.

16.3.5 Consultar dados XML

Existem duas formas para consultar dados XML em DB2:

- Utilizar SQL com extensões XML (SQL/XML)
- Utilizar XQuery

Em ambos os casos, o DB2 segue os standards internacionais XML.

Consultar dados XML com SQL/XML
Utilizando comandos SQL podemos trabalhar com linhas e colunas. Um comando SQL pode ser usado para trabalhar com documentos XML completos; no entanto, não ajuda quando tentamos obter apenas uma parte do documento. Nesses casos, teremos de usar SQL com extensões XML (SQL/XML).

A Tabela 16.5 descreve algumas das funções SQL/XML disponíveis com o standard SQL 2006

Nome da função	Descrição
XMLPARSE	Faz *parsing* de uma *string* ou objecto binário e produz um documento XML
XMLSERIALIZE	Converte um XML em *string* ou objecto binário
XMLVALIDATE	Valida um valor XML com base num XML Schema e anota os elementos com informação sobre o tipo de dados
XMLEXISTS	Determina se uma XQuery retorna um resultado (i.e. uma sequência de um ou mais items)
XMLQUERY	Executa uma XQuery e retorna o resultado
XMLTABLE	Executa uma Xquery, retorna a sequência do resultado como uma tabela relacional (se possível)
XMLCAST	Converte valores de e para o tipo XML

Tabela 16.5 – Funções SQL/XML

Os seguintes exemplos podem ser testados com a base de dados "mydb" criada anteriormente.

Exemplo 1
Este é um exemplo de um problema de consulta simples. Imagine que precisa de localizar o nome de todos os clientes que vivem num código postal específico. A tabela "clients" armazena os endereços dos clientes incluindo os códigos postais, numa coluna XML. Utilizando a função XMLEXISTS, pode procurar numa coluna XML pelo código postal pretendido e restringir o conjunto de resultados.

```
SELECT name FROM clients
 WHERE xmlexists(
     '$c/Client/Address[zip="95116"]'
     passing clients.contact as "c"
     )
```

A primeira linha é uma cláusula SQL que especifíca que quer retornar a informação na coluna "name" da tabela "clients".

A cláusula WHERE invoca a função XMLEXISTS, especificando a expressão XPath que faz com que o DB2 navegue até ao elemento "zip" e procure pelo número 95116

A expressão "$c/Client/Address" indica o caminho dentro da hierarquia do documento XML onde o DB2 pode localizar o elemento "zip". O simbolo dolar ($) é utilizado para especificar a variável; portanto "c" é uma variável. Esta variável é então definida por esta linha: `passing clients.contact as "c"`. Neste caso, "clients" é o nome da tabela e "contact" é o nome da coluna do tipo de dados XML. Estamos portanto a passar o documento XML para a variável "c".

O DB2 inspecciona os dados XML contidos na coluna "contact", navega da raíz "Client" para o nodo "Address" e depois para o nodo "zip" e finalmente determina se o cliente vive no código postal específico. A função XMLEXISTS retorna "verdadeiro" - "true" - e o DB2 retorna o nome do cliente associado a essa linha.

Com DB2 9.5, a consulta anterior pode ser simplificada pela seguinte:

```
SELECT name FROM clients
 WHERE xmlexists(
     '$CONTACT/Client/Address[zip="95116"]'
     )
```

Uma variável com o mesmo nome de uma coluna XML é criada automaticamente pelo DB2. No exemplo acima, a variável CONTACT é criada automaticamente pelo DB2. O seu nome corresponde ao nome da coluna XML CONTACT.

Exemplo 2
Vamos agora resolver o problema de como criar uma lista com os endereços de e-mail dos clientes com o estatuto "Gold". A seguinte *query* poderia ser executada:

```
SELECT xmlquery('$c/Client/email' passing contact as "c")
  FROM  clients
  WHERE status = 'Gold'
```

A primeira linha indica que queremos retornar o endereço de email que é um elemento de um documento XML (e não uma coluna relacional). Como no exemplo anterior, "$c" é uma variável que contém o documento XML. Neste exemplo utilizamos a função XMLQUERY que pode ser utilizada depois de um SELECT, enquanto que a função XMLEXISTS pode ser utilizada depois de uma cláusula WHERE.

Exemplo 3
Pode haver situações em que queremos apresentar dados XML como tabelas. Isto é possível com a função XMLTABLE como o exemplo seguinte mostra.

```
SELECT t.comment#, i.itemname, t.customerID, Message
  FROM items i,
  xmltable('$c/Comments/Comment' passing i.comments as "c"
      columns Comment# integer path 'CommentID',
      CustomerID integer path 'CustomerID',
      Message varchar(100) path 'Message') AS t
```

A primeira linha especifíca as colunas que devem ser incluídas no conjunto de resultados. As colunas prefixadas com a variável "t" são baseadas em valores de elementos XML.

A segunda linha invoca a função XMLTABLE que especifíca a coluna DB2 XML que contém os dados pretendidos ("i.comments") e o caminho entre as colunas XML onde os elementos de interesse estão localizados.

A cláusula "columns", entra as linhas 4 e 6, indifica os elementos XML específicos que iram ser mapeados para as colunas de saída no conjunto de resultados SQL especificado na linha 1. Parte deste mapeamento involve especificar os tipos de dados para os quais os valores dos elementos XML iram ser convertidos. Neste exemplo todos os dados XML irão ser convertidos para tipos de dados SQL tradicionais.

Exemplo 4

Vamos agora explorar um exemplo simples no qual incluiremos uma expressão XQuery FLWOR dentro de uma função SQL/XML XMLQUERY

```
SELECT name, xmlquery(
    'for $e in $c/Client/email[1] return $e'
    passing contact as "c"
    )
  FROM clients
  WHERE status = 'Gold'
```

A primeira linha especifíca que os nomes dos clientes e o output da função XMLQUERY estarão incluídos no conjunto de resultados. A segunda linha indica que o primeiro sub-elemento "email" do elemento "Client" irá ser retornado. A terceira linha identifica a fonte dos nossos dados XML (coluna "contact". A quarta linha diz-nos que esta coluna é proveniente da tabela "clients"; e a quinta linha indica que apenas estamos interessados nos clientes "Gold".

Exemplo 5
Este exemplo demonstra novamente uma função XMLQUERY que utiliza uma expressão Xquery FLWOR, mas note que desta vez não estamos a retornar apenas XML, mas também HTML.

```
SELECT xmlquery('for $e in $c/Client/email[1]/text()
    return <p>{$e}</p>'
    passing contact as "c")
  FROM clients
  WHERE status = 'Gold'
```

A cláusula de retorno da Xquery permite-nos transformar o output XML da forma que desejar. Utilizando a função *text()* na primeira linha indicamos que apenas a representação textual do primeiro endereço de email dos clientes qualificados nos interessa. A segunda linha especifíca que esta informação deverá ser associada a tags de paragrafos HTML.

Exemplo 6
O exemplo seguinte utiliza a função XMLELEMENT para criar uma série de elementos, em que cada um contém sub-elementos para os valores de "id", "brandname" e "sku" obtidos pelas colunas correspondentes na tabela "items". Basicamente, pode usar a função XMLELEMENT quando quer converter dados relacionais para dados XML.

```
SELECT
  xmlelement (name "item", itemname),
  xmlelement (name "id", id),
  xmlelement (name "brand", brandname),
  xmlelement (name "sku", sku)
  FROM items
  WHERE srp < 100
```

A consulta anterior deverá retornar o seguinte output:

```
<item>
```

```
      <id>4272</id>
      <brand>Classy</brand>
      <sku>981140</sku>
</item>
…
<item>
      <id>1193</id>
      <brand>Natural</brand
      <sku>557813</sku>
</item>
```

Consultar dados XML com XQuery

Na secção anterior, analisámos como consultar dados XML utilizando SQL com extensões XML. SQL é sempre o primeiro método de consulta, e XPath foi embebido dentro de SQL. Nesta secção, iremos discutir como consultar dados XML com Xquery. Desta vez, utilizaremos Xquery como um método primário de consulta, e em alguns casos, utilizaremos SQL embebido em Xquery (através da função "db2-fn:sqlquery"). Ao utilizar Xquery, iremos invocar algumas funções, e também utilizeremos a expressão FLWOR.

Exemplo 1
Uma simples Xquery para retornar os dados de contacto de um cliente

```
xquery db2-fn:xmlcolumn('CLIENTS.CONTACT')
```

Assinale sempre todas expressões Xquery com o comando "xquery" para que o DB2 saiba que deverá usar o parser Xquery, de outra forma o DB2 irá assumir que está a tentar executar uma expressão SQL. A função **db2-fn:xmlcolumn** é uma função que obtém os documentos XML da coluna especificada como parâmetro. Esta função é equivalente ao seguinte comando SQL, que obtém o contéudo de colunas inteiras:

```
SELECT contact FROM clients
```

Exemplo 2
Neste exemplo, utilizaremos a expressão FLWOR para retornar os dados de fax do cliente

```
xquery
  for $y in db2-fn:xmlcolumn('CLIENTS.CONTACT')/Client/fax
  return $y
```

A primeira linha invoca o *parser* Xquery. A segunda linha indica ao DB2 para iterar através dos sub-elementos fax contidos na coluna CLIENTS.CONTACT. Cada elemento fax é atribuído à variável $y. A terceira linha indica que para cada iteração, o valor "$y" é retornado.

O output desta consulta é semelhante a este (pode incluir o *namespace* por default, mas não o mostramos em baixo, porque de outra forma este output seria mais dificil de ler já que pode abranger várias linhas):

```
<fax>4081112222</fax>
<fax>5559998888</fax>
```

Exemplo 3
O próximo exemplo consulta dados XML e retorna os resultados como HTML.

```
xquery
  <ul> {
    for $y in db2-fn:xmlcolumn('CLIENTS.CONTACT')/Client/Address
    order by $y/zip
    return <li>{$y}</li>
  }
  </ul>
```

O HTML retornado será parecido com:

```
<ul>
<li>
<address>
    <street>9407 Los Gatos Blvd.</street>
    <city>Los Gatos</city>
    <state>ca</state>
    <zip>95302</zip>
</address>
</li>
<address>
<street>4209 El Camino Real</street>
    <city>Mountain View</city>
    <state>CA</state>
    <zip>95302</zip>
</address>
</li>
...
</ul>
```

Exemplo 4
O exemplo seguinte mostra como embeber SQL dentro de Xquery através da função *db2-fn:sqlquery*. A função *db2-fn:sqlquery* executa uma consulta SQL e retorna apenas os dados XML seleccionados. A consulta SQL passada para *db2-fn:sqlquery* deve apenas retornar dados XML. Estes dados XML podem ser depois processados por XQuery

```
xquery
  for $y in
  db2-fn:sqlquery(
    'select comments from items where srp > 100'
  )/Comments/Comment
  where $y/ResponseRequested='Yes'
  return (
    <action>
      {$y/ProductID
       $y/CustomerID
       $y/Message}
```

```
    </action>
  )
```

Neste exemplo, a consulta SQL filtra linhas baseada na condição de que a coluna "srp" possui um valor maior do que 100. Destas linhas filtradas, irá seleccionar a coluna "comments" que é uma coluna XML. De seguida Xquery (ou XPath) é aplicada para ir para os sub-elementos.

Nota: O DB2 é *case insensitive* e trata todos os nome de tabelas e colunas em maiúsculas, enquanto Xquery é *case sensitive*. As funções utilizadas anteriomente são funções interface Xquery e portanto todos os nomes de tabelas e colunas deverão ser passados para estas funções em maiúsculas. Passar o nome dos objectos em minúsculas pode resultar num erro de nome de objecto indefinido.

16.3.6 Joins com SQL/XML

Esta secção descreve como realizar operações JOIN entre duas colunas XML de diferentes tabelas, ou entre uma coluna XML e uma coluna relacional. Assuma que criou duas tabelas com estes comandos:

```
CREATE TABLE dept (unitID CHAR(8), deptdoc XML)

CREATE TABLE unit (unitID  CHAR(8) primary key not null,
                   name    CHAR(20),
                   manager VARCHAR(20),
                   ...
                   )
```

Pode realizar uma operação JOIN de duas maneiras:

Metodo 1:

```
SELECT u.unitID
  FROM dept d, unit u
  WHERE XMLEXISTS (
    '$e//employee[name = $m]'
    passing d.deptdoc as "e", u.manager as "m")
```

A linha 3 deste comando mostra que a operação JOIN ocorre entre o elemento "name" que é um sub-elemento da coluna XML "deptdoc" na tabela "dept", e a coluna relacional "manager" na tabela "unit".

Metodo 2:

```
SELECT u.unitID
  FROM dept d, unit u
  WHERE u.manager = XMLCAST(
    XMLQUERY('$e//employee/name '
      passing d.deptdoc as "e")
    AS char(20))
```

Neste método alternativo, a coluna relacional está no lado esquerdo do JOIN. Se a coluna relacional está no lado esquerdo da equação, umíndice relacional pode ser usado em vez de um índice XML.

16.3.7 Joins com XQuery

Assuma que as seguintes tabelas foram criadas:

```
CREATE TABLE dept(unitID CHAR(8), deptdoc XML)
CREATE TABLE project(projectDoc XML)
```

Se utilizarmos SQL/XML, a consulta ficaria parecida com:

```
SELECT XMLQUERY (
  '$d/dept/employee' passing d.deptdoc as "d")
  FROM dept d, project p
  WHERE XMLEXISTS (
    '$e/dept[@deptID=$p/project/deptID]'
      passing d.deptdoc as "e", p.projectDoc as "p")
```

A consulta equivalente utilizando Xquery seria:

```
xquery
  for $dept in db2-fn:xmlcolumn("DEPT.DEPTDOC")/dept
  for $proj in db2-fn:xmlcolumn("PROJECT.PROJECTDOC")/project
  where $dept/@deptID = $proj/deptID
  return $dept/employee
```

Este segundo método é mais fácil de interpretar – a variável "$dept" está associada ao documento XML da coluna XML "deptdoc" na tabela "dept". A variável "$proj" está associada ao documento XML da coluna "projectdoc" na tabela "project". A linha 4 realiza a operação JOIN entre um atributo do primeiro documento XML e um elemento do segundo documento XML.

16.3.8 Operações de actualização e remoção

Operações de actualização e remoção em dados XML podem ser realizados numa de duas formas:

- Utilizando comandos SQL UPDATE e DELETE
- Utilizando a expressão TRANSFORM do XQuery

Para primeira forma utilizando os comandos SQL UPDATE e DELETE, a actualização ou remoção ocorre ao nível do documento; isto é, o documento XML inteiro é substituído pelo actualizado. Por exemplo, se no exemplo seguinte apenas o que quisessemos fosse alterar o elemento <state>, o documento XML inteiro seria na realidade substituído.

```
UPDATE clients SET contact=(
  xmlparse(document
    '<Client>
```

```
      <address>
        <street>5401 Julio ave.</street>
        <city>San Jose</city>
        <state>CA</state>
        <zip>95116</zip>
      </address>
      <phone>
        <work>4084633000</work>
        <home>4081111111</home>
        <cell>4082222222</cell>
      </phone>
      <fax>4087776666</fax>
      <email>newemail@someplace.com</email>
    </Client>')
  )
  WHERE id = 3227
```

Para o segundo método, podemos realizar actualizações a sub-documentos utilizando a expressão TRANSFORM, que é muito mais eficiente. Esta expressão permite-nos substituir, inserir, remover ou renomear nodos num documento XML. Podemos também alterar o valor de um nodo sem substituir o nodo, tipicamente mudar o valor de um elemento ou atributo, que é um tipo muito comum de actualização. Isto é novo com o DB2 9.5.

A expressão TRANSFORM é parte da linguagem Xquery. Pode utilizá-la em qualquer lugar em que normalmente utilize Xquery, por exemplo numa expressão FLWOR ou numa função XMLQUERY num comando SQL/XML. A utilização mais usual é num comando SQL UPDATE para modificar um documento XML numa coluna XML.

Eis a sintaxe da expressão TRANSFORM:

```
>>-transform--| copy clause |--| modify clause |--| return clause
|-><

copy clause

         .-,---------------------------------------.
         V                                         |
|--copy----$VariableName--:=--CopySourceExpression-+------------|

modify clause

|--modify--ModifyExpression-------------------------------------|

return clause

|--return--ReturnExpression-------------------------------------|
```

O comando `copy` é utilizado para atribuir a uma variável os documentos XML que queremos processar. Com o comando `modify`, podemos invocar uma expressão `insert`, `de-`

`lete`, `rename`, ou `replace`. Estas expressões permitem actualizar o documento XML. Por exemplo, se quisermos adicionar novos nodos a um documento, utilizamos a expressão `insert`, para remover nodos de um documento XML, a expressão `delete`, para renomear um elemento ou atributo num documento XML, a expressão `rename`, e para substituir um nodo existente por um novo nodo ou sequência de nodos, utilizamos a expressão `replace`. O valor da substituição pode ser usado apenas para alterar o valor de um elemento ou atributo. A cláusula return retorna o resultado da expressão transform.

Eis um exemplo de um comando UPDATE utilizando a expressão TRANSFORM.

```
(1)-- UPDATE customers
(2)-- SET contactinfo = xmlquery( 'declare default element namespace
(3)--                              "http://posample.org";
(4)--    transform
(5)--    copy $newinfo := $c
(6)--          modify do insert <email2>my2email.gm.com</email2>
(7)--                    as last into $newinfo/customerinfo
(8)--    return $newinfo' passing contactinfo as "c")
(9)-- WHERE id = 100
```

No exemplo anterior, as linhas (1), (2) e (9) são parte da sintaxe SQL UPDATE. Na linha (2) a função XMLQUERY é invocada, que chama a expressão transform na linha (4). O bloco da expressão transform vai desde a linha (4) à linha (8), e é utilizada para inserir um novo nodo num documento XML que contém o elemento email. Note que a actualização de elementos num documento XML através de uma vista não é suportado.

Remover documentos XML inteiros de tabelas é tão simples como usar o comando SELECT em SQL/XML. Utilize o comando SQL DELETE e especifique qualquer precidicado WHERE que precisar.

16.3.9 Indexar documentos XML

Num documento XML, índices podem ser criados para elementos, atributos, ou para valores (nodos de texto). Seguem-se alguns exemplos. Assuma que a seguinte tabela foi criada:

```
CREATE TABLE customer(info XML)
```

E assuma que possui um documento XML armazenado:

```
<customerinfo Cid="1004">
  <name>Matt Foreman</name>
  <addr country="Canada">
    <street>1596 Baseline</street>
    <city>Toronto</city>
    <state>Ontario</state>
    <pcode>M3Z-5H9</pcode>
  </addr>
  <phone type="work">905-555-4789</phone>
  <phone type="home">416-555-3376</phone>
  <assistant>
```

```
    <name>Peter Smith</name>
    <phone type="home">416-555-3426</phone>
  </assistant>
</customerinfo>
```

1) Este comando cria um índice sobre o atributo “Cid”

```
CREATE UNIQUE INDEX idx1 ON customer(info)
  GENERATE KEY USING
  xmlpattern '/customerinfo/@Cid'
  AS sql DOUBLE
```

2) Este comando cria um índice sobre o elemento “name”

```
CREATE INDEX idx2 ON customer(info)
  GENERATE KEY USING
  xmlpattern '/customerinfo/name'
  AS sql VARCHAR(40)
```

3) Este comando cria um índice sobre todos os elementos “name”

```
CREATE INDEX idx3 ON customer(info)
  GENERATE KEY USING
  xmlpattern '//name'
  AS sql VARCHAR(40);
```

4) Este comando cria um índice sobre todos os nodos de texto (todos os valores). Este procedimento não é recomendado, porque é muito “caro” manter o índice para operações de actualização, remoção e inserção, e o índice será demasiado grande.

```
CREATE INDEX idx4 ON customer(info)
  GENERATE KEY USING
  xmlpattern '//text()'
  AS sql VARCHAR(40);
```

QuickLab #12 - SQL/XML e XQuery

Objectivo

Neste momento já vimos vários exemplos de sintaxe SQL/XML e Xquery e já foi apresentado ao DB2 Command Editor e ao IBM Data Studio. Neste Quicklab, irá testar o seu conhecimento sobre SQL/XML e Xquery enquanto ganha experiência com estas ferramentas. Iremos utilizar a base de dados “mydb” criada com o ficheiro script **table_creation.txt** que já foi explicado neste capítulo.

Procedimento

- Crie a base de dados “mydb” e carregue os dados XML.

- Utilizando o Command Editor ou o IBM Data Studio Developer Workbench:
  1. Obtenha todos os documentos XML comments da tabela ITEMS de duas formas, mas apenas usando Xquery
  2. Porque é que este comando SQL não retorna exactamente o mesmo output?:
     ```
     SELECT comments FROM items
     ```
  3. Obtenha o ID e o BRANDNAME para os registos nos documentos XML em que o elemento ResponseRequested possui o valor “No”

SOLUÇÕES:

2a)

```
xquery db2-fn:xmlcolumn('ITEMS.COMMENTS')
xquery db2-fn:sqlquery("select comments from
items")
```

2b)

O output é diferente porque o SQL retorna valores NULL quando o valor não está presente, enquanto Xquery não retorna nada.

2c)

SELECT id, brandname FROM items WHERE XMLEXISTS('$c/Comments/Comment[ResponseRequested="No"]' `passing ITEMS.COMMENTS as "c"`

17

Capítulo 17 – Desenvolvimento em Java, PHP e Ruby

Este capítulo discute os conceitos básicos de desenvolvimento de aplicações em Java, PHP e Ruby on Rails utilizando um servidor DB2. O objectivo deste capítulo não é ensinar estas linguagens, mas fornecer informações pertinentes para usá-las com o DB2.

Nota:

Para mais informações sobre este tópico, veja este video:
http://www.channeldb2.com/video/video/show?id=807741:Video:4402

17.1 Desenvolvimento de aplicações em Java

O *driver* IBM DB2 para JDBC (também conhecido como o *driver* JCC) foi optimizado para todos os servidores DB2, em todas as plataformas. O ficheiro jar db2jcc.jar (com.ibm.db2.jcc) inclui os *drivers* de tipo 2 e tipo 4. O ficheiro db2jcc.jar está incluído em qualquer cliente DB2, ou pode ser obtido (IBM DB2 Driver para JDBC e SQLJ), a partir do site DB2 Express-C(ibm.com/db2/express)

17.1.1 *Driver* JDBC Tipo 2

O *driver* JDBC tipo 2 necessita de um cliente DB2 para ser instalado onde a aplicação JDBC está a funcionar. A figura 17.1 ilustra uma aplicação JDBC utilizando o *driver* de tipo 2.

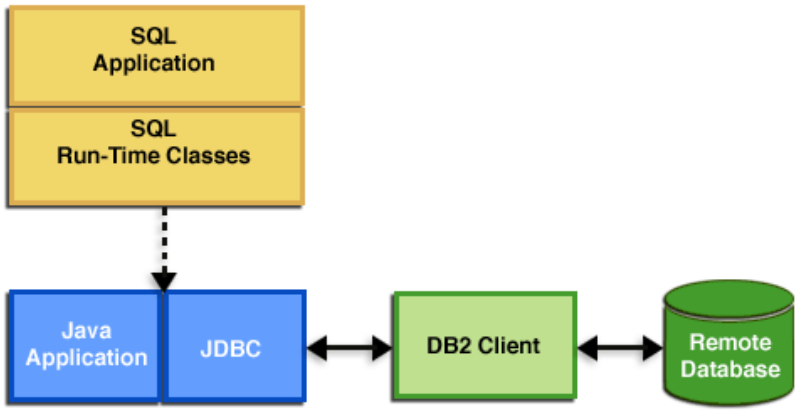


Figura 17.1 – O *driver* JDBC tipo 2

A figura 17.2 mostra um excerto de código que explica como estabelecer uma conexão utilizando o *driver* JDBC Tipo 2. Note que a URL não inclui informação de *host* ou porto porque este é retirado do cliente DB2.

```
...
  public static final String DB_URL = "jdbc:db2:sample";
  Properties connectProperties = new Properties();
  connectProperties.put("user", "db2admin");
  connectProperties.put("password", "ibmdb2");
  Connection connection = null
  try
  {
     Class.forName("com.ibm.db2.jcc.DB2Driver").newInstance();
     connection = DriverManager.getConnection(url, connectProperties)
  }
  catch (Exception e)
  throw e;
  }
...
```

Figura 17.2 – Estabelecer uma ligação utilizando o *driver* JDBC tipo 2

17.1.2 *Driver* JDBC Tipo 4

O *driver* JDBC tipo 4 não requer um cliente DB2 para se ligar a um servidor DB2. A figura 17.3 ilustra uma aplicação utilizando o *driver* JDBC tipo 4.

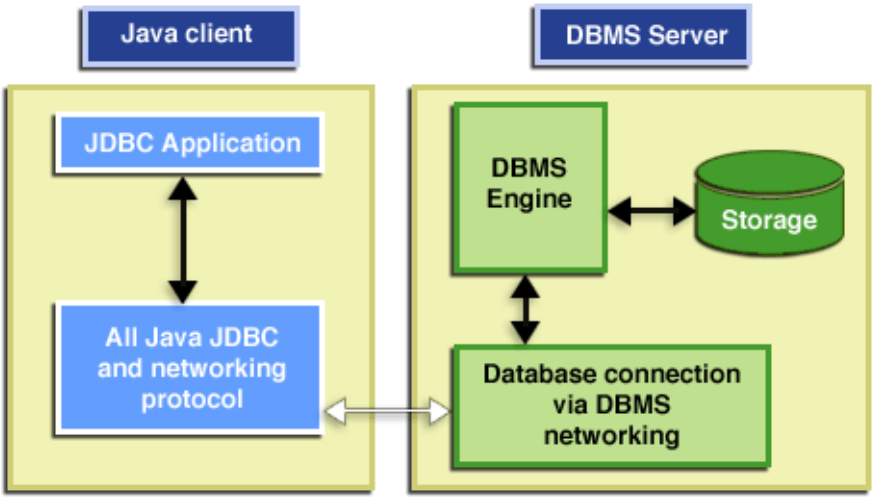


Figura 17.3 – O *driver* JDBC tipo 4

A figura 17.4 mostra um excerto de código que explica como estabelecer uma conexão utilizando o *driver* JDBC Tipo 4. Note que o URL inclui o nome do *host* ou porto.

```
...
  public static final String DB_URL = "jdbc:db2://server1:50000/sample";
  Properties connectProperties = new Properties();
  connectProperties.put("user", "db2admin");
  connectProperties.put("password", "ibmdb2");
  Connection connection = null
  try
  {
     Class.forName("com.ibm.db2.jcc.DB2Driver").newInstance();
     connection = DriverManager.getConnection(url,connectProperties)
  }
  catch (Exception e)
    throw e;
  }
...
```

Figura 17.4 – Estabelecer uma ligação utilizando o *driver* JDBC tipo 4

17.2 Desenvolvimento de aplicações em PHP

PHP (PHP Hypertext Preprocessor) é uma plataforma independente de linguagem, *open source*, que é adequada para o desenvolvimento de aplicações web. É uma das linguagens mais utilizada na web actualmente. A popularidade do PHP é baseada nas seguintes características da linguagem:

- Rápida, ciclos iterativos de desenvolvimento com uma baixa curva de aprendizagem.
- Robusta, de alta performance e escalável;
- Estável e segura.
- Uma alternativa ao J2EE™ e ao .NET na Web.
- Fácil de integrar com sistemas/ambientes heterógeneos
- Comprovada mediante a implantação generalizada
- Bem estabelecida numa vibrante comunidade

PHP faz parte do pacote LAMP (Linux, Apache HTTP Server, MySQL, **PHP**/Perl /Python). Esta é uma tecnologia web *open source*, muitas vezes disponível nos ISPs por um razoável custo mensal.

17.2.1 Opções de conecção DB2 para PHP

A IBM suporta o acesso a uma base de dados DB2 a partir de uma aplicação PHP através de duas extensões.

ibm_db2:
A extensão ibm_db2 oferece uma interface de programação de aplicações procedimental para criar, ler, escrever e actualizar operações de bases de dados para além do acesso extensivo à base de dados. Pode ser compilado para trabalhar com PHP 4 ou 5. A extensão está disponível a partir do repositório PECL sob a licença Apache 2.0. A extensão foi desenvolvida e é mantida pela IBM. O ibm_db tem todo o suporte para procedimentos armazenados e LOBs, é rápido, pois foi optimizado para o DB2.

PDO_ODBC:
O PDO_ODBC é um *driver* para Objectos de Dados do PHP (PHP Data Objects - PDO), e oferece acesso a bases de dados DB2 através de uma interface de base de dados standard orientada aos objectos introduzidos no PHP 5.1. Pode ser compilado directamente com bibliotecas DB2 e oferece uma interface padrão de acesso aos dados em PHP. É rápido, leve, e orientado aos objectos. A extensão PDO_ODBC utiliza bibliotecas DB2 para acesso nativo, e foi construído em PHP 5.1. Para mais informações, consulte estes sites:

10. http://pecl.php.net/package/pdo
11. http://pecl.php.net/package/PDO_ODBC

Ligação a bases de dados DB2 não catalogadas

A Listagem 17.1 mostra como se conectar a uma base de dados DB2 usando qualquer das duas opções anteriormente descritas.

```
$host = 'localhost';
$port = 50000;
$DSN = "DRIVER={IBM DB2 ODBC DRIVER}; PORT=$port;
        HOSTNAME=$host; DATABASE=$database; PROTOCOL=TCPIP;
        USER=$user; PWD=$password";

-- Se usar a extensão ibm_db2 --
$uconn = db2_connect($DSN, null, null);

-- Se usar a extensão PDO_ODBC --
try {
  $uconn = new PDO("odbc:$DSN", null, null);
}
catch (PDOException $e) { print $e->errmsg(); }
```

Listagem 17.1 – Ligação a uma base de dados Db2 não catalogada

A Listagem 17.2 exemplifica uma aplicação simples em PHP a usar a extensão ibm_db2.

```
<?php
$sql = "SELECT name, breed FROM ANIMALS WHERE weight < ?";
$conn = db2_connect($database, $user, $password);
$stmt = db2_prepare($conn, $sql);
$res = db2_execute($stmt, array(10));
while ($row = db2_fetch_assoc($stmt)) {
    print "{$row['NAME']} is a {$row['BREED']}.\n";
}
?>
```

Listagem 17.2 – Aplicação simples em PHP a usar a extensão ibm_db2

Configurar PHP para ibm_db2

No Linux ou UNIX pode ser necessário modificar o ficheiro php.ini para o seguinte:

```
extension=ibm_db2.so
ibm_db2.instance_name=<nome da instância>
```

No Windows, modifique o ficheiro php.ini para o seguinte:

```
extension=php_ibm_db2.dll
```

Alternativamente, pode fazer *download* e instalar a aplicação **Zend Core for IBM** que é descrita na secção seguinte, assim não terá que se preocupar com estes problemas de configuração.

17.2.2 Zend Core for IBM

Zend Core é um ambiente PHP de produção e desenvolvimento pronto para uso para aplicações web para soluções críticas. Oferece fiabilidade, produtividade e a flexibilidade necessárias para executar aplicações PHP. Pode fazer o *download* gratuitamente em: http://ibm.com/software/data/info/zendcore

Zend Core for IBM instala os clientes DB2 e IDS, um servidor opcional Apache HTTP Server, PHP 5, e populares extensões PHP incluindo ibm_db2, e PDO_INFORMIX. O Zend Core for IBM pode opcionalmente instalar o servidor DB2 Express-C, IBM Cloudscape ™ server, o manual completo do PHP, e demonstrações de aplicações DB2. Vem com um ambiente PHP fácil de usar e de configurar, como mostra nas figuras 17.5, 17.6 e 17.7.

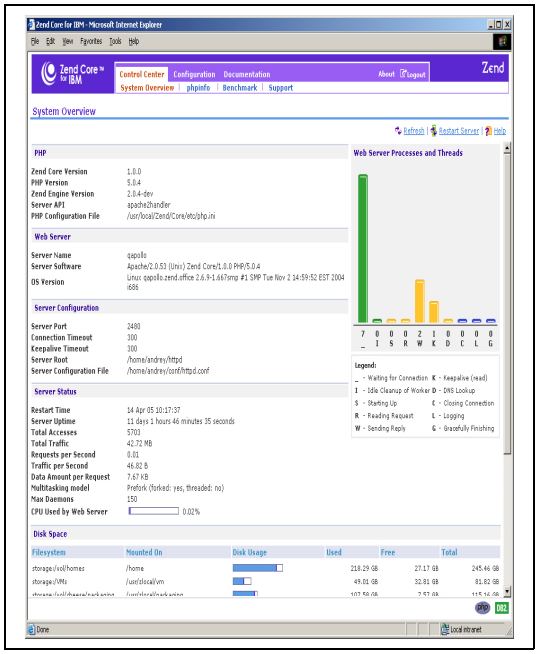


Figura 17.5 - Zend Core interface de gestão e controlo

Figura 17.6 - Zend Core PHP – interface de configuração

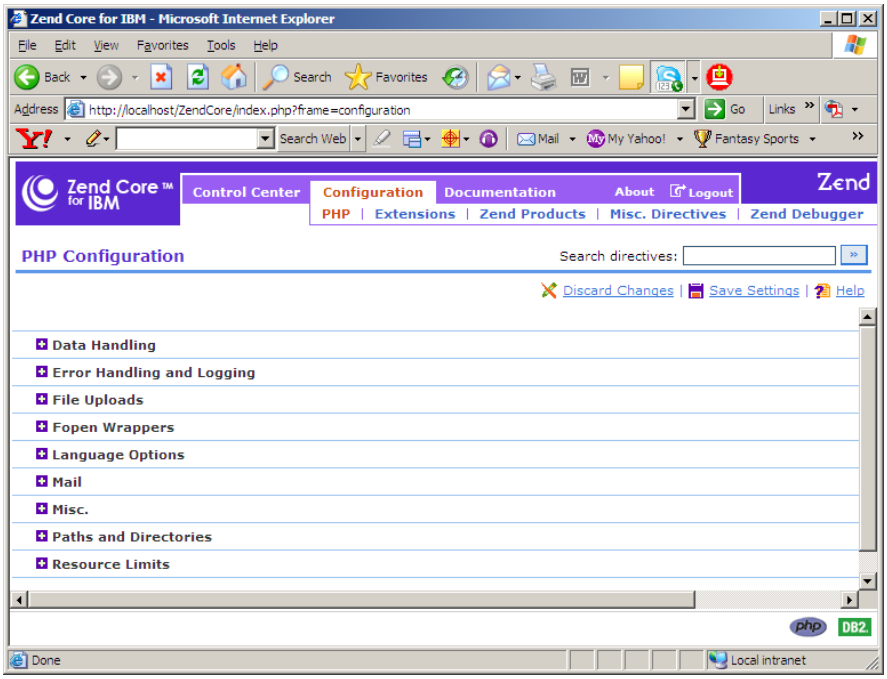


Figura 17.7 - Zend Core PHP – interface de configuração

17.3 Desenvolvimento de aplicações em Ruby on Rails

Ruby é uma linguagem de script multi-plataforma orientada aos objectos e dinâmica. Facilita o desenvolvimento rápido e inclui uma biblioteca poderosa. Ruby é uma linguagem de programação simples e divertida inventada por Yukihiro Matsumoto ("Matz") em 1995.

Rails é uma *framework* de base de dados que apoia as aplicações web escritas em Ruby. Implementa a arquitectura model-view-control (MVC). É extremamente produtiva e fácil de usar. Rails é uma das web *frameworks* de mais rápido desenvolvimento desde 2004 e foi inventada por David Heinemeier Hansson.

17.3.1 *Startup Toolkit* para DB2 on Rails

A IBM reconhece a importância de Ruby on Rails no desenvolvimento da comunidade; por isso, criou-se um pacote chamado **Startup Toolkit para DB2 on Rails**. Este é um instalador que cria um ambiente de desenvolvimento com DB2 Ruby on Rails. Pode fazer o *download* gratuito e utilizar a partir do Web site da IBM alphaWorks: http://www.alphaworks.ibm.com/tech/db2onrails.

O Startup Toolkit para DB2 on Rails:

- Inclui um instalador integrado
  - Ajuda-o a instalar e configurar facilmente Ruby e Rails
- Instala o DB2 Express – C 9 e ferramentas
- Inclui um *driver* desenvolvido pela IBM para DB2 Ruby e um adaptador DB2 Rails
- Inclui várias demonstrações e tutoriais

A

Apêndice A – *Troubleshooting*

Neste apêndice discutimos a forma de encontrar soluções para problemas com que nos podemos deparar quando estamos a trabalhar com DB2. A figura A.1 fornece uma visão geral das acções para resolver um problema.

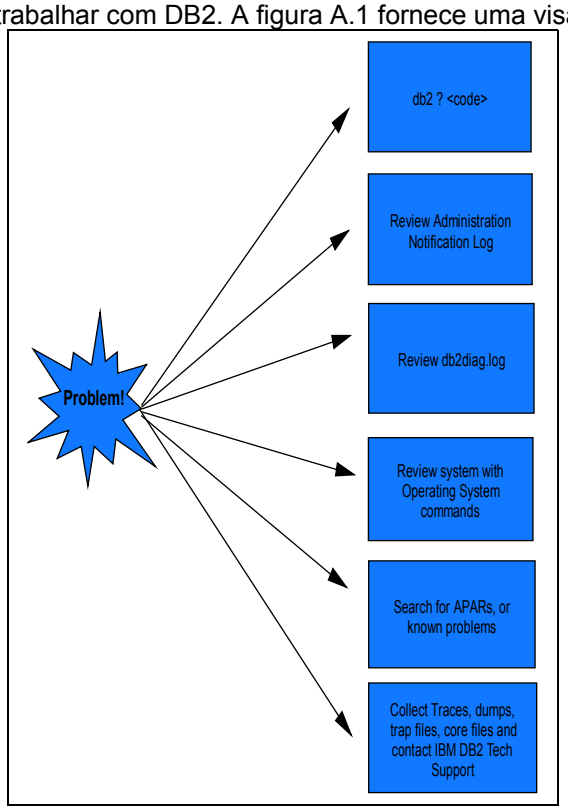


Figura A.1 – Visão global das acções para resolver um problema

> **Nota**:
>
> Para mais informações sobre *troubleshooting*, veja este video:
> http://www.channeldb2.com/video/video/show?id=807741:Video:4462

A.1 Encontrar mais informação acerca dos códigos dos erros

Para obter mais informações sobre o código de um erro, escreva, no *Command Editor,* o código do erro com o prefixo ponto de interrogação (?) e clique no botão Executar *(Execute)* (Figura A.2.).

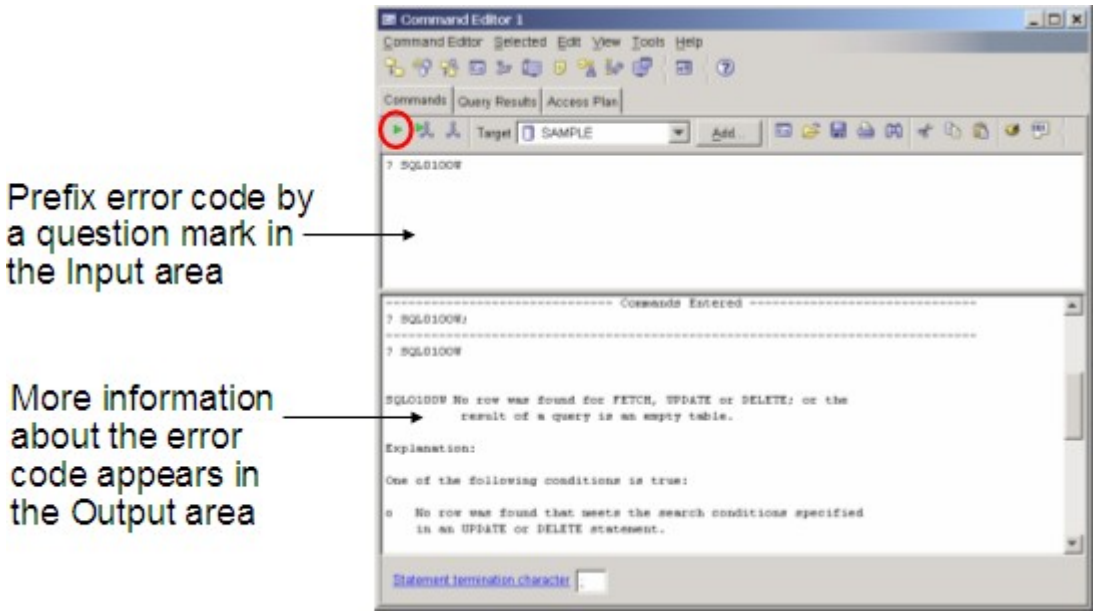


Figura A.2 – Procurar mais informação sobre um código de erro DB2

O ponto de interrogação (?) invoca o comando de ajuda DB2. Abaixo estão alguns exemplos de como invocar a ajuda DB2 se receber, por exemplo, o código de erro SQL "-104". Todos os exemplos a seguir são equivalentes.

```
db2 ? SQL0104N
db2 ? SQL104N
db2 ? SQL-0104
db2 ? SQL-104
db2 ? SQL-104N
```

A.2 SQLCODE e SQLSTATE

Um SQLCODE é um código recebido depois de cada comando SQL ser executado. O significado dos valores estão resumidos a seguir:

SQLCODE = 0; o comando foi bem-sucedido
SQLCODE > 0; o comando foi bem-sucedido, mas devolveu um aviso
SQLCODE < 0; o comando não foi bem-sucedido e devolveu um erro

O SQLSTATE é uma *string* com cinco caracteres que está em conformidade com as normas standard ISO/ANSI SQL92. Os dois primeiros caracteres são conhecidos como o código de classe SQLSTATE:

Uma classe de código 00 significa que o comando foi bem sucedido.
Uma classe de código 01 implica um aviso.
Uma classe de código 02 implica uma *not found condition*.
Todas as outras classes de códigos são consideradas erros.

A.3 DB2 Administration Notification Log

O *DB2 administration notification log* fornece informações de diagnóstico sobre erros no momento da falha. Em plataformas Linux/UNIX, o *administration notification log* é um ficheiro de texto chamado <nome da instância>.nfy (por exemplo, "db2inst1.nfy"). No Windows, todos os *administration notification logs* são escritos para o Windows *Event Log*.

O parâmetro de configuração DBM `notifylevel`, permite aos administradores especificar o nível de informação a ser gravada:

0 -- Nenhuma notificação de mensagem para a administração capturada (não recomendado)
1 -- Erros fatais ou irrecuperáveis
2 -- Acção imediata necessária
3 -- Informação importante, mas não necessária de uma acção imediata (padrão)
4 -- Mensagens informativas

A.4 db2diag.log

O db2diag.log fornece informações mais detalhadas do que o DB2 *administration notification log*. É normalmente apenas utilizada pelo apoio técnico IBM DB2 ou por DBAs experientes. As informações que estão no db2diag.log incluem:

- O código DB2 de localização que relata um erro.
- Identificadores que permitem identificar aplicações registadas no db2diag.log do cliente e servidor.
- Uma mensagem de diagnóstico (começando com "DIA") a explicar a razão do erro.
- Quaisquer dados disponíveis, tais como estruturas de dados SQLCA e apontadores para a localização de outros ficheiros de *dump* ou *trap*.

No windows, o db2diag.log está localizado, por defeito, na directoria:

```
C:\Program Files\IBM\sqllib\<nome da instância>\db2diag.log
```

No Linux/UNIX, o db2diag.log está, por defeito, na directoria:

```
/home/<dono da instância>/sqllib/db2dump/db2diag.log
```

A verbosidade do texto de diagnóstico é determinada pelo parâmetro de configuração dbm cfg DIAGLEVEL. O intervalo é de 0 a 4, onde 0 é a menor verbosidade, e 4 é a maior. O nível padrão é 3.

A.5 CLI *traces*

Para aplicações CLI e Java, pode-se activar a ferramenta de *tracing* da CLI, para ajudar a solucionar problemas com a sua aplicação. Isto pode ser feito alterando o ficheiro db2cli.ini no servidor onde a aplicação está a correr. Abaixo pode ver algumas entradas típicas do ficheiro db2cli.ini:

```
[common]
```

```
trace=0
tracerefreshinterval=300
tracepathname=/path/to/writeable/directory
traceflush=1
```

Um nível mais detalhado de monitorização (db2trc) está também disponível, mas é geralmente apenas aconselhado para o suporte técnico do DB2.

A.6 Defeitos e correções no DB2

Por vezes podemos encontrar um problema que pode ser causado por um defeito no DB2. A IBM disponibiliza *fix packs* regularmente, que contêm o código que corrige os defeitos (APARs). A documentação do *fix pack* contém uma lista das correcções contidas no *fix pack*. Quando se está a desenvolver uma nova aplicação, recomendamos sempre a utilização do ultimo *fix pack* para que o programador possa usufruir das últimas correcções feitas. Para ver a versão actual e o nível do *fix pack*: *Control Center*, seleccione a opção **Sobre** (***About***) a partir do menu **Ajuda** (***Help***); a partir da consola DB2, escreva "db2level". Note que os *fix packs* e o apoio oficial técnico da IBM DB2 são oferecidos no DB2 Express-C apenas se comprar a licença para 12 meses.

Recursos

Web sites:

1. Web site DB2 Express-C:
   www.ibm.com/db2/express
   Use este web site para fazer o download da imagem para obter o servidor DB2 Express-C, clientes DB2, drivers DB2, manuais, acesso ao blog da equipa, registo nas mailing lists, etc

2. Forum DB2 Express: www.ibm.com/developerworks/forums/dw_forum.jsp?forum=805&cat=19
   Use o fórum para colocar questões técnicas, quando não consegue encontrar a resposta nos manuais.

3. Centro de informações DB2
   http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp
   O centro de informações fornece acesso aos manuais online. É a fonte informação mais actualizada.

4. developerWorks
   http://www-128.ibm.com/developerworks/db2
   Este Web site é um excelente recurso para os programadores e DBAs fornecendo acesso aos artigos recentes, tutoriais, etc. gratuitamente.

5. alphaWorks
   http://www.alphaworks.ibm.com/
   Este Web site fornece acesso directo às tecnologias emergentes da IBM. É um lugar onde se pode encontrar as mais recentes tecnologias da *IBM Research*.

6. planetDB2
   www.planetDB2.com
   Este é um agregador de muitos blogs de contribuintes que escrevem sobre DB2.

7. Suporte técnico DB2
   Se comprou licença de 12 meses do DB2 Express-C, pode fazer o download dos *fix packs* a partir deste Web site.
   http://www.ibm.com/software/data/db2/support/db2_9/

8. ChannelDB2
   O ChannelDB2 é uma rede social para a comunidade DB2. Contém vídeos relacionados com DB2, demos, podcasts, blogues, etc. para Linux, UNIX, Windows, z/OS, e i5/OS.
   http://www.channeldb2.com/

Livros

- Free Redbook: DB2 Express-C: The Developer Handbook for XML, PHP, C/C++, Java, and .NET
  Whei-Jen Chen, John Chun, Naomi Ngan, Rakesh Ranjan, Manoj K. Sardana, August 2006 - SG24-7301-00
  http://www.redbooks.ibm.com/abstracts/sg247301.html?Open

- Understanding DB2 – Learning Visually with Examples V9.5
  Raul F. Chong, et all. January 2008
  ISBN-10: 0131580183

- DB2 9: pureXML overview and fast start by Cynthia M. Saracco, Don Chamberlin, Rav Ahuja June 2006 SG24-7298
  http://www.redbooks.ibm.com/abstracts/sg247298.html?Open

- DB2® SQL PL: Essential Guide for DB2® UDB on Linux™, UNIX®, Windows™, i5/OS™, and z/OS®, 2nd Edition
  Zamil Janmohamed, Clara Liu, Drew Bradstock, Raul Chong, Michael Gao, Fraser McArthur, Paul Yip
  ISBN: 0-13-100772-6

- Free Redbook: DB2 pureXML Guide
  Whei-Jen Chen, Art Sammartino, Dobromir Goutev, Felicity Hendricks, Ippei Komi, Ming-Pang Wei, Rav Ahuja, Matthias Nicola. August 2007
  http://www.redbooks.ibm.com/abstracts/sg247315.html?Open

- Information on Demand - Introduction to DB2 9 New Features
  Paul Zikopoulos, George Baklarz, Chris Eaton, Leon Katsnelson
  ISBN-10: 0071487832
  ISBN-13: 978-0071487832

- Redbook: Developing PHP Applications for IBM Data Servers.
  Whei-Jen Chen, Holger Kirstein, Daniel Krook, Kiran H Nair, Piotr Pietrzak
  May 2006 - SG24-7218-00
  http://www.redbooks.ibm.com/abstracts/sg247218.html?Open

Emails de contacto
General DB2 Express-C mailbox: db2x@ca.ibm.com
General DB2 on Campus program mailbox: db2univ@ca.ibm.com