Delphi 7 Studio

Olivier Dahan

Paul Toth

ISBN : 2-212-11143-6

EYROLLES

9

Les différents modes d'accès aux données

Depuis Turbo Pascal et son « Toolbox fichier », l'accès aux données a toujours été intégré dans les environnements Pascal de Borland. Aujourd'hui, Delphi offre de nombreuses techniques fort différentes. Si on ajoute à ces moyens les types Pascal de gestion de fichiers, si on puise dans Windows lui-même ou l'ensemble des composants développés pour Delphi par de nombreuses sociétés dans le monde, les possibilités de choix deviennent pléthoriques.

Procéder à une sélection objective et appropriée pour chaque développement réclame de bien connaître chacune des possibilités offertes par Delphi. Le présent chapitre expose l'ensemble de ces dernières pour que vous puissiez adopter la meilleure solution en toute connaissance de cause. Les chapitres qui suivent détailleront certains modes d'accès en raison de leur introduction récente dans Delphi et de l'intérêt technique qu'ils présentent (comme dbExpress apparu dans Kylix 1 et Delphi 6, expliqué au chapitre suivant).

Les données, matière première de l'application

En dehors de quelques cas extrêmes, les applications travaillent sur des données persistantes, quel que soit le mode de stockage choisi. Avant de se déterminer pour un moyen d'accès aux données, il faut au préalable bien définir les données à traiter.

> Dans cet esprit, le titre d'un célèbre ouvrage de Niklaus Wirth, créateur du Pascal, est fort éclairant : *Algorithmes + structures de données = programmes* (1976). Il pose bien l'essentiel du problème de l'écriture d'une application : trouver l'adéquation entre structures de données (et les moyens d'accès) et les algorithmes qui les traitent. On pourra trouver le titre de cet ouvrage un peu réducteur car il évacue la nécessaire qualité de l'analyse fonctionnelle et la prise en compte des impératifs d'exploitation, mais Wirth traitait de l'aspect technique de l'écriture d'un programme et non de méthodologie.

Une application typique gère habituellement les types de données suivants :

- des données de configuration du logiciel ;
- des données de mémorisation de l'état de l'interface utilisateur ;
- des données temporaires « de travail » ;
- des données techniques, consignation de diagnostics, journaux d'application, messages d'erreurs sauvegardés ;
- des données propres au métier de l'utilisateur final.

Les données de configuration et de mémorisation de l'état de l'interface sont le plus souvent stockées dans des fichiers `INI` ou dans la base de registre de Windows. On peut parfois stocker une partie de ces informations dans une base de données afin d'échapper au partage de la base de registre en réseau et de permettre aux utilisateurs de retrouver leur configuration quel que soit le poste de travail qu'ils utilisent. Reportez-vous au chapitre 8, « Persistance de l'interface et autres paramètres », qui détaille la gestion des fichiers paramètres.

Les données temporaires utilisent tous les formats possibles et sont spécifiques aux traitements qui les impliquent. Il peut s'agir de texte, d'images, de collections d'objets ou de n'importe quoi d'autre. Elles sont stockées soit dans la mémoire, soit sur disque, et leur format dépend du contexte.

Les données techniques sont un peu comme les données de configuration : on peut les stocker tout ou partie en local sur le disque de l'utilisateur ou préférer un stockage en base de données pour les mêmes raisons. La gestion de ces paramètres applicatifs est aussi abordée au chapitre 8.

Enfin, les données propres au métier de l'utilisateur occupent un rôle central et sont le plus souvent stockées dans des bases de données, ne serait-ce que pour en assurer le partage sur un réseau. On rencontre aussi des fichiers dits externes liés à l'application. Il s'agit généralement de documents de toutes sortes : images, traitement de texte, etc.

Delphi propose des solutions pour chaque cas :

- les fichiers texte pur, gérés par le type Pascal `TextFile` ;
- les fichiers d'enregistrements, gérés par le type Pascal `File Of Record` ;
- les fichiers `INI` gérés *via* des `TStrings` ou la classe `TIniFile` ;
- la base de registre de Windows, gérée par la classe `TRegistry` ;
- les fichiers binaires, gérés par le type Pascal `File` ;
- les fichiers binaires, gérés par des flux (classe `TFileStream`) ;

- les fichiers texte enrichi (`RTF`), gérés par la classe `TRichEdit` ;
- les bases de données, gérées selon leur nature.

Il est important de comprendre que tout ne se gère pas avec une base de données et savoir utiliser tous ces moyens est un atout majeur pour une programmation efficace. Nous ne traiterons pas ici de l'ensemble de ces moyens, certains chapitres du présent ouvrage se chargeant d'en présenter les détails les plus utiles.

Nous nous concentrerons en revanche sur les bases de données car, malgré tout, la nécessité de partager des données au travers d'un réseau local ou non, voire d'un intranet ou d'Internet, impose de plus en plus de stocker le maximum d'informations dans de telles bases. Le choix du moyen d'accès devient alors capital pour de nombreuses raisons dont principalement la simplicité de développement, la facilité de déploiement et, bien entendu, la rapidité des échanges avec la base.

À proprement parler, une « base de données » n'est rien d'autre qu'une collection structurée de données. On peut créer une telle base à l'aide de fichiers binaires, ou de fichiers textes pourquoi pas, cela n'enlève rien à la nature même de la base. Toutefois, ce terme est employé couramment pour désigner un type bien particulier de base de données : les bases de données dites relationnelles.

Panorama des stratégies d'accès aux bases de données

Sous l'appellation « base de données relationnelle », on trouve de nombreuses solutions techniquement fort différentes, de l'antédiluvienne dBase et ses dérivés tels que FoxPro ou Nantucket aux bases les plus puissantes comme Oracle, MS SQL Server ou Interbase en passant par des solutions intermédiaires comme PostgreSQL, Paradox ou MS Access. Il ne nous appartient pas ici de faire un classement entre ces produits. Ils existent et il faut pouvoir y accéder.

Nous n'aborderons pas non plus les notions théoriques liées à l'élaboration et l'exploitation des SGBD-R (système de gestion de bases de données relationnelles) pour nous occuper essentiellement des choix que Delphi propose pour accéder à ces systèmes, à savoir :

- le Borland Database Engine (BDE)
- ODBC (*via* le BDE ou dbExpress)
- ADO
- dbExpress/DataSnap Direct
- Interbase Express
- DataSnap
- MyBase

Le BDE face aux autres solutions, épilogue d'une fin annoncée

Le Borland Database Engine (BDE) est hérité d'un lointain passé. S'il est aujourd'hui *deprecated* pour sa partie SQL Links, il reste très utilisé, et nombreux sont les développeurs Delphi qui ne connaissent que lui. Nous l'aborderons en premier car sa popularité est telle que les nouveaux moyens d'accès aux données ont calqué leur fonctionnement apparent sur la logique offerte par le BDE. Comprendre les grandes lignes du BDE, si vous ne le connaissez pas, vous permettra de mieux aborder les autres moyens mis à disposition par Delphi.

Historique du BDE

L'histoire du BDE commence après le rachat par Borland en 1987 de Paradox, créé par ANSA Software en 1985. À partir de ce moment, les évolutions de ce produit furent nombreuses. Module « résidant » en mémoire sous DOS, le moteur de Paradox est devenu le BDE sous Windows au moment du rachat de dBase à ASTON TATE par Borland en juillet 1991. À la différence de Microsoft qui a préféré gérer ses différents produits de façon totalement séparée (FoxPro et MS Access), Borland a alors choisi de fusionner les deux moteurs en sa possession, le moteur de Paradox (ODAPI) et dBase, ce qui a donné le premier embryon du BDE (IDAPI).

Pour répondre aux besoins d'accès aux bases de données externes, tout en profitant de l'interface du logiciel Paradox, Borland créa deux ouvertures au moteur : l'accès natif à certains SGBD-R par le biais des « SQL Links » et l'accès à toutes les bases déjà utilisables par le système ODBC. Il fut d'ailleurs déjà question à cette époque de concevoir un système séparé du BDE pour gérer les bases SQL. Ce projet non divulgué au grand public et qui était connu sous divers noms dont « SQL Express » fut abandonné au profit des SQL Links lors de l'intégration du BDE dans Delphi 1. On voit aujourd'hui, avec l'avènement de dbExpress, que cette idée a continué son chemin chez Borland pour devenir la nouvelle stratégie dans Delphi 6 et les versions suivantes.

À ce stade, le BDE était devenu ce qu'il est encore aujourd'hui : une interface standardisée de programmation ouvrant la voie vers la quasi-totalité des SGBD du marché. Et c'est tout naturellement sous cette forme que le BDE fut intégré à la première version de Delphi qui fonctionnait sous Windows 3.x en 16 bits et qu'on a ensuite retrouvé en 32 bits dans les versions suivantes de ce produit et de C++ Builder.

Souvent critiqué notamment pour son poids (gênant pour un déploiement de logiciel shareware *via* Internet principalement, et en entreprise sur un réseau d'une centaine de machines), le BDE est un merveilleux outil ayant rendu des services inestimables durant des années. La partie native étant libre de diffusion (formats Paradox et dBase), le BDE est encore utilisé par de nombreux logiciels sur le marché.

Mais tout a une fin… Avec la sortie de Kylix, Borland n'avait pas d'autres choix que de réécrire le BDE pour le porter sous Linux ou d'inventer un nouveau système d'accès aux données utilisable sous Linux et Windows. C'est cette dernière solution qui a été préférée et qui a donné naissance à dbExpress que nous verrons dans la suite de ce chapitre.

Doit-on encore choisir le BDE aujourd'hui ?

La réponse n'est pas si évidente… En effet, Borland ayant annoncé mi-2002 la fin du BDE, il serait logique de dire que ce produit ne doit plus être utilisé aujourd'hui. D'un autre côté, cette fin est toute relative et la réponse peut être mitigée.

Comme nous le disions plus haut, pour Delphi 6 et Kylix 1 Borland a créé un nouveau système d'accès aux données, dbExpress. Mais cet outil est exclusivement destiné à l'accès aux serveurs SQL, même si on commence à trouver des drivers pour des bases de technologie un peu différente. Ce n'est ainsi que la partie SQL Links du BDE qui disparaîtra à partir de 2003. Dans Delphi 7 Studio, les SQL Links sont toujours livrés mais sous le mode *deprecated* (qui ne doit plus être utilisé) afin de laisser le temps nécessaire à la migration des applications vers dbExpress. À partir de 2003, le BDE reviendra à ses origines, c'est-à-dire un moteur pour les formats Paradox et dBase. S'il ne connaîtra plus aucune évolution, il pourra toujours être utilisé pour gérer ces formats. La question se pose donc en réalité différemment, à savoir « quelle base de données utiliser pour un développement donné ».

De deux choses l'une : soit vous avez le choix d'imposer un SGBD spécifique, soit vous devez vous adapter à celui que le client a déjà sélectionné. Dans le premier cas, vous opterez certainement pour une base solide, donc Interbase, Oracle, MS SQL Server ou tout autre SGBD que vous maîtrisez. Le BDE n'est donc ici plus nécessaire puisque ces bases peuvent être gérées au travers de composants d'accès direct ou par dbExpress.

Dans le second cas, vous devrez vous accommoder et surtout vous plier aux exigences techniques du client. Si ce dernier impose Paradox, dBase ou FoxPro, le BDE est l'outil *ad hoc*. En dehors de ce créneau bien particulier et de moins en moins fréquent, vous serez plutôt confrontés à de vraies bases SQL.

Plusieurs moyens d'accès sont ici possibles, à savoir :

- le BDE avec le SQL Link correspondant au SGBD cible ;
- des composants d'accès direct n'utilisant pas le BDE ;
- le nouveau système dbExpress.

La fin du support des SQL Links fait que cette solution n'est plus raisonnable pour de nouveaux développements même s'ils restent utilisables encore quelque temps. Nous ne pouvons ainsi que vous déconseiller cette approche.

Des composants d'accès direct existent pour les principales bases. Delphi intègre les IB Express pour Interbase (ou FireBird), les autres peuvent être acquis auprès de fournisseurs tiers. Il existe même des librairies gratuites de ce type (IB Express est lui-même issu d'une telle librairie). Simplifiant le déploiement, ce mode d'accès aux SGBD est plus performant que le BDE qui souffre de son côté généraliste, imposant un circuit complexe à l'information au travers des différentes couches logicielles (DLL cliente, SQL Links, BDE).

dbExpress reprend l'esprit du BDE en offrant une interface centralisée et la possibilité d'utiliser des drivers spécifiques en alliant des performances proches ou identiques aux

composants d'accès direct. Cela est dû en grande partie à l'extrême simplicité de ce nouveau système réduisant le nombre de couches à traverser par l'information. Si vous visez en plus la portabilité sous Linux, dbExpress sera un excellent choix. À la lecture de ce qui précède, on s'aperçoit que le BDE est un choix qui ne se justifie plus que pour les formats Paradox et dBase, formats de moins en moins utilisés en environnement professionnel.

Quand utiliser le BDE ?

Arrivé en fin de vie, le BDE n'est plus une solution d'avenir. Toutefois, enrichi par ses années d'évolution, ce produit sait rendre des services qui ne peuvent être satisfaits par les autres solutions disponibles. Connaître ces avantages particuliers peut vous sauver la mise dans certaines situations.

Les requêtes hétérogènes

Les requêtes hétérogènes dont on parle assez peu sont souvent d'un très grand secours. Ignorer leur existence conduit à se priver d'une solution simple pour des problèmes qui, parfois, se révèlent complexes à résoudre autrement. Sur ce point, le BDE est la seule solution, parmi toutes celles évoquées jusqu'à maintenant, qui offre un support pour ces requêtes d'un type spécial.

Qu'est-ce qu'une requête hétérogène ?

On serait tenter de dire qu'elle n'est pas homogène justement… mais ce serait un peu court ! Il s'agit en fait d'une requête SQL mettant en relation des tables qui se trouvent dans des bases de données différentes. Il peut s'agir par exemple de croiser des informations entre une base Interbase, une base Oracle et une base Paradox dans la même requête, ou bien, et cela est encore moins bien compris généralement, plusieurs bases de même nature simultanément (plusieurs bases Interbase dans la même requête par exemple).

Aussi étonnant que cela puisse paraître, bon nombre de serveurs SQL ne permettent pas à une requête de mixer des tables provenant de bases différentes mais pourtant du même type. Ainsi, il ne sera pas possible de faire un rapprochement entre des données de deux bases Interbase (deux fichiers GDB différents) en passant directement par le moteur Interbase. Seul une requête hétérogène gérée par le BDE le permettra. Il en est de même avec les autres serveurs SQL en général.

Si le BDE peut se permettre ce tour de passe-passe, c'est qu'il est avant tout lui-même un moteur SQL, notamment pour Paradox. Une requête hétérogène est gérée selon la syntaxe du SQL dite *Local SQL*, et c'est la machine cliente qui traite l'ensemble de la requête. Cela implique que toutes les tables utilisées dans la requête sont rapatriées par le BDE pour être traitées. Les performances ne sont jamais excellentes et l'encombrement du réseau est maximal, mais lorsqu'il n'existe qu'un seul pont pour traverser un fleuve, peut-on se permettre de tergiverser sur la beauté de son infrastructure ou se refuser à l'emprunter car sa couleur n'est pas de notre goût…

Seul le BDE vous permettra de faire ce grand écart multibase. Dans ce cas particulier, son utilisation se justifie pleinement mais plutôt au sein d'un utilitaire précis, découplé des

applications. Les SQL Links étant encore livrés avec Delphi 6 et 7, cela laisse malgré tout quelques années pour trouver le moyen d'éviter les requêtes hétérogènes (en fusionnant les bases, en utilisant des systèmes de réplications…).

L'importation et l'exportation de données

Voici un joli piège… L'auteur a vu des développeurs déployer des efforts conséquents pour se passer du BDE, tout cela pour lire dans les dernières pages du cahier des charges (celles auxquelles personne ne prête attention !) que l'utilisateur pourra exporter (ou importer) des données au format Paradox ou dBase !

Bien que ces formats ne soient plus d'actualité pour des développements professionnels, ils restent des standards très utilisés pour l'échange de données entre logiciels. XML se veut un nouveau format universel mais il reste verbeux et lent à traiter car nécessairement « parsé[1] » là où les formats de fichier plat, tels que ceux évoqués, sont plus compacts et autorisent des accès directs quasi instantanés à quelque enregistrement que ce soit. De plus, il faut se méfier des effets de mode… Aujourd'hui encore, de nombreuses applications utilisent les formats dBase et Paradox et il ne suffit pas de décréter que ces derniers sont *has-been* pour les voir disparaître en un trimestre. On voit encore aujourd'hui des PC sous Windows 3.x dans les entreprises alors que depuis plus de sept ans ce système n'est plus vendu ni maintenu…

Dès lors, vous pouvez être confrontés au besoin d'importer ou d'exporter des données dans ces formats. Si on peut trouver des librairies gérant les anciennes normes dBase (notamment dBase III et IV), aucune ne gère totalement les dernières versions du format et aucune ne sait gérer les formats V et VII de Paradox.

Si votre application doit se plier à une exigence de cet ordre, inutile de vous poser trop de questions… Sauf à convaincre le client de faire un autre choix, le BDE sera votre seul moyen de lire et d'écrire des fichiers Paradox et dBase. Le BDE en tant que moteur Paradox et dBase étant maintenu dans les prochaines versions de Delphi, vous pourrez vous en servir dans ce cadre bien délimité. Nous vous conseillons toutefois assez clairement de dissocier le code utilisant le BDE de vos applications en fournissant les fonctions d'importation et d'exportation sous la forme de modules autonomes ne rendant pas la fourniture du BDE indispensable.

Alors, BDE ou pas ?

Comme nous venons de le voir, si le BDE en tant que grand fédérateur de bases de données cède la place à dbExpress, on ne peut pas, non plus, l'enterrer aussi facilement. Le nombre des services uniques qu'il rend peut en faire un allié indispensable dans certains cas que nous limiterons aujourd'hui à des utilitaires découplés des applications utilisant dbExpress.

1. La technique du parsing, du verbe *to parse* qui, en anglais, signifie « faire une analyse grammaticale de », consiste à décomposer un flux, généralement textuel, en série de tokens (éléments syntaxiques de base), typés selon leur position dans le flux, cela en vu d'interpréter ce même flux.

Les composants propres au BDE

Delphi propose une palette de composants liée au BDE, que voici (figure 9-1) :

Figure 9-1
Les composants du BDE

De gauche à droite :

- `TTable`, accès en lecture et écriture à toute table d'une base.
- `TQuery` permet d'envoyer des ordres SQL à la base et d'obtenir le résultat (pour les requêtes de type `SELECT`). La base n'offre qu'un accès en lecture seule mais peut accepter un `TUpdateSQL` ou fonctionner en mode requête vivante sur des requêtes monotables.
- `TStoredProc` donne accès aux procédures stockées.
- `TDatabase` ouvre un canal de communication avec une base de données.
- `TSession` regroupe les communications de tous les `TDatabase` utilisés dans l'application.
- `TBatchmove` assure les opérations de copies et de mises à jour par lot.
- `TUpdateSQL` stocke des requêtes de mise à jour. Associés à un `Tquery`, il permet de rendre ce dernier modifiable.
- `TnestedTable` autorise la gestion de tables imbriquées dans le champ d'une autre table.

Logique de fonctionnement

La logique d'accès aux données *via* le BDE est donnée par la figure 9-2. Nous la présentons en détail car, bien que le BDE ne soit plus un choix d'avenir, cette logique est commune à tous les autres moyens d'accès offerts par Delphi qui ont plus ou moins calqués leur fonctionnement sur celui du BDE (en tout cas dans la logique des composants d'accès).

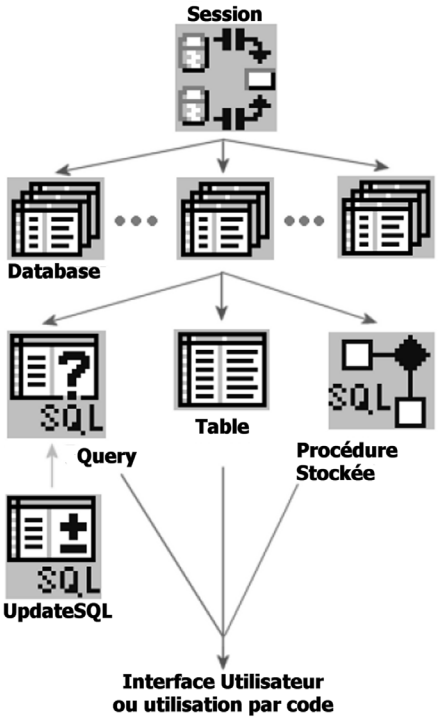
Une application possède en général un objet `Session` auquel sont connectés les objets `Database`. À ces derniers sont connectés soit des objets requêtes (avec leurs éventuels objets de mise à jour), soit des objets tables, soit encore des objets procédures stockées.

Chacun de ces objets est ensuite connecté (ou non) à l'interface utilisateur *via* des composants source (`TDatasource` non représenté ici car non spécifique BDE), eux-mêmes servant de concentrateurs aux objets d'interface tels que les champs d'édition.

Il s'agit d'un schéma simple et compréhensible rendant le développement assez intuitif. C'est d'ailleurs la raison pour laquelle cette organisation proposée dès Delphi 1 et modifiée dans Delphi 3 (séparation du `TDataSet` pour le rendre indépendant du BDE) tend à

être reprise par toutes les autres librairies d'accès aux données (malgré quelques différences imposées par la logique même des bases sous-jacentes).

Figure 9-2
Logique d'accès du BDE



ODBC : un standard d'accès aux bases de données

L'*Open DataBase Connectivity*, ODBC, est un standard pour l'accès unifié aux bases de données. Il a été conçu pour fonctionner sur plusieurs plates-formes et a été porté sous Win32, Unix, Macintosh, OS/2 et quelques autres. Décrié pour sa lenteur et sa stabilité toute relative à ses débuts, il a connu un tel succès que des éditeurs comme IBM, Informix ou Watcom, l'ont utilisé pour fonder leurs API. Souvent associé à tort à Microsoft, ODBC a en réalité été conçu par un consortium d'éditeurs et de groupes de normalisation dont Microsoft n'est qu'un acteur aux côtés de X/Open, du SQL Access Group, de l'ANSI, l'ISO, Digital, Sybase, Novell et de nombreux autres.

Il utilise une notion proche de l'alias du BDE : le DSN (Data Source Name). Des composants tiers gratuits ou payants permettent d'accéder directement à ODBC depuis une application Delphi. Toutefois, le BDE offre aussi une passerelle vers ses drivers, ce qui permet de bénéficier de tout le support standard des bases de données intégré dans Delphi. Il faut noter qu'il existe aussi des drivers dbExpress pour ODBC. Lorsqu'une connexion ODBC est exploitée *via* le BDE ou dbExpress, tous les composants de ces derniers sont disponibles.

ADO – l'accès base de données par objets

ADO est un sigle signifiant « ActiveX Data Objects ». C'est un système d'accès aux données annoncé comme stratégique par Microsoft même s'il reste difficile de bien comprendre les différences entre tous les systèmes de ce type proposés par cet éditeur. ADO, OLE DB, OLAP et ODBC, qui subsiste encore, forment un ensemble peu lisible.

Selon Microsoft, OLE DB est un moyen d'accès bas niveau aux bases de données utilisant le modèle COM et ses interfaces. OLAP est intégré à OLE DB et signifie Online Analytical Processing, autant dire que même éclaté le sigle reste assez nébuleux !

Quant à ADO, Microsoft le positionne comme un système d'accès aux données de haut niveau.

Tout cela est bien confus et nous laissons aux aficionados de la firme de Redmond le soin de démêler l'écheveau. Côté Delphi, les choses sont plus claires[1], bien heureusement, et l'accès aux bases de données par ADO s'effectue par un ensemble de composants assez semblable, dans son principe, à celui du BDE.

La palette de composants

Figure 9-3
Les composants ADO

De gauche à droite, nous trouvons les composants suivants :

- `TADOConnection` assure la connexion à une base de données et offre le support des transactions. On peut l'assimiler à un ensemble `TSession` + `TDatabase` du BDE.
- `TADOCommand` permet d'envoyer des commandes SQL à une base sans retour d'information.
- `TADODataset`, outil générique permettant d'accéder aux tables tout autant qu'envoyer des requêtes SQL retournant des données.
- `TADOTable` spécialise le précédent en simulant le comportant d'un `TTable` du BDE.
- `TADOQuery`, selon le même principe, se présente sous la forme équivalente à celle du `TQuery` du BDE.
- `TADOStoredProc` autorise le dialogue avec les procédures stockées comme le `TStoredProc` du BDE.
- `TRDSConnection` permet le transfert des données en assurant le marshaling des données d'une machine à l'autre. S'utilise principalement dans les applications multiniveaux.

1. Effet probable de la profusion de noms à la façon MS, il faut noter que les composants ADO de Delphi apparaissent sous plusieurs noms, notamment ADO Express ou dbGO qui, selon Borland, se réfèrent au même produit.

La logique de fonctionnement étant très proche de celle des composants BDE présentés plus haut, vous ne rencontrerez pas de difficultés majeures lors de la mise en œuvre de ADO.

dbExpress

Nouveau standard proposé par Borland depuis la sortie de Kylix 1 et Delphi 6, dbExpress se veut le successeur du BDE en termes d'API unifiée d'accès aux données et plus spécialement aux serveurs SQL. dbExpress se matérialise par une suite de composants qui, une fois compilés dans vos applications, donnent un accès direct à des drivers (DLL Windows ou librairies Linux) en charge de la liaison avec la base de données cible choisie.

Outre la portabilité Linux/Windows de ce nouveau standard, Borland a eu la bonne idée de fournir dès le départ tout ce qu'il faut pour que chacun puisse écrire s'il le désire de nouveaux drivers. On trouve ainsi de nombreux projets en cours de développement ou achevés offrant des drivers dbExpress pour des bases non supportées à l'origine, et ce, sans être tributaires de Borland.

Le fait que dbExpress s'affiche comme remplaçant du BDE en termes d'API unifiée, doublé par sa présence simultanée sous Linux et Windows et son ouverture au développement de nouveaux drivers, place ce nouveau système en tête de nos préférences pour tout projet ayant à s'inscrire dans le temps et dans le cadre d'un éventuel portage sous Linux.

La technique mise en œuvre dans dbExpress est d'une grande simplicité, ce qui lui garantit des temps de réponse excellents. Là où le BDE perd son temps entre ses diverses couches, dbExpress trace son chemin rapidement.

Les éléments qui plaident pour dbExpress contre les SQL Links et qui justifient la disparition de ces derniers se résument ainsi :

- disponible sous Delphi, Kylix, C++Builder ;
- cross-platform Linux/Windows ;
- traite les requêtes et les procédures stockées comme le BDE ;
- plus léger que le BDE, plus simple à déployer ;
- plus rapide que le BDE ;
- mieux taillé pour DataSnap (Midas).

La légèreté et la rapidité ont un prix, celui de la simplicité… Ainsi, de base, dbExpress ne gère que des ensembles de données unidirectionnels. Pas question de connecter une `TDBGrid` ou un `TDBNavigator` qui imposent de pouvoir monter et descendre dans les données. De même que dbExpress ne fait aucun *caching* des données ni des métadonnées.

Delphi propose toutefois un composant utilisant le concept du `TclientDataset` pour offrir sous dbExpress des ensembles de données navigables. Mais cela se paie par le déploiement de la DLL DataSnap (certes peu encombrante) et par certaines difficultés dont la création de relations maître-détail déconseillée avec les `TClientDataset`.

Développer avec dbExpress offre de multiples avantages mais impose donc d'adopter un style différent et une présentation des données au niveau de l'interface utilisateur en accord avec les possibilités du système.

dbExpress est en quelque sorte construit autour d'un concept « anti-gadget ». Cela garantit son efficacité mais supprime les petits avantages qu'on reconnaissait au BDE et qui en faisaient l'intérêt. Ainsi, sous dbExpress vous ne trouverez pas :

- la notion de Live Query ;
- le mode Cached Update ;
- de support de création de schémas ;
- de support pour les requêtes hétérogènes ;
- des services tels que le Batch Move ;
- les copies, tris et autres API du BDE.

Avec dbExpress tout se passe par requête SQL, qu'il s'agisse de créer une table, de modifier un trigger ou de présenter des données triées selon un certain ordre. Mais comme ces contraintes sont celles d'un bon développement soucieux d'être en harmonie avec la technologie des bases de données SQL, dbExpress ne fera que vous obliger à respecter cet esprit, ce qui ne peut que vous aider à concevoir des applications efficaces.

Les composants dbExpress

Figure 9-4
Les composants dbExpress

De gauche à droite :

- `TSQLConnection` encapsule la connexion au serveur de données, équivalent au couple `Tsession` + `TDatabase` du BDE.
- `TSQLDataset`, outil générique de communication avec les bases de données (SQL, tables, etc).
- `TSQLQuery` permet d'envoyer des ordres SQL à la base et de récupérer l'éventuel résultat. Est équivalent au `TQuery` BDE, en dehors de spécificités propres à dbExpress (pas de requête « live », résultat unidirectionnel, etc.).

- `TSQLStoredProd` donne accès aux procédures stockées.
- `TSQLTable` fonctionne un peu comme un `TTable` du BDE. Tout comme le `TSQLQuery`, sa présence se justifie pour simplifier le portage d'applications BDE vers dbExpress.
- `TSQLMonitor` intercepte et stocke les ordres SQL qui transitent vers la base de données afin d'autoriser une analyse de l'activité d'une application.
- `TSQLClientDataset` est l'équivalent du `TSQLDataset` mais il encapsule, en plus, un `TClientDataset` MIDAS (DataSnap) pour offrir à dbExpress à la fois un ensemble de données exploitant un cache local et surtout le navigationnel absent de dbExpress. Ce composant est aujourd'hui *deprecated* (on ne doit plus l'utiliser), et il est remplacé par :
- `TSimpleDataset` remplace le `TSQLClientDataset` pour les utilisations 2-tiers. Il intègre à la fois le dataset et la connexion à la base.

Du point vue pratique, dbExpress se révèle simple à mettre en œuvre et offre un excellent niveau de performance, supérieur à celui du BDE. L'utilisation de dbExpress est présentée plus en détail au chapitre 10.

DataSnap

DataSnap est un nouveau concept introduit dans Delphi 6 qui semble de prime abord plus commercial que technique puisque sous ce vocable se cache Midas, système de transport de données pour la création d'architectures dites 3-tiers ou n-tiers.

En fait, DataSnap a ajouté à Midas une ouverture sur le Web des plus intéressantes. Son positionnement est d'offrir un support d'architecture distribuée dont l'un des aspects marquants est de pouvoir s'adapter à la montée en charge de l'utilisation d'une base de données en termes de transactions et d'utilisateurs connectés. La communication s'effectue sur la base des standards industriels de l'instant à savoir SOAP/XML, COM et TCP/IP (Corba reste utilisable mais le composant de connexion présent dans Delphi 6 n'est plus maintenu dans Delphi 7 Studio).

Résolument positionné pour le Web, il ne faut pas oublier que DataSnap permet aussi de faire des applications dites à « client léger » hors Web et que ces performances sont excellentes, expérience faite sur de gros réseaux (plusieurs centaines d'utilisateurs en ligne). De telles applications sont découpées en une partie cliente qui utilise des `TClientDataset` connectés à des fournisseurs de données `TDatasetProvider` eux-mêmes situés dans l'application serveur, et qui communiquent *via* une couche réseau. La base de données pouvant être placée sur une autre machine et ce, même après-coup, on comprend que la gestion de la montée en charge s'en trouve grandement facilitée (en dehors des capacités d'équilibrage de charge – *load balancing* – propres à DataSnap).

DataSnap est un excellent choix dès lors qu'une application doit pouvoir supporter de nombreuses connexions et que la montée en charge doit être maîtrisée dans le temps. De la même façon, ce système simplifie la création d'applications exploitant le Web pour communiquer.

L'utilisation de DataSnap en simple remplacement du modèle classique en 2-tiers est aussi à conseiller, ne serait-ce qu'en raison de la légèreté de la partie cliente. En effet, dans une application client-serveur classique, le poste client doit être équipé, outre de l'application elle-même, du système d'accès client propre à la base de données choisie. Si cela est supportable sur 10 postes, mettre à jour la partie cliente d'Oracle ou Interbase pour ne citer qu'eux sur plus de 200 postes devient vite une charge énorme. Si l'application est conçue en mode 3-tiers grâce à DataSnap, le poste client ne doit être équipé que de l'application cliente (et de la DLL DataSnap). On comprend que la maintenance du parc s'en trouve allégée et simplifiée. La suppression des coûts d'intervention pour installer, mettre à jour ou paramétrer les parties clientes de la base de données sur un grand parc de micro-ordinateurs remboursait dès l'origine très largement le faible coût de la licence DataSnap, aujourd'hui gratuite, et le surcoût lors de la production de l'application en 3-tiers.

Le développement est en outre assez simple grâce à la bonne intégration dans l'IDE de Delphi des outils nécessaires au codage et à la mise au point de telles applications. Jusqu'à Delphi 6, il fallait acquérir une licence pour déployer une application DataSnap. Borland venant d'annoncer la gratuité totale du déploiement de DataSnap, ce petit frein n'existe même plus (la licence était vendue moins de 400 €). Comme l'annonce de cette gratuité est récente et qu'elle peut dépendre de la version de Delphi que vous avez achetée, reportez-vous à la documentation de déploiement fournie avec Delphi pour vous assurer des conditions de distribution de DataSnap.

Les composants DataSnap

Figure 9-5
Les composants DataSnap

De gauche à droite :

- `TDCOMConnection` sert à établir une communication DCOM avec un serveur distant.
- `TSocketConnection` sert à établir une communication TCP/IP avec un serveur distant.
- `TSimpleObjectBroker`, courtier de connexion à utiliser pour automatiser la connexion de l'application sur plusieurs serveurs (choix d'un serveur disponible).
- `TWebConnection` sert à établir une communication http avec un serveur distant.
- `TConnectionBroker` permet de centraliser toutes les connexions de l'application et d'éviter la réécriture de celle-ci en cas de changement de technique de connexion (TCP/IP, http, etc.).
- `TSharedConnection` permet à l'application de se connecter à plusieurs modules de données distants.

- `TLocalConnection` simule la connexion distante en local afin de préparer l'application à une évolution multi-tiers distant sans reprogrammation.

DataSnap sans Corba

Dans Delphi 7 Studio, l'objet `TCorbaConnection` n'est plus intégré dans la version finale du produit. En effet, cet objet est basé sur l'ORB de Visibroker 3.3 alors que Delphi 7 sera livré avec Visibroker 4.5 avec lequel il n'est pas compatible. Il semble donc qu'une suppression pure et simple soit au final décidée.

On notera que, pour créer une application utilisant SOAP comme transport, il faut créer le serveur comme un service Web, ce qui s'effectue en utilisant l'expert Delphi fourni à cet effet.

Il ne faut pas oublier non plus d'ajouter aux composants orientés connectivité présentés plus haut tous ceux servant à l'accès proprement dit aux données et se trouvant dans la palette « Accès BD » comme le `TClientDataSet` qui représente une table ou requête obtenue par DataSnap, ou comme le `TDatasetProvider` qui est son pendant côté serveur et dont la tâche est d'encoder les données pour les envoyer au client. L'utilisation de DataSnap est présentée plus en détail au chapitre 12.

Interbase Express

Déjà présent dans Delphi 5, Interbase Express a marqué l'officialisation de la fin du « tout BDE » dans Delphi. Issu d'une librairie gratuite, IBX est un ensemble de composants permettant de se connecter à Interbase et FireBird directement, sans passer donc par le BDE ni aucune autre couche (en dehors de la DLL cliente d'Interbase).

IBX est à préconiser chaque fois qu'on doit développer une application client-serveur utilisant Interbase ou FireBird. Ces derniers sont eux-mêmes à conseiller chaudement à chaque fois qu'on a le pouvoir d'influencer le choix de la base de données chez le client. IBX sera aussi parfaitement à sa place dans une application en mode n-tiers, notamment parce que ces ensembles de données unidirectionnels fonctionnent en totale adéquation avec la logique DataSnap (le navigationnel est réalisé par le `TClientDataset` se trouvant côté client).

Interbase, gratuit ou payant ?

Il semble que beaucoup de gens se posent encore la question de savoir si Interbase est réellement gratuit ou non. La réponse est fort simple : Interbase 6.0 a été publié en Open Source par Borland et cette version particulière est gratuite. Borland continue de faire évoluer le produit et propose ainsi toujours des licences payantes intégrant de nombreux outils non disponibles gratuitement comme le moteur de réplication. À partir de la version Open Source, un groupe de travail composé de bénévoles, dont d'anciens développeurs ayant travaillé sur Interbase, a conçu un projet parallèle connu sous le nom de FireBird qui est un Interbase 6 évoluant dans sa direction propre. Ce produit est Open Source et gratuit.

Les composants IB Express

Figure 9-6
Les composants Interbase

La palette de composants Interbase est particulièrement riche d'autant qu'elle est complétée par une seconde palette dédiée à l'administration que nous verrons plus bas.

De gauche à droite :

- `TIBTable` correspond au `TTable` du BDE et n'est présent que pour simplifier le portage d'applications BDE.
- `TIBQuery` correspond au `TQuery` du BDE et sa présence se justifie pour la même raison que le `TIBTable`.
- `TIBStoredProc` correspond au `TStoredProc` du BDE pour l'accès aux procédures stockées sur le serveur.
- `TIBDatabase` correspond au couple `Tdatabase` + `TSession` du BDE. Il assure la centralisation des communications vers la base de données.
- `TIBTransaction`, objet spécifique Interbase permettant de contrôler finement les transactions.
- `TIBUpdateSQL` correspond au `TUpdateSQL` du BDE et offre les mêmes services (à savoir fournir les requêtes de mises à jour à un objet `Requête` pour le rendre actif en écriture).
- `TIBDataset`, composant générique pour accéder aux tables, vues et autres objets retournant des données.
- `TIBSQL`, version spécifique du `TIBQuery` plus proche du moteur, plus rapide mais non orienté interface et offrant un accès unidirectionnel aux données.
- `TIBDatabaseInfo` retourne de nombreuses informations sur une base de données (utilisateurs, mémoire utilisée, etc).
- `TIBSQLMonitor`, moniteur SQL permettant d'analyser les instructions échangées avec le serveur.
- `TIBEvents`, spécifique à Interbase et sa gestion des événements. Permet d'être averti lorsqu'un événement précis est déclenché par la base.
- `TIBExtract` permet d'extraire les métadonnées d'une base.
- `TIBClientDataset` propose un cache local pour les mises à jour vers la base de données (utilise un `TClientDataset` DataSnap de façon interne).

- En fin de palette, les autres composants sont ceux des IBX Extras téléchargeables sur *Code Central*. Produits par Jeff Overcash, créateur des IBX, ils ne sont pas fournis en tant qu'éléments standards avec Delphi.

Les composants d'administration

Figure 9-7
Les composants d'administration Interbase

De gauche à droite :

- `TIBConfigService` permet de configurer les paramètres de la base de données (intervalle de balayage, mode asynchrone, etc).
- `TIBBackupService` permet d'exécuter depuis un poste client une sauvegarde sur le serveur.
- `TIBRestoreService` permet d'exécuter depuis un poste client une restauration de base de données sur le serveur.
- `TIBValidationService` permet d'exécuter une validation des données de la base.
- `TIBStatisticalService` retourne l'ensemble des statistiques serveur d'une base.
- `TIBLogService` assure le contrôle et l'accès au fichier journal tenu par le serveur.
- `TIBSecurityService` donne le contrôle sur la gestion des utilisateurs de la base.
- `TIBLicensingService`, inutile avec FireBird qui est gratuit, permet de gérer les licences Interbase actives.
- `TIBServerProperties` retourne les informations de configuration du serveur Interbase.
- `TIBInstall` automatise l'installation d'Interbase lui-même depuis une application.
- `TIBUninstall` automatise la désinstallation d'Interbase depuis une application.

Comme vous pouvez le constater, le support d'Interbase est particulièrement bien soigné dans Delphi. De l'installation du serveur lui-même au contrôle des transactions, tout est là pour vous permettre d'écrire des applications particulièrement performantes. L'utilisation d'Interbase est présentée plus en détail au chapitre 11.

MyBase

Delphi 7 Studio a été l'occasion pour Borland d'un « relookage » marketing des noms de techniques proposées parfois depuis plusieurs versions (ce changement a commencé avec la version 6). On l'a vu pour Midas, qui est devenu DataSnap (avec quelques changements techniques toutefois), et il en est de même pour MyBase qui était déjà accessible sous Delphi 5 sous le vocable de « modèle Briefcase » du `TClientDataset`.

Il s'agit en fait de la capacité de ce composant, originellement utilisé pour créer des applications 3-tiers avec Midas, de pouvoir sauvegarder en local dans un fichier binaire les données qu'il reçoit d'un serveur. Le « modèle Briefcase » consiste à dire que, si on n'exploite cette capacité que lors de la sauvegarde/relecture de données locales sans connexion à un serveur d'application, on obtient une mini-gestion de base de données ne réclamant rien d'autre que la DLL Midas en plus de l'application.

Pour remplacer le BDE, Borland se devait de proposer des équivalences. Pour le mode partagé (multi-utilisateur), la fourniture d'Interbase 6 en mode Open Source (et l'existence de FireBird tout aussi gratuit) ainsi que des composants IBX s'affranchissant du BDE est une excellente solution, bien plus riche techniquement que ne l'était Paradox. Pour les applications monopostes, il restait à trouver une solution. MyBase, en reprenant les potentialités du `TClientDataset` de DataSnap et en n'imposant pas de licence pour le déploiement, peut être substitué au BDE pour les petites bases de données.

Dans le cadre de bases de données dépassant plusieurs milliers d'enregistrements (ce qui reste fort modeste), on préférera d'emblée travailler avec Interbase et IBX. On se simplifiera la vie pour un portage en client-serveur (normalement, il n'y a rien à faire) et on disposera tout de suite d'une véritable base de données, sans compter la portabilité vers Linux.

Toutefois, pour de petites applications devant stocker des données persistantes, MyBase est une solution séduisante et bien adaptée à ce type de besoin. Les applications en Shareware ou Freeware distribuées par Internet sont de bonnes candidates à l'utilisation de MyBase.

On notera que l'exploitation du `TClientDataset` de DataSnap offre une nouvelle possibilité à MyBase, à savoir le support du format XML pour les fichiers de données.

Si le format XML est bien trop lourd et verbeux pour une exploitation en base de données dans la majorité des cas, cette ouverture de MyBase est particulièrement intéressante, ne serait-ce que pour offrir un support d'importation et d'exportation de données XML dans vos applications. C'est une solution simple pour répondre à ce besoin de plus en plus fréquent, les formats Paradox, dBase ou même ASCII laissant la place désormais à XML comme format pivot d'échange de données entre applications.

Les composants MyBase

Figure 9-8
Le composant MyBase

Mybase exploite la capacité du `TclientDataset`, déjà présenté, à lire et écrire des données en local dans un format soit binaire, soit XML. On utilise ici ce composant en place et lieu d'un composant `TTable` du BDE. On dispose de nombreuses possibilités telles que celles de définir des index, des filtres, etc. L'utilisation de MyBase est détaillée au chapitre 12.

Récaputilatif des solutions d'accès aux données

Le tableau 9-1 regroupe l'ensemble des solutions évoquées, le tout assorti d'informations et conseils sur chacune.

Tableau 9.1. Récapitulatif des solutions d'accès aux données

	BDE	ADO	dbExpress	DataSnap	IBX	MyBase
Poids du middleware	Lourd	Lourd	Léger	Léger/ultra-léger	Aucun (compilé dans l'application)	Ultra-léger
Gestion SQL	Oui	Oui	Oui	Oui	Oui	Non
Taille des bases utilisables	Grande	Grande	Grande et très grande	Grande et très grande	Grande et très grande	Petite
Richesse des composants	Moyenne	Moyenne	Moyenne	Moyenne	Grande	Faible
Portabilité	Windows	Windows	Windows/Linux	Windows/Linux	Windows/Linux	Windows/Linux
Conseil	Ne plus utiliser sauf pour Paradox et dBase	Middleware Microsoft avec les inconvénients liés.	Parfait pour le 3-tiers. Remplace le BDE pour le 2-tiers.	Transport des données en 3-tiers et réalisation de clients légers.	À préférer pour Interbase et FireBird.	Pour les utilitaires ou micro-applications.

Le poids du middleware indique la taille des drivers et DLL nécessaires pour exploiter la méthode choisie. La partie cliente de la base de données n'est pas prise en compte car elle est systématiquement nécessaire à l'exception des clients légers créés avec DataSnap. Pour sa part, le serveur DataSnap doit toutefois pouvoir dialoguer avec la librairie de la base de données.

La richesse des composants est généralement moyenne mais le plus souvent suffisante. Seuls les composants IBX sortent du lot en raison de leur parfaite adaptation à leur unique cible, Interbase (notamment en raison des composants spécifiques pour l'administration de la base).

Ce tableau ne représente pas un classement ni ne crée de hiérarchie qualitative, chacune des solutions est intéressante et bien adaptée à certains types de développement. L'erreur pourrait seulement consister à les utiliser en dehors de ce cadre (par exemple, choisir le BDE aujourd'hui pour une base SQL ou MyBase pour une application dont les volumes de données évolueront grandement).

Conclusion

Delphi propose un très vaste choix d'accès aux données, soit en ouvrant des portes vers des systèmes existants comme ADO, soit en proposant une gestion directe de certains serveurs comme Interbase/FireBird, soit en fournissant une solution totalement intégrée comme MyBase.

Le développeur n'a hélas que fort rarement son mot à dire dans le choix de la base de données imposée par le client ou le service commercial de la SSII ; c'est dommage mais c'est ainsi… De par son ouverture à l'ensemble des techniques présentes sur le marché, des meilleures aux autres, Delphi évite d'être un jour piégé face à une demande particulière. Qu'il s'agisse d'accéder à une base Access ou de programmer une application multi-tiers ouverte sur le Web, vous trouverez toujours l'outil adapté à vos besoins.

Toutefois, lorsque vous avez le choix de la base ou la possibilité d'influencer ce choix, et pour exercer pleinement ce rôle, apprenez à bien cerner chaque méthode d'accès proposée par Delphi ; vos développements n'en seront que plus sûrs, plus rapides et fiables, et vous serez à même de conseiller efficacement l'utilisateur final.