Advantage Database Server pro programátory ve Visual FoxPro

Doug Hennig

Úvod

Visual FoxPro je báječný vývojový nástroj. Díky své objektové orientaci, bohatému jazyku, integrovanému generátoru sestav a otevřenému a rozšiřitelnému interaktivnímu vývojovému prostředí (IDE) je to jeden z nejlepších dostupných nástrojů pro vývoj desktopových aplikací. Vestavěný datový stroj Visual FoxPro je však současně jednou z nejsilnějších i nejslabších jeho stránek. Jeho síla je v tom, že je pevně integrován do VFP a je nesmírně rychlý. Na druhé straně soubory DBF mohou být snadno pozměněny, jsou málo bezpečné a mají omezenou velikost. Naštěstí foxaři nejsou nuceni ukládat data jen do tabulek VFP; VFP je výborné prostředí pro přístup typu klient/server k databázím jako SQL Server, Oracle nebo Sybase.

Tento článek představuje další produkt na databázovém trhu: Advantage Database Server. Napřed ukáže jeho vlastnosti a funkce a potom porovná jednotlivé způsoby, jak k němu přistupovat z aplikací VFP. Protože pro některé čtenáře mohou být technologie klient/server poměrně málo známé, zabývá se článek přístupem ke vzdáleným databázím podrobněji a ukazuje, jak se to dělá.

Co je Advantage Database Server

Advantage Database Server, zkráceně ADS, pochází z dílny Sybase iAnywhere, což je dnes součást společnosti Sybase. Podle propagačních materiálů je Advantage Database Server „bohatě vybavený a velmi výkonný systém pro správu dat v prostředí klient/server, navržený speciálně pro programátory aplikací“. Čím víc o něm čtete, tím víc zjišťujete, že jeho vlastnosti a funkce velmi dobře doplňují databázový stroj Visual FoxPro; nejde však o náhradu VFP. Podobně jako SQL Server, i ADS je databázový stroj, nikoliv plnohodnotný programovací jazyk, a lze k němu z VFP snadno přistupovat s využitím ODBC nebo ADO. Jak se ale ukáže v dalším textu, ADS podporuje VFP lépe, než je tomu u jiných databázových strojů, a jeho poslední verze 9 tuto podporu významně rozšiřuje.

Přehled vlastností ADS v porovnání s VFP:

- Je to opravdový databázový stroj typu klient/server. U souborově orientovaných strojů, jako je VFP, je server obsahující datové soubory jenom file server. Veškeré zpracování, například výběr záznamů, se odehrává na pracovní stanici, takže je nutno nejprve data stáhnout ze serveru na stanici. Naopak u strojů typu klient/server probíhá zpracování rovnou na serveru a pak jsou odeslány na stanici jen výsledky. To přináší mnohé výhody, včetně sníženého provozu po síti a lepších možností správy databází. Navíc je tento stroj vícevláknový a schopný využívat více procesorů, takže se lépe přizpůsobí vyššímu zatížení.
- ADS má ve skutečnosti dva databázové stroje: lokální a vzdálený. Lokální stroj není opravdový databázový server, spíš se podobá VFP, protože používá souborově orientovaný přístup k datům. Využívá „in-process“ knihovnu DLL, která se načte do ODBC driveru na klientském počítači. Vzdálený stroj je opravdový databázový server poskytující všechny výhody architektury klient/server. Advantage Local Server je užitečný pro testování, pro samostatného vývojáře nebo jako levný (je totiž zadarmo) databázový stroj pro komerční aplikace, má však oproti Advantage Remote Serveru několik důležitých omezení. Jeho přínosem je zejména to, že zajišťuje mechanismus klient/server, který pak lze v případě potřeby snadno aplikovat na plnohodnotném vzdáleném serveru.
- Ke vzdálenému serveru se dá přistupovat i po Internetu, je-li to třeba. Pro vyšší bezpečnost a výkon umožňuje přenos zašifrovaných a zkomprimovaných dat.
- Jednou z nejzajímavějších vlastností ADS je, že může pracovat s daty uloženými buď ve vlastním formátu (soubory s příponou ADT) nebo v souborech DBF. Data v souborech ADT mají své výhody, například další typy dat, které se v souborech DBF nedají použít, naopak soubory DBF usnadňují převod existující aplikace VFP na model klient/server. Opravdu zajímavé je však to, že ke svým souborům DBF můžete přistupovat prostřednictvím ADS, abyste využili jeho přínosů, a přitom je dál můžete používat přímo jako tabulky VFP. Díky tomu lze zvolit velmi atraktivní

strategii migrace aplikací: můžete převádět na model klient/server postupně jeden modul za druhým, protože dosud nepřevedené moduly budou dál fungovat beze změny.

- Při přístupu k souborům DBF jsou k dispozici dva mechanismy zamykání: kompatibilní (compatible) a interní (proprietary). Kompatibilní zamykání využívá zámky operačního systému na bytech souborů DBF a dovoluje, aby k datům souběžně přistupovaly jak ADS, tak jiné aplikace (např. VFP). Interní zamykání využívá mechanismus zamykání, který zaručuje lepší stabilitu, ale ADS pak otevírá soubory pro výlučné použití a dokud je nezavře, nemůže s nimi pracovat jiná aplikace.
- Bezpečnost databází je zajištěna tím, že pro připojení k databázi je nutný platný uživatelský účet. K datům tedy mohou přistupovat jen oprávnění uživatelé. Různé uživatelské účty mohou mít různá přístupová práva. Není například pravděpodobné, že by běžní uživatelé potřebovali vytvářet či mazat tabulky nebo měnit jejich strukturu. Proto můžete v provádění těchto zásahů do databáze zabránit všem kromě administrátorů. I v případě, že využíváte ADS k práci se soubory DBF, můžete tyto soubory umístit do adresáře na serveru, k němuž běžní uživatelé nemají přístup, takže se k datům dostanou jedině prostřednictvím ADS. To by se v čistě foxovském řešení implementovalo mnohem obtížněji.
- Pro zvýšení bezpečnosti může ADS tabulky zašifrovat s využitím hesla rozlišujícího velká a malá písmena. K něčemu takovému je v čistě foxovském řešení nutný další produkt, například Cryptor od firmy Xitech, a vlastní ošetření přístupu k zašifrovaným datům.
- Podobně jako ve VFP, i tabulky ADS mohou být buď volné, nebo vázané na datový slovník (soubor ADD), přičemž ani zde soubor ADD „neobsahuje“ jednotlivé tabulky, jen o nich udržuje doplňkové informace neboli meta data. Datový slovník ADD je velmi blízký databázovému kontejneru DBC, obstarává např. dlouhé názvy polí, primární klíče, pravidla referenční integrity, výchozí hodnoty polí, ověřovací pravidla a vlastní chybová hlášení pro pole (i když ADS nedokáže zpracovat výrazy umožňující libovolné ověřování – umí kontrolovat jen minimální a maximální hodnoty a hodnotu null), ověřovací pravidla a vlastní chybová hlášení pro tabulky, pohledy, triggery a uložené procedury. Jak se dočtete dále, ADS nabízí utilitu, která při generování datového slovníku ADD ze souboru DBC udělá většinu práce za vás.
- Ačkoliv se v dokumentaci k ADS vyskytuje termín „Advantage optimized filters“, popis této technologie velmi výkonných výběrů vypadá úplně stejně jako popis technologie Rushmore, která zajišťuje vysokou rychlost VFP: ke zjištění, které záznamy odpovídají filtrovacím podmínkám, prozkoumává ADS jen index a k fyzickým záznamům přistupuje pouze v případě, že index není dostupný. Stejně jako v dokumentaci k VFP se i u ADS vyskytují pojmy jako „plně optimalizovaný“ a „částečně optimalizovaný“. Z toho plyne, že foxaři mohou při optimalizaci výběrů z databází ADS využít svých dosavadních znalostí z VFP.
- V ADS je k dispozici fulltextové vyhledávání, zajišťující velmi rychlé prohledávání polí memo. Mnozí foxaři k tomu využívají ve svých aplikacích produkty jiných výrobců jako PhDbase, ale některé z nich už nejsou dostupné nebo nebyly aktualizovány pro poslední verze VFP.
- Ačkoliv může ADS přistupovat k souborům DBF, nepodléhá stejným omezením jako VFP. Například ve VFP mohou být soubory DBF a FPT velké maximálně 2 GB. V ADS není velikost souborů omezena přímo, jediným limitem je počet záznamů, který nesmí překročit cca 2 miliardy (2 147 483 648). Samozřejmě pokud tabulka DBF přesáhne 2 GB, VFP ji bude považovat za neplatnou a budete k ní moci přistupovat jen prostřednictvím ADS.
- Protože ODBC driver ADS plně podporuje datové typy VFP 9, můžete ho používat místo ODBC driveru VFP, který byl naposled aktualizován pro VFP 6, takže neumí pracovat s novinkami, jako jsou Varchar, Varbinary a Blob.
- ADS zvládá transakce, jejich dokončení s potvrzením (commit) nebo odvoláním (rollback) a automatické odvolání transakce, jestliže stanice nebo server během jejího zpracovávání zhavaruje.
- Replikace je proces přenášející změny záznamů v tabulkách jedné databáze do tabulek jiné databáze, například změny provedené v databázích na vzdálených pobočkách do sloučené databáze na centrále a naopak. Ve VFP si musíte replikace naprogramovat sami, vypořádat se přitom s nejrůznějšími problémy, třeba s ošetřením konfliktů, a důkladně celé řešení otestovat, aby

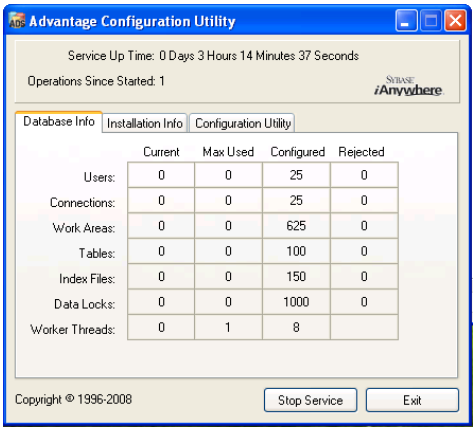
správně fungovalo v každé situaci. ADS má vestavěné replikační funkce, takže se nemusíte namáhat.

- ADS zahrnuje funkci zálohování za chodu (online backup), což znamená, že lze zálohovat i tabulky otevřené nějakou aplikací – to u tabulek VFP zálohovaných běžnými procedurami není možné. Lze provádět jak plné, tak inkrementální zálohy.

Instalace Advantage Database Serveru

Celý produkt tvoří tyto součásti: samotný databázový server, Advantage Data Architect, ODBC driver a OLE DB provider. V době vzniku tohoto článku (duben 2008) byla na světě jen beta verze ADS 9, ke stažení na http://devzone.advantagedatabase.com, a plná verze měla vyjít co nevidět. [pozn. z překladu v listopadu 2008: ke stažení je již i plná verze 9.0 a její aktualizace 9.1]

ADS běží na platformách Windows, Netware a Linux. Instalační soubor databázového serveru pro Windows se jmenuje NT.EXE. Spusťte tento program na serveru, kam se má ADS nainstalovat. Samozřejmě ho můžete nainstalovat na svůj vývojový systém a ne na samostatný server, ale za normálních okolností byste ho měli instalovat na skutečný server v produkčním prostředí. Výchozím adresářem pro instalaci databázového stroje (a také ostatních komponent) je C:\Program Files\Advantage 9.0. Po instalaci souborů si instalační program vyžádá tato nastavení: jméno registrovaného vlastníka, zda se má příslušná služba Windows spouštět automaticky nebo ručně (výchozí hodnota je automaticky), která znaková sada ANSI se má použít (výchozí hodnota je výchozí nastavení daného stroje) a která znaková sada OEM/localized se má použít. Po zadání těchto hodnot se otevře okno Advantage Configuration Utility (viz **Obrázek 1**) se statistickými údaji o serveru, včetně počtu uživatelů a připojení, v němž můžete nakonfigurovat určité vlastnosti jako timeout, používané porty a umístění souborů s logy.



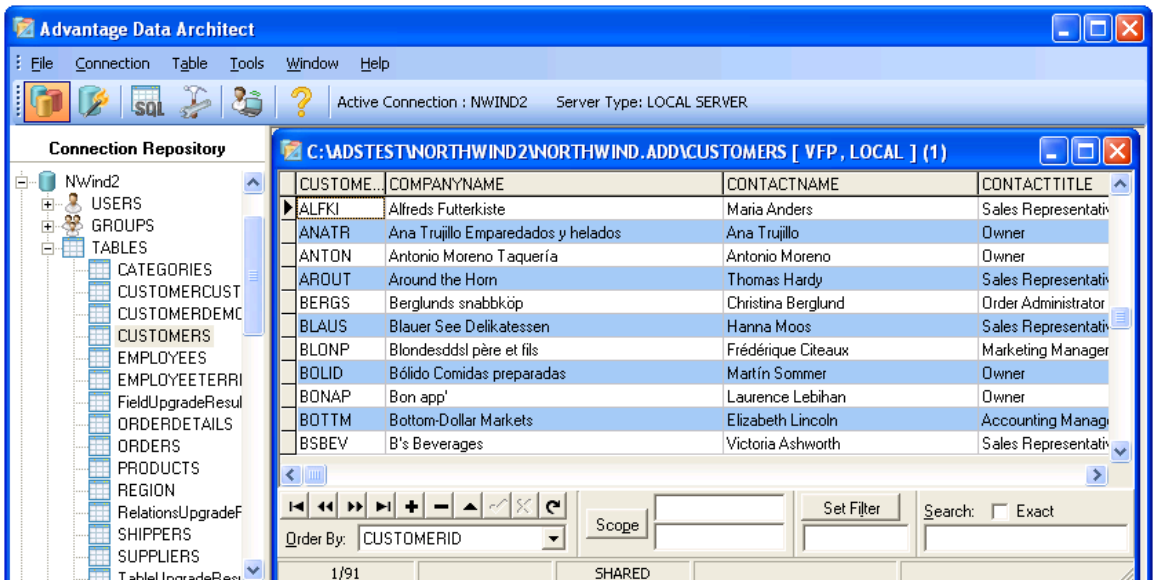
Obrázek 1. Po instalaci ADS se zobrazí okno Configuration Utility, v němž můžete nakonfigurovat vlastnosti serveru.

Jako další v pořadí se instaluje Advantage Data Architect, utilita ADS popsaná v následující sekci. Instalační soubor se jmenuje Arc32.EXE. Stejně jako při instalaci serveru i zde můžete zadat název adresáře pro instalaci, znakovou sadu ANSI a znakovou sadu OEM. Potom nainstalujte ODBC driver ze souboru ODBC.EXE, pokud hodláte přistupovat k ADS prostřednictvím ODBC, a OLE DB provider ze souboru OLEDB.EXE (ten byste měli nainstalovat bez ohledu na to, zda se chystáte používat ADO nebo ne, protože OLEDB.EXE nainstaluje jednu utilitu specifickou pro VFP, o níž se dočtete níže). Oba instalační programy se zeptají na název adresáře pro instalaci, znakovou sadu ANSI a znakovou sadu OEM.

A teď už se můžete pustit do práce s ADS.

Advantage Data Architect

Advantage Data Architect, zkráceně ARC, je nástroj ADS pro správu spojení a databází. Pokud používáte Enterprise Manager nebo Management Studio v SQL Serveru, případně Data Explorer ve VFP, bude vám tato utilita připadat docela povědomá. Zajímavé je, že je v ní zahrnut i kompletní zdrojový kód, napsaný v Delphi. ARC ukazuje **Obrázek 2**.



Obrázek 2. Advantage Data Architect nabízí podobné funkce jako Data Explorer ve VFP nebo Management Studio v SQL Serveru.

ARC umožňuje:

- Vytvářet, udržovat a odstraňovat databáze a tabulky
- Procházet tabulky a využívat při tom filtry, prohledávání, řazení a přímou navigaci
- Importovat a exportovat data
- Exportovat struktury tabulek jako kód
- Spravovat bezpečnostní nastavení a uživatelské účty
- Spouštět dotazy v nástrojích SQL Utility a Query Builder
- Porovnávat datové slovníky

Levé podokno ARC je Connection Repository (Přehled spojení). Zajišťuje snadný přístup k databázím ADS zaregistrovaným v ARC. (Používat databázi v jiných aplikacích je možné i bez registrace v ARC.) Chcete-li vytvořit novou databázi a přidat ji na seznam, vyberte z nabídky File položku Create New Data Dictionary; tím vznikne prázdný soubor ADD, jehož vlastnosti zadáte v dialogu, který se zobrazí. Chcete-li na seznam přidat existující databázi nebo adresář s volnými tabulkami, vyberte z nabídky File položku New Connection Wizard a dál postupujte podle pokynů v příslušném dialogu.

Po celém tomto článku jsou příklady využití různých funkcí ARC.

Převod databáze VFP

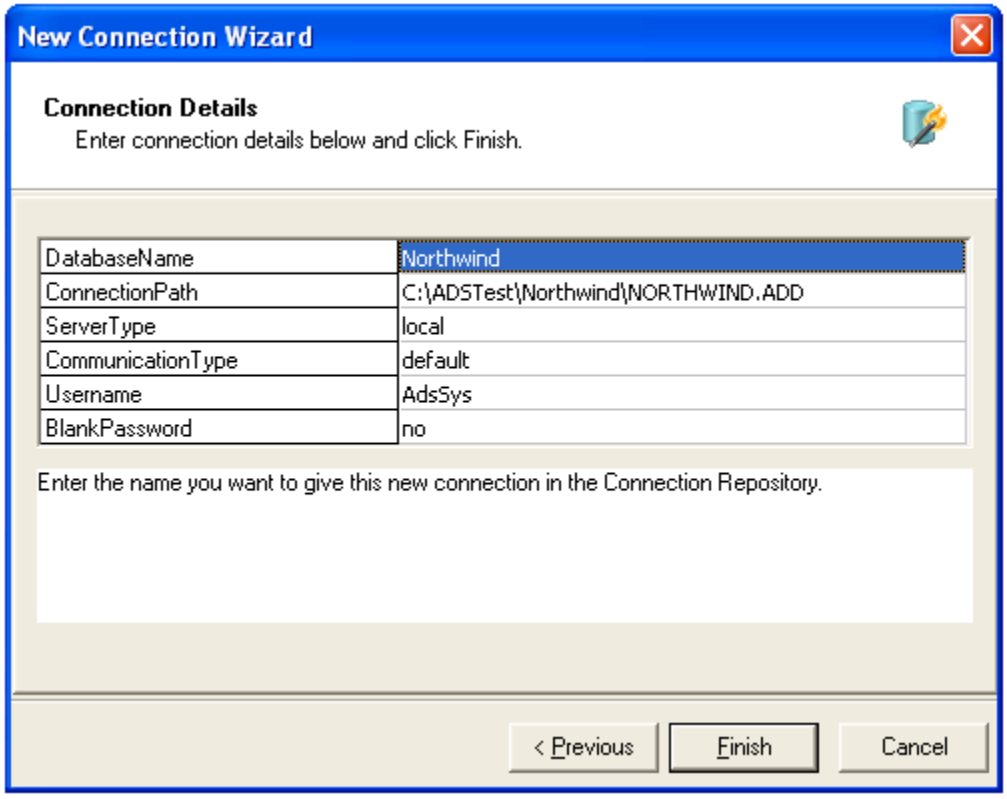
ADS 9 sice podporuje většinu vlastností dat VFP, některé z nich se však uplatní jen v případě použití databáze ADS, nikoliv „volných“ tabulek. (Z pohledu ADS jsou považovány za „volné“ i tabulky foxovského databázového kontejneru DBC, nejsou-li začleněny do databáze ADS.) Patří mezi ně dlouhé názvy polí, primární klíče, referenční integrita, ověřovací pravidla pro pole a tabulky, triggery atd.; jinými slovy tytéž záležitosti, kvůli kterým se ve VFP používá databázový kontejner. Nejde-li o malinkou databázi, bylo by vytvoření databáze ADS k existující databázi VFP dost pracné. Naštěstí je ADS vybaven utilitou ADSUpsize.PRG napsanou ve VFP, která vytvoří databázi ADS a naplní ji informacemi o tabulkách databáze VFP. Název ADSUpsize.PRG je poněkud zavádějící, protože program „nepovyšuje“ soubor DBC ani neprovádí žádné změny v tabulkách DBF. V ostré verzi ADS 9 Sybase iAnywhere přejmenoval tuto utilitu na DBCConvert.PRG.

Abychom se seznámili s tím, jak ADSUpsize.PRG funguje, převedeme ukázkovou databázi Northwind, která se dodává s VFP. Začneme založením nového adresáře, do něhož zkopírujeme všechny soubory z podadresáře Samples\Northwind domovského adresáře VFP; tak zůstanou původní soubory nedotčené, až budeme provádět změny popsané níže. Spustíme z VFP program ADSUpsize.PRG, který je součástí příloh k tomuto článku; oproti originálnímu programu dodávanému s beta verzí zahrnuje úpravy popsané v odstavci „Jak opravit utilitu ADS pro převod databáze VFP“. Jako zdrojovou databázi vybereme Northwind.DBC v novém adresáři, do něhož jsme zkopírovali soubory. Program se se všemi úkoly vypořádá za chviličku a zobrazí hlášení: vyskytlo se 15 chyb, jenže se vůbec nedozvíme, jakých. Naštěstí ADSUpsize.PRG zapisuje průběh do protokolu, jak si ukážeme vzápětí.

Když se podíváme do adresáře s databází Northwind, uvidíme několik nových souborů:

- Northwind.ADD, AI a AM: to je databáze ADS.
- Záložní verze všech souborů DBF a FPT: tyto soubory pro jistotu zálohuje sám ADS (nikoliv ADSUpsize.PRG).
- FieldUpgradeResults.ADM a ADT, RelationsUpgradeResults.ADM a ADT, TableUpgradeResults.ADM a ADT, ViewUpgradeResults.ADM a ADT: tyto tabulky ADS obsahují protokoly s informacemi o průběhu převodu, včetně záznamu o všech chybách, které se vyskytly. Budete je intenzivně využívat k nalezení a vyřešení problémů s převodem.
- FAIL_EMPLOYEES.DBF a FPT: viz níže.

Spustíme ARC a z nabídky File vybereme New Connection Wizard nebo klikneme na tlačítko New Connection Wizard v panelu nástrojů. V prvním kroku vybereme „Create a connection to an existing data dictionary“. Ve druhém kroku zadáme název databáze a cestu k databázi Northwind.ADD, kterou vytvořila převáděcí utilita. U ostatních položek ponecháme výchozí hodnoty a klikneme na tlačítko Finish (viz **Obrázek 3**). ARC nás vyzve, abychom se přihlásili jako uživatel AdsSys (to je výchozí jméno administrátora); protože v tomto příkladu nemá tento uživatel žádné heslo, klikneme na OK.



Obrázek 3. Volbou New Connection Wizard programu Advantage Database Architect se vytvoří připojení k převedené databázi VFP.

Pod Tables se objeví všechny tabulky databáze Northwind, ale pod Views vidíme jen pět ze sedmnácti pohledů. Navíc jsou tam [kde?] čtyři nové tabulky zmíněné výše, FieldUpgradeResults, RelationsUpgradeResults, TableUpgradeResults a ViewUpgradeResults.

Poklepáním otevřeme TableUpgradeResults. Každá převedená tabulka se zde vyskytuje několikrát, za každou operaci jednou. Různé sloupce této tabulky ukazují, jaký proces byl proveden v každém kroku. Například pokud jde o tabulku Customers, jsou zde záznamy pro přidání tabulky do databáze ADS, stanovení dlouhých názvů polí, vytvoření indexů, stanovení ověřovacích pravidel a hlášení pro tabulku a definování primárního klíče. Všimněte si, že u jedné z tabulek, Categories, došlo k chybě v kroku stanovení dlouhých názvů polí. Chybové hlášení vypadá takto (upraveno kvůli místu):

```
The requested operation is not legal for the given field type.   ALTER TABLE CATEGORIES
ALTER "CATEGORYID"  "CATEGORYID" autoinc  NOT NULL ALTER "CATEGORYNA"  "CATEGORYNAME"
char( 15 ) NOT NULL ALTER "DESCRIPTIO"  "DESCRIPTION"  memo  NULL ALTER "PICTURE"
"PICTURE" blob  NULL
```

Otevřeme FieldUpgradeResults. Tato tabulka ukazuje výsledky převodu komentářů, ověřovacích pravidel a výchozích hodnot polí. Opět je zde jeden záznam s chybou, oznamující, že utilita nedokázala nastavit komentář pro Categories.CategoryName; toto pole v datovém slovníku neexistuje, protože chyba zaznamenaná v tabulce TableUpgradeResults zabránila definování dlouhých názvů polí tabulky Categories, takže pole se ve skutečnosti jmenuje CategoryNa, což je 10znakový název uložený v hlavičce DBF.

RelationsUpgradeResults obsahuje informace o převedených relacích. V této tabulce nejsou zaznamenány žádné chyby. Naopak tabulka ViewUpgradeResults s informacemi o převedených pohledech obsahuje spoustu chyb. Ve skutečnosti se podařilo převést pouhých pět pohledů. Některé ze zbývajících pohledů se odkazují na neexistující pole Categories.CategoryName, ale je tu i několik jiných příčin, proč převod ostatních pohledů neprošel.

Když si ve VFP nastavíme SET DELETED OFF a otevřeme tabulku Employees, vidíme, že jeden ze záznamů, Andrew Fuller, je vyřazen. Když otevřeme tabulku FAIL_EMPLOYEES vytvořenou při převodu, najdeme v ní tento záznam. Proč byl při převodu vyřazen? Hned uvidíme, co se stalo.

I když převod databáze VFP znamená výrazný posun směrem k dokončení databáze ADS, je nutno mít na paměti několik bodů.

- ADS nezná vnořené JOINy typu SELECT * FROM Table1 JOIN Table2 JOIN Table3 ON Table2.Field = Table3.Field ON Table1.Field = Table2.Field. Před převodem je potřeba u pohledů, které používají tuto syntaxi, změnit vnořené JOINy na obvyklejší sekvenční. Jednou z možností, jak to udělat, je otevřít pohled v Návrháři pohledů (View Designer) a znovu ho uložit. To funguje díky tomu, že zatímco ve starších verzích VFP používal View Designer vnořenou syntaxi, od verze VFP 8 je už výchozí syntaxe sekvenční.
- ADS nepodporuje pohledy s frází ORDER BY.
- Pohledy obsahující funkce VFP se nepřevedou správně. Například pohled Sales_Totals_By_Amount v databázi Northwind používá v jedné z podmínek WHERE foxovskou funkci BETWEEN(). V tomto případě lze problém vyřešit náhradou za BETWEEN jazyka SQL. Jiné pohledy se však nedají opravit tak snadno. Například ve frázi WHERE pohledů Summary_of_Sales_by_Year a Summary_of_Sales_by_Quarter se vyskytují funkce EMPTY() a NVL(), takže je nelze převést a musí být znovu vytvořeny ručně.
- Tabulky s poli typu General se nepřevedou správně, protože ADS nepodporuje tento datový typ (další důvod, proč ho nepoužívat). To je příčinou chyby v tabulce Categories: pole Categories.Picture je typu General. Když se pokusíte zobrazit strukturu této v ARC (klikněte na tabulce pravým tlačítkem a vyberte Properties), dostanete chybové hlášení a pro toto pole se nezobrazí žádné vlastnosti. Musíte buď odstranit pole Picture z tabulky, nebo změnit jeho typ, třeba na Blob.
- Z foxovských vlastností vázaných na pole podporuje ADS jen výchozí hodnotu, zda je povolena hodnota NULL, popis (ten se převede z vlastnosti komentář k poli), ověřovací zprávu pole a zda se má obsah překládat do jiné kódové stránky. Tyto vlastnosti se tedy převedou. ADS nepodporuje ověřovací pravidla pole, pouze minimální a maximální hodnotu, takže můžete po dokončení převodu doplnit tyto hodnoty ručně.

- Ostatní vlastnosti polí (formát, vstupní maska, mapování pole a titulek) se nepřevedou, protože je ADS nepodporuje. Všechny tyto vlastnosti se vztahují k uživatelskému rozhraní a proto nejsou v ADS nutné.
- Z foxovských vlastností vázaných na tabulku podporuje ADS velikost bloku memo, popis tabulky (převádí se z vlastnosti komentář k tabulce), ověřovací pravidlo tabulky a ověřovací zprávu, všechny tyto vlastnosti se tedy převedou. Pravidla obsahující funkce VFP však obvykle představují problém.
- Triggery se nepřevedou, používají totiž kód VFP, kterému ADS nerozumí. Nejběžnější uplatnění triggerů je ovšem pro dodržení pravidel referenční integrity a ta se převedou. Pozor, ADS nepodporuje pravidlo Ignore RI, proto bude převedeno jako Set to NULL. Triggery určené k jiným účelům budete muset vytvořit znovu.
- Ze stejného důvodu se nepřevedou ani uložené procedury. O procedury určené k zachování referenční integrity se nemusíte starat, protože pravidla RI se převedou. Všechny ostatní uložené procedury však budete muset přepsat do jazyka SQL ADS, případně přesunout kód do nějaké komponenty ve střední vrstvě.
- Během převodu se uplatňuje referenční integrita. Ta způsobila, že byl záznam Andrew Fuller tabulky Employees vyřazen. Tabulka Employees má sloupec ReportsTo obsahující EmployeeID nadřízeného každé osoby a u Andrewa je v tomto sloupci hodnota 0, která neexistuje ve sloupci EmployeeID v žádném záznamu. Pravidlo referenční integrity pro Insert je u této vazby typu self-join nastaveno na Ignore, takže ve VFP nedojde u tohoto záznamu k žádné chybě. ADS však nemá pravidlo referenční integrity pro Insert, jen pro Update a Delete, a ta jsou obě nastavena na Restrict. Protože má Andrew ve sloupci ReportsTo neplatnou hodnotu, pravidlo RI selže a záznam je vyřazen. Můžete namítnout, že je to docela drsné; možná by stačilo, kdyby ADS místo toho zapsal do sloupce ReportsTo hodnotu NULL. Zajímavé je, že obnovení záznamu ve VFP funguje.
- ADS verze 9 nepodporuje binární indexy VFP, ale Sybase má v plánu jejich podporu ve verzi 9.1 případně v service packu vydaném ještě před verzí 9.1.

Můžeme některé problémy opravit a pokusit se o nový převod. Protože ADSUpsize.PRG vytvoří databázi ADS, tuto databázi nelze otevřít v ARC [??], takže zavřeme připojení na Northwind: klepneme pravým tlačítkem na připojení a z místní nabídky vybereme Disconnect. Otevřeme databázi Northwind ve VFP a provedeme tyto změny:

- Invoices a Product_Sales_For_1997: tyto pohledy upravíme tak, že z nich odstraníme relace, a pak je vytvoříme znovu. Tyto pohledy používají vnořené vazební podmínky a jejich opětovným vytvořením ve VFP 9 se tato syntaxe převede na sekvenční. (Ověříme si to volbou View SQL v návrháři pohledů.) Totéž bychom sice mohli udělat i s pohledem Sales_By_Category, ale jak si ukážeme za chvíli, tento pohled se nedá převést z jiných důvodů.
- Quarterly_Orders a Sales_Totals_By_Amount: tyto pohledy stačí otevřít a znovu uložit bez provádění jakýchkoliv změn. Tyto pohledy používají foxovskou funkci BETWEEN() v jedné z podmínek WHERE. Při uložení dojde ke změně na frázi BETWEEN jazyka SQL. (V pohledech Product_Sales_For_1997 a Sales_By_Category se také vyskytuje funkce BETWEEN(), ta však byla automaticky převedena, když jsme je uložili v předchozím kroku.)
- Alphabetical_List_of_Products, Current_Product_List, Invoices a Products_By_Category: z těchto pohledů odstraníme frázi ORDER BY. To bohužel nemůžeme udělat s pohledem Ten_Most_Expensive_Products, protože ten obsahuje frázi TOP, která frázi ORDER BY vyžaduje. Tento pohled budeme muset znovu vytvořit ručně. A i kdybychom tutéž změnu provedli v pohledech Sales_By_Category, Summary_of_Sales_by_Quarter a Summary_of_Sales_by_Year, nepomohlo by to, protože se nedají převést z jiných důvodů.
- Invoices: otevřeme tento pohled, zvolíme View SQL a změníme oba výskyty funkce ALLTRIM na TRIM, protože funkci ALLTRIM(), na rozdíl od funkce TRIM(), ADS nepodporuje.
- Categories: z této tabulky odstraníme pole Picture.

Zavřeme všechny tabulky (CLOSE TABLES ALL) a znovu spustíme ADSUpsize.PRG. Tentokrát se vyskytnou jen čtyři chyby. Otevřeme v ARC připojení k databázi Northwind a nahlédneme do tabulek s protokoly. Tabulka Categories se převedla správně, takže všechny chyby jsou v pohledech:

- Sales_by_Category se nedá převést, protože provádí výběr z jiného pohledu, což ADS nepodporuje.
- Ten_Most_Expensive_Products používá frázi TOP a proto vyžaduje frázi ORDER BY, což ADS nepodporuje.
- Summary_of_Sales_by_Year a Summary_of_Sales_by_Quarter používají ve frázi WHERE funkce EMPTY() a NVL().

Přístup k datům ve VFP nebo ADS

Převod databáze VFP do ADS neznamená, že už se k ní nedá přistupovat z VFP. Znamená to jen, že teď máte pro přístup k databázi na výběr dvě možnosti: jako k nativním tabulkám VFP nebo prostřednictvím ADS. Změníte-li data ve VFP, uvidíte to i v ADS, a naopak. To se týká i autoinkrementálních polí.

Otevřeme například v ARC převedenou databázi Northwind a poklepáním otevřeme tabulku Employees. Tlačítkem + na panelu nástrojů v dolní části okna s tabulkou (viz Obrázek 2) přidáme záznam. Pole EmployeeID necháme prázdné, ostatní pole však vyplníme; pozor, do pole ReportsTo musíme zadat platnou hodnotu (například „2“), protože pravidla RI pro tuto tabulku nepovolují v tomto poli prázdnou nebo neplatnou hodnotu. Až se z tohoto řádku posuneme jinam, uvidíme, že se do pole EmployeeID automaticky dosadila následující hodnota.

Zavřeme okno s tabulkou a odpojíme se od databáze v ARC. Otevřeme databázi ve VFP a otevřeme tabulku Employees. Vidíme zde nový záznam. Přidáme další a všimneme si, že se do pole EmployeeID opět automaticky dosadilo následující číslo ID. VFP nevyžaduje platnou hodnotu pole ReportsTo; může zůstat prázdné nebo sem můžeme napsat neplatnou hodnotu, třeba „99“, aniž by došlo k chybě, protože pravidlo RI pro Insert je u této vazby typu self-join nastaveno na Ignore.

Jak už bylo uvedeno výše, schopnost přistupovat k tabulkám jak přímo z VFP, tak prostřednictvím ADS umožňuje postupnou migraci existujících aplikací na model klient/server jeden modul po druhém. Například v účetním systému byste pro přístup k datům z ADS upravili modul Pohledávky a po dokončení změn a důkladném otestování byste ho zavedli do provozu. Všechny ostatní moduly by dál přistupovaly k tabulkám přímo z VFP. Výhodou tohoto postupu je, že nemusíte převést celou aplikaci naráz a čelit tak mnohem rozsáhlejšímu zatížení programátorů i testerů, jaké taková hromadná změna vyvolá.

Začínáte-li s novou aplikací, která nemá žádné nároky na zpětnou kompatibilitu dat, možná dáte před soubory DBF přednost nativním tabulkám ADS (souborům ADT). Soubory ADT mají oproti souborům DBF četné výhody, například žádný přebujelý soubor memo, více datových typů (třeba znaková pole nerozlišující velká a malá písmena), delší názvy polí, větší velikost souborů, více než 255 polí v tabulce, automatické opětovné použití vyřazených záznamů.

Fulltextové prohledávání

ADS disponuje rychlým a výkonným fulltextovým prohledáváním. FTS (full text search) využívá index nad každým slovem pole memo a tím zajišťuje rychlé vyhledání požadovaných slov. K uplatnění FTS na tabulce je nutné založit FTS index nad jedním nebo několika poli memo dané tabulky.

Chcete-li si otestovat výkonnost FTS, ale nemáte dost velkou tabulku se spoustou textu v polích memo, pomůže vám program MakeDemoMemo.PRG, který je součástí příloh k tomuto článku. Probere celý váš pevný disk, vyhledá textové soubory (PRG, TXT a HTML) a také foxovské soubory VCX a SCX a nacpe je do pole Content tabulky DemoMemo.DBF. Asi si dokážete představit, že mu to bude chvíli trvat. Jeden test zabral asi deset minut a vytvořil tabulku s 81 000 záznamů a souborem FPT o velikosti 545 MB.

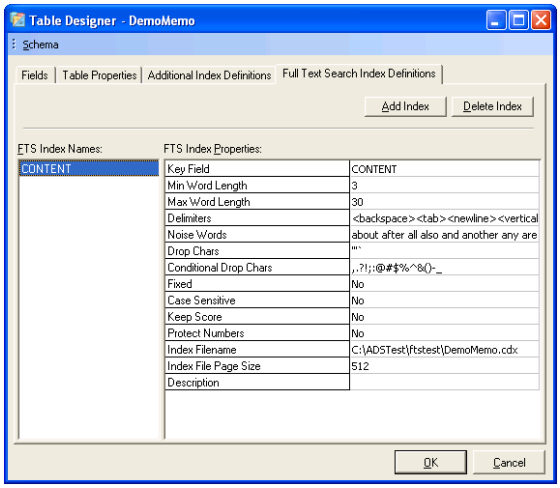
Před přidáním této tabulky do ADS hledal testovací program všechny výskyty slova „tableupdate“ v poli Content:

```
select * from DemoMemo where atc('tableupdate', Content) > 0 into cursor Temp
select * from DemoMemo where 'tableupdate' $ Content into cursor Temp
```

První příkaz, který při hledání nerozlišuje velká a malá písmena, si vyžádal 305 vteřin. Druhý, rychlejší, avšak rozlišující velká a malá písmena, zabral 65 vteřin.

Abychom tuto tabulku připravili pro hledání pomocí FTS, musíme podniknout následující kroky:

- Spustíme ARC a z nabídky File vybereme New Connection Wizard, dále „Create a connection to a directory of existing tables“ a klikneme na Next. Zadáme požadovaný název databáze, vybereme adresář obsahující DemoMemo.DBF, jako TableType označíme „vfp“ a klikneme na OK. (Tím zařídíme přístup k DemoMemo jako k volné tabulce, nikoliv pomocí datového slovníku ADS, protože beta verze má potíže s přístupem k FTS indexům nad soubory DBF pomocí datového slovníku.)
- Pod Tables klikneme pravým tlačítkem na tabulku DemoMemo a z místní nabídky vybereme Properties. Zde se přepneme na kartu Full Text Index Definitions (viz **Obrázek 4**), klikneme na Add Index a zadáme požadovaný název indexu. Jako Key Field zvolíme CONTENT (tedy pole memo obsahující text, který se má indexovat). Index je možno doladit nastavením hodnot dalších vlastností – minimální a maximální délky slova, tzv. „noise words“ (jde o slova jako „and“ či „the“, která se mají z indexování vyloučit), zda má index rozlišovat velká a malá písmena. Volbou OK se index vytvoří.



Obrázek 4. Karta Full Text Search Index Definitions dialogu Table Designer v ARC slouží k tvorbě indexů zajišťujících rychlé a výkonné fulltextové prohledávání.

Vytvoření FTS indexu může nějakou dobu trvat, protože ADS zakládá indexovou položku pro každé slovo v každém záznamu. Jakmile je však index vytvořen, implicitně se automaticky aktualizuje při přidávání, úpravách a mazání záznamů stejně jako jiné indexy. Je ale možné automatické aktualizace vypnout a znovu vytvořit index na vyžádání, což přispívá k lepšímu výkonu při úpravách záznamů.

Po vytvoření FTS indexu si následující příkaz nerozlišující velká a malá písmena, vykonaný strojem ADS, vyžádal pouhých 0.070 vteřiny:

```
select * from DemoMemo where contains(Content, 'tableupdate')
```

Fulltextové prohledávání však dokáže mnohem víc než jen najít určité slovo. Můžete například:

- Hledat slovo ve všech polích, zadáte-li místo názvu pole „*“.
- Hledat vícenásobné výskyty stejného slova v daném záznamu s využití funkce SCORE. Přestože to pro tuto funkci není nezbytné, nastavením vlastnosti Keep Score tohoto indexu na ON dosáhnete vyšší výkonnosti, protože „skóre“ se uloží v indexu.

```
select * from DemoMemo where contains(Content, 'tableupdate') and
  score(Content, 'tableupdate') > 1
```

- Hledat jedno slovo poblíž jiného:

```
select * from DemoMemo where contains(Content, 'tableupdate near aerror')
```

Je důležité si uvědomit, že po vytvoření FTS indexu už nebude možné k dané tabulce přistupovat přímo z VFP. Když se o to pokusíte, dojde k chybě „operation is invalid for a Memo, Blog, General or Picture field“, neboť VFP nepodporuje indexy nad těmito typy polí.

Připojení k ADS

Existují dva způsoby, jak přistupovat k datům, která spravuje ADS: s využitím jeho ODBC driveru nebo jeho OLE DB provideru. (Ve skutečnosti existuje ještě třetí způsob, s využitím jeho funkcí API, ale to jsou nízkoúrovňové funkce a vyžadují spoustu programování.) Pro OLE DB je nutný spojovací řetězec (connection string), ODBC umožňuje použít buď název zdroje dat ODBC (data source name, DSN), nebo spojovací řetězec, který bývá někdy označován jako „spojení bez DSN“ (DSNless connection).

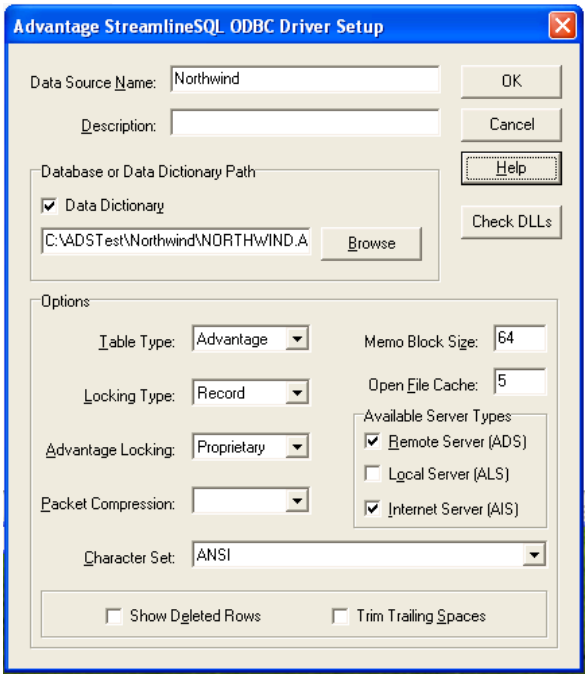
Příklad spojovacího řetězce OLE DB:

```
provider=Advantage.OLEDB.1;data source=C:\ADSTest\Northwind\Northwind.add;
  User ID=adssys;Password=""
```

Příklad spojovacího řetězce ODBC je podobný:

```
driver=Advantage StreamlineSQL ODBC;DataDirectory=C:\ADSTest\Northwind\Northwind.add;
  uid=adssys;pwd=""
```

Chcete-li používat raději DSN než spojovací řetězec, můžete si ho nadefinovat pomocí správy datových zdrojů ODBC. V Ovládacích panelech (Control Panel) Windows otevřete Nástroje pro správu (Administrative Tools) a v nich Datové zdroje (Data Sources) (ODBC). Zde se přepnete na kartu Uživatelské (User) DSN, chcete-li založit DSN jen pro sebe, nebo na kartu Systémové (Systém) DSN, má-li vzniknout DSN pro kohokoliv, kdo se přihlásí k vašemu počítači. Tlačítkem Přidat (Add) vytvoříte nový zdroj dat, z ovladačů vyberete Advantage StreamlineSQL ODBC a kliknete na tlačítko Dokončit (Finish). **Obrázek 5** ukazuje dialog pro nastavení ODBC driveru ADS.



Obrázek 5. Můžete definovat DSN využívající ODBC driver ADS.

Zadejte název zdroje dat a cestu k datovému slovníku ADS. Upravte si nastavení podle svého (zatím tam jsou výchozí hodnoty) a klikněte na OK.

Přístup k ADS pomocí vzdálených pohledů

Tak jako lokální pohled, i vzdálený pohled je prostě jen předem definovaný příkaz SELECT jazyka SQL uložený v databázovém kontejneru. Rozdíl je v tom, že vzdálený pohled nepřistupuje k datům nativně, nýbrž skrze ODBC.

Vzdálený pohled se dá vytvořit buď v programu příkazem CREATE SQL VIEW nebo vizuálně v Návrháři pohledů (View Designer). V obou případech je nutno uvést, které spojení ODBC se má použít. Může to být buď název zdroje dat (DSN) ODBC zadaného v systému nebo objekt Connection, který už byl definován v téže databázi. Následující příklad definuje objekt Connection a vytvoří vzdálený pohled na tabulku Customers převedené databáze Northwind. Jde o výňatek z kódu vygenerovaného utilitou GENDBC; ve skutečnosti je tam mnohem víc kódu nastavujícího různé vlastnosti spojení, pohledu a polí.

```
CREATE CONNECTION NORTHWINDCONNECTION ;
   CONNSTRING "DSN=Northwind"
CREATE SQL VIEW "CUSTOMERSVIEW" ;
   REMOTE CONNECT "NorthwindConnection" ;
   AS SELECT * FROM CUSTOMERS Customers
DBSetProp('CUSTOMERSVIEW', 'View', 'UpdateType', 1)
DBSetProp('CUSTOMERSVIEW', 'View', 'WhereType', 3)
DBSetProp('CUSTOMERSVIEW', 'View', 'FetchMemo', .T.)
DBSetProp('CUSTOMERSVIEW', 'View', 'SendUpdates', .T.)
DBSetProp('CUSTOMERSVIEW', 'View', 'Tables', 'CUSTOMERS')
DBSetProp('CUSTOMERSVIEW.customerid', 'Field', 'KeyField', .T.)
DBSetProp('CUSTOMERSVIEW.customerid', 'Field', 'Updatable', .F.)
DBSetProp('CUSTOMERSVIEW.customerid', 'Field', 'UpdateName', 'CUSTOMERS.CUSTOMERID')
DBSetProp('CUSTOMERSVIEW.companyname', 'Field', 'Updatable', .T.)
DBSetProp('CUSTOMERSVIEW.companyname', 'Field', 'UpdateName', 'CUSTOMERS.COMPANYNAME')
```

Jedním z nejjednodušších postupů, jak převést existující aplikaci, je vygenerovat databázi ADS z databáze VFP pomocí již zmíněné utility ADS, pak založit novou databázi VFP (např. REMOTE.DBC) a v ní vytvořit vzdálené pohledy pojmenované stejně jako tabulky, na nichž jsou pohledy založeny. Kód otevírající vzdálený pohled bude tedy úplně stejný jako kód otevírající lokální tabulku, až na to, že je nutno předem otevřít jinou databázi. Existuje-li například objekt typu aplikace pojmenovaný oApp a ten má vlastnost lUseLocalData, která určuje, zda bude aplikace přistupovat k lokálním nebo vzdáleným datům, následující kousek kódu otevře patřičnou databázi a pak buď tabulku Customers nebo vzdálený pohled Customers:

```
if oApp.lUseLocalData
  open database Local
else
  open database Remote
endif
use Customers
```

Používáte-li v datovém prostředí formulářů a sestav objekty typu kurzor, budete s nimi mít trochu víc práce, protože tyto objekty v sobě nesou odkaz na konkrétní databázi, která byla aktuální při přetahování pohledů do datového prostředí [??]. To ošetříte tak, že do metody BeforeOpenTables datového prostředí umístíte kód podobný tomuto:

```
local loObject
for each loObject in This.Objects
  if upper(loObject.BaseClass) = 'CURSOR' and not empty(loObject.Database)
    loObject.Database = iif(oApp.lUseLocalData, 'local.dbc', 'remote.dbc')
  endif
next
```

Výhody

Výhody vzdálených pohledů jsou:

- Můžete využít návrhář pohledů a vytvořit vzdálený pohled vizuálně. Je báječné vidět všechna pole zúčastněných tabulek, snadno sestavovat různé části příkazu SQL SELECT v přívětivém uživatelském rozhraní a rychle nastavovat vlastnosti pohledu pomocí zaškrtávacích políček a dalších ovládacích prvků.

- Z hlediska jazyka se se vzdálenými pohledy pracuje stejně jako s tabulkami. Z toho plyne, že se dají používat téměř libovolně: otevírat příkazem USE, přidávat do datového prostředí formuláře nebo sestavy, svázat s gridem, procházet v cyklu SCAN atd.
- Převést existující aplikaci tak, aby využívala vzdálené pohledy, zvláště pokud už využívá lokální pohledy, je jednodušší než jiné metody popsané v dalším textu.
- Protože se dá vzdálený pohled přidat do datového prostředí formuláře nebo sestavy, lze pak těžit z výhod vizuálních postupů, které datové prostředí nabízí: můžete přetahovat myší jednotlivá pole nebo celý kurzor a tak automaticky vytvářet ovládací prvky, můžete svázat ovládací prvek s polem, které snadno vyberete ze seznamu v okénku Vlastnosti (Properties), a podobně. Také můžete nastavením vlastností AutoOpenTables a OpenViews zařídit, že bude VFP automaticky otevírat vzdálené pohledy za vás.
- Je snadné přenést aktualizovaná data na vzdálené úložiště: za předpokladu, že byly vlastnosti pohledu nastaveny správně, stačí zavolat TABLEUPDATE(). Zpracování transakcí a detekce konfliktů jsou vestavěné.
- Vzdálené pohledy se snadno používají i ve vývojovém prostředí: stačí k tomu příkazy USE a BROWSE.

Nevýhody

Nevýhody vzdálených pohledů jsou:

- Vzdálené pohledy jsou vázány na databázový kontejner, takže je do klientského systému nutno nainstalovat a pak udržovat další sadu souborů.
- Příkaz SQL SELECT vzdáleného pohledu je předem definován a nedá se měnit za chodu. Pro běžný formulář určený ke vkládání dat je to sice v pořádku, ale u dotazů a sestav to může představovat problém. Možná bude zapotřebí vytvořit pro tutéž sadu dat několik pohledů, které se budou lišit ve vybraných sloupcích, struktuře fráze WHERE apod.
- Ze vzdáleného pohledu nelze volat uloženou proceduru, proto vzdálený pohled potřebuje přímý přístup k zúčastněným tabulkám.
- Když k zápisu změn do vzdálené databáze používáte TABLEUPDATE(), máte jen málo možností (jiných než nastavení několika vlastností) ovlivnit, jak VFP ten update provede.
- Stejně jako u lokálních pohledů, načítáte-li z určité tabulky všechna pole příkazem SELECT * a struktura vzdálené tabulky se změnila, je pohled neplatný a musíte ho vytvořit znovu.
- Při otevírání pohledu se VFP snaží zamknout všechny záznamy o něm v databázovém kontejneru, byť jen nakrátko. To může způsobit zádrhely v hodně vytížených aplikacích, kde se ve stejném okamžiku pokusí otevřít stejný formulář více uživatelů. I když se to dá obejít (zkopírovat DBC na lokální stanici a pak pracovat s touto kopií nebo, ve VFP 7 a vyšších, příkazem SET REPROCESS SYSTEM zvýšit časový interval pro čekání na uzamknutí), je to něco, s čím musíte počítat.
- Až do verze VFP 8 byla jen malá možnost spravovat spojení, která použila vaše aplikace. Od verze VFP 8 se dá zadat manipulátor příkazu (handle) spojení, které má použít příkaz USE při otevírání vzdáleného pohledu.
- Informace o spojení použitém pro vzdálený pohled jsou uloženy v souboru DBC jako obyčejný text. To znamená, že nějaký hacker se může snadno dostat ke klíčům od vašeho vzdáleného království (jako jsou uživatelské jméno a heslo) prostým otevřením databázového kontejneru v tak jednoduché aplikaci, jakou je Poznámkový blok (Notepad). Počínaje verzí VFP 7 už to není takový problém, protože lze zadat spojovací řetězec až při otevírání vzdáleného pohledu příkazem USE, takže si můžete těsně před otevřením pohledu dynamicky poskládat cestu k databázi, uživatelské jméno a heslo z pravděpodobně dobře zašifrovaných informací.

Tím jsme se dostali k jádru věci: vzdálené pohledy usnadňují práci se vzdálenými daty, ovšem za cenu omezení kontroly, která vám nad nimi zůstane.

Přístup k ADS pomocí přímého předávání příkazů SQL (SQL passthrough, SPT)

VFP nabízí několik funkcí, někdy označovaných jako funkce přímého předávání příkazů SQL neboli SPT, které umožňují přístup ke vzdálené databázi. SQLCONNECT() a SQLSTRINGCONNECT() navážou spojení se vzdáleným databázovým strojem. Rozdíl mezi nimi spočívá v tom, že SQLCONNECT() vyžaduje existující ODBC DSN, zatímco SQLSTRINGCONNECT() používá spojovací řetězec. SQLDISCONNECT() slouží k odpojení od vzdálené databáze. SQLEXEC() pošle databázovému stroji příkaz, například SQL SELECT, a obvykle (ne však vždy, záleží to na konkrétním příkazu) uloží vrácené výsledky do kurzoru VFP.

Následující příklad se připojí k převedené databázi Northwind, načte všechny zákazníky a odpojí se. (Předpokládá, že existuje DSN „Northwind“ definující, jak se k této databázi připojit.)

```
lnHandle = sqlconnect('Northwind')
sqlexec(lnHandle, 'select * from Customers')
browse
sqldisconnect(lnHandle)
```

Chceme-li místo toho použít spojení bez DSN, nahradíme příkaz SQLCONNECT() takto:

```
lcConnString = 'driver=Advantage StreamlineSQL ODBC;' + ;
  'DataDirectory=C:\ADSTest\Northwind\Northwind.add;'
lnHandle = sqlstringconnect(lcConnString)
```

Tabulka 1 představuje klíčová slova, která může spojovací řetězec obsahovat; většina z nich má svůj ekvivalent v dialogu ODBC, který ukazuje Obrázek 1. Všechna klíčová slova až na DataDirectory jsou nepovinná.

***Tabulka** 1. Klíčová slova přípustná ve spojovacím řetězci ODBC driveru ADS.*

Klíčové slovo	Popis
DataDirectory	Cesta a název souboru ADD v případě databáze ADS, adresář s tabulkami v případě volných tabulek.
DefaultType	Pro volné tabulky VFP se zadává „FoxPro“, pro volné tabulky ADS (soubory ADT) se zadává „Advantage“. Pro databáze je toto nastavení ignorováno.
ServerTypes	Numerická hodnota, součet typů serverů ADS, k nimž se navazuje spojení: Remote (2), Local (1) nebo Internet (4). Například pro Remote a Local to bude hodnota 3 (2 + 1).
AdvantageLocking	„ON“ (výchozí) pro interní zamykání ADS, „OFF“ pro zamykání kompatibilní s VFP.
Locking	„Record“ (výchozí) pro zamykání jednotlivých záznamů během aktualizace dat, „File“ pro uzamknutí celého souboru.
Rows	Podobá se SET DELETED [hh: až na to, že s převrácenými hodnotami ☺]. „True“ zobrazuje i vyřazené záznamy, „False“ (výchozí) je vynechává. Toto nastavení je ignorováno pro tabulky ADS, protože v tomto případě nejsou vyřazené záznamy vidět nikdy.
TrimTrailingSpaces	Podobá se polím typu Varchar. „True“ odstraňuje ze znakových polí, která se vracejí aplikaci, pravostranné mezery, „False“ (výchozí) je tam ponechává.
MemoBlockSize	Podobá se příkazu SET BLOCKSIZE. Určuje velikost bloku pro pole memo v nových tabulkách. Výchozí hodnota je 64 pro tabulky VFP a 8 pro tabulky ADS.
CharSet	Která třídicí sekvence (collation) se má používat: „ANSI“ (výchozí) nebo „OEM“. Pro „OEM“ je nutno zadat i parametr Language.
Language	Jazyk pro parametr CharSet=OEM.
MaxTableCloseCache	Počet kurzorů v paměti cache ADS; výchozí hodnota je 5.
Compression	Jaký typ komprese se má používat. Podrobnější informace jsou uvedeny v souboru nápovědy k ADS.
CommType	Který komunikační protokol se má používat. Podrobnější informace jsou uvedeny v souboru nápovědy k ADS.

Bez ohledu na to, zda jste pro připojení k ADS použili DSN nebo spojovací řetězec, dále už zadáváte stejné příkazy jazyka SQL, jakými byste načítali, aktualizovali nebo vyřazovali záznamy v tabulkách VFP, ale voláte je pomocí funkce SQLEXEC(). Vedle funkcí DML (Data Manipulation Language), jako jsou SELECT, INSERT, UPDATE a DELETE, podporuje ODBC driver i funkce DDL (Data Definition Language), tedy CREATE DATABASE | TABLE | INDEX | VIEW | PROCEDURE, DROP INDEX | TABLE | VIEW | PROCEDURE a ALTER TABLE.

Je dobré si uvědomit, že i když ADS podporuje většinu datových typů VFP, způsob zadávání hodnot polí typu Logical, Date a DateTime se od syntaxe VFP poněkud liší. Hodnoty typu Logical se z ADS předávají zpět do VFP zase jako Logical, ale musí být zadány jako True nebo 1 resp. False nebo 0. Například z následujících čtyř příkazů projdou jen poslední dva:

```
sqlexec(lnHandle, "select * from Products where Discontinued")
sqlexec(lnHandle, "select * from Products where Discontinued=.T.")
sqlexec(lnHandle, "select * from Products where Discontinued=True")
sqlexec(lnHandle, "select * from Products where Discontinued=1")
```

(Podle informací ze Sybase iAnywhere je možné, že ostrá verze bude zvládat i syntaxi s „.T.“ a „.F.“.)

Hodnoty typu Date a DateTime musí dodržovat standardní syntaxi ODBC: {d 'YYYY-MM-DD'} pro Date a {ts 'YYYY-MM-DD HH:MM:SS'} pro DateTime. Příklad:

```
sqlexec(lnHandle, "select * from orders " + ;
  "where OrderDate between {d '1997-07-01'} and {d '1997-07-31'}")
```

Takto vypadá funkce VFP2ODBCDate pro převod hodnot typu Date a DateTime ze syntaxe VFP do syntaxe ODBC:

```
lparameters tuDate
local lcDate, ;
  lcReturn
lcDate = transform(year(tuDate)) + ;
  '-' + padl(month(tuDate), 2, '0') + ;
  '-' + padl(day(tuDate), 2, '0')
if vartype(tuDate) = 'D'
  lcReturn = "{d '" + lcDate + "'}"
else
  lcReturn = "{t '" + lcDate + ;
    ' ' + padl(hour(tuDate), 2, '0') + ;
    ':' + padl(minute(tuDate), 2, '0') + ;
    ':' + padl(sec(tuDate), 2, '0') + "'}"
endif vartype(tuDate) = 'D'
return lcReturn
```

Příklad s využitím této funkce:

```
ldFrom = {^1997-07-01}
ldTo   = {^1997-07-31}
sqlexec(lnHandle, "select * from orders " + ;
  "where OrderDate between " + VFP2ODBCDate(ldFrom) + " and " + VFP2ODBCDate(ldTo))
```

Místo převádění hodnot ze syntaxe VFP do syntaxe ODBC lze použít parametrický dotaz, v němž je „natvrdo“ zadaná hodnota nahrazena názvem proměnné, který má před sebou otazník (příkaz samozřejmě musí spadat do oboru platnosti dané proměnné). V tomto případě obstará VFP konverzi datových typů za vás. Má to ještě další výhodu – vyhnete se útokům tzv. SQL injection (jejich popis už překračuje možnosti tohoto článku). Příklad:

```
ldFrom = {^1997-07-01}
ldTo   = {^1997-07-31}
sqlexec(lnHandle, 'select * from orders ' + ;
  'where OrderDate between ?ldFrom and ?ldTo')
```

Ve VFP je sice možné vybrat si jako omezovače řetězců apostrofy, uvozovky nebo hranaté závorky, znakové hodnoty předávané ve funkcích SPT je však nutné uzavírat výhradně do apostrofů.

I když to ve výše uvedených příkladech chybí, rozhodně nezapomínejte zkontrolovat hodnotu, kterou vrací SQLEXEC(). Vrátí-li číslo menší než 1, příkaz zhavaroval a musíte pomocí funkce AERROR() zjistit příčinu. Jako třetí parametr funkce SQLEXEC() lze uvést název kurzoru, který má vzniknout; pokud tento parametr vynecháte, bude se kurzor jmenovat SQLResult.

Výhody

Výhody přímého předávání příkazů SQL jsou:

- Oproti vzdáleným pohledům je tento způsob přístupu k datům mnohem flexibilnější, pomocí funkce SQLEXEC() se třeba dají volat uložené procedury.
- Parametry spojení lze nastavovat za běhu podle potřeby. Například uživatelské jméno a heslo může být někde zašifrované a stačí je odšifrovat těsně před použitím ve funkci SQLCONNECT() nebo SQLSTRINGCONNECT(). Jak je uvedeno výše, význam této výhody už poklesl, protože od verze VFP 7 je možné zadávat spojovací řetězec rovnou v příkazu USE.
- Příkaz SQL SELECT si můžete snadno upravovat podle potřeby, například obměňovat seznam sloupců, podmínky ve frázi WHERE (některé třeba úplně vynechat), tabulky apod.
- K přímému předávání příkazů SQL není zapotřebí databázový kontejner DBC, není tedy co udržovat nebo instalovat, odpadá potenciální problém se zamknutými záznamy a nehrozí, že by po změně struktury vzdálených tabulek přestal být příkaz SELECT * platný.
- Podobně jako u vzdálených pohledů lze výsledek volání SPT používat ve VFP téměř libovolně, protože je to foxovský kurzor.
- Musíte si to sice naprogramovat sami (podrobněji o tom pojednává následující odstavec Nevýhody), máte však větší kontrolu nad tím, jak se provádějí aktualizace. Můžete například naplnit kurzor příkazem SQL SELECT, ale k aktualizaci obsahu tabulek ADS volat uloženou proceduru.
- Svá vlastní spojení si můžete sami spravovat. Můžete třeba vytvořit objekt, který bude spravovat všechna spojení, která používá vaše aplikace, pěkně z jednoho místa.

Nevýhody

Nevýhody přímého předávání příkazů SQL jsou:

- Je to pracnější, protože si musíte sami naprogramovat úplně všechno: vytváření a zavírání spojení, příkazy SQL SELECT, které se mají vykonat, atd. Nemáte k dispozici žádný pěkný vizuální nástroj jako Návrhář pohledů (View Designer), aby vám ukázal, která pole jsou v kterých tabulkách.
- Kurzor vytvořený pomocí SPT se nedá vizuálně přidat do datového prostředí formuláře nebo sestavy. Místo toho si musíte sami naprogramovat otevírání kurzorů (například v metodě BeforeOpenTables), ručně vytvořit ovládací prvky a nastavit jim vlastnosti, které je svážou s daty (třeba ControlSource). A když se při zadávání názvu alias nebo některého pole uklepnete, nebude formulář fungovat.
- Ve vývojovém prostředí se s nimi pracuje hůř než se vzdálenými pohledy: místo prostého příkazu USE musíte napřed založit spojení a pak pomocí SQLEXEC() načíst data, která si chcete prohlédnout. Můžete si trochu ulehčit život a napsat si sadu programů, které to pro vás udělají, nebo k prozkoumání struktur a obsahu tabulek využít Data Explorer, který se dodává s VFP. Případně můžete vytvořit soubor DBC a sadu vzdálených pohledů čistě pro účely nakouknutí do dat jen z vývojového prostředí.
- Kurzory vytvořené pomocí SPT sice nemusí sloužit jen ke čtení dat, ale musíte to zařídit sami sérií volání funkce CURSORSETPROP(), v nichž nastavíte vlastnosti SendUpdates, Tables, KeyFieldList, UpdatableFieldList a UpdateNameList. Musíte také sami ošetřit zpracování transakcí a vyřešení případných konfliktů.
- Protože kurzory vytvořené pomocí SPT nejsou definovány jako vzdálené pohledy, nelze při přímém předávání příkazů SQL jednoduše přepínat mezi lokálními a vzdálenými daty, jako když u vzdálených pohledů prostě jen zadáte, který pohled se má pro formulář nebo sestavu otevřít.

Přístup k ADS pomocí ADO

OLE DB provider se podobá ODBC driveru: obstarává standardní, konzistentní způsob přístupu k datovým zdrojům. OLE DB je sada nízkoúrovňových rozhraní COM, takže není snadné s ním pracovat v jazycích

jako VFP. Proto Microsoft vytvořil ActiveX Data Objects (ADO), sadu objektů COM zajišťujících objektově orientované rozhraní (front-end) pro OLE DB.

ADO se skládá z několika objektů. Patří sem:

- Connection: tento objekt je zodpovědný za komunikaci s datovým zdrojem.
- Recordset: ekvivalent foxovského kurzoru. Má nadefinovanou strukturu, obsahuje data a nabízí vlastnosti a metody pro přidávání, odstraňování nebo aktualizaci záznamů, pohyb po záznamech, filtrování nebo řazení dat a aktualizaci datového zdroje.
- Command: tento objekt dovoluje provádět i náročnější příkazy než jednoduchý SELECT, např. parametrické dotazy a volání uložených procedur.

Následující příklad (ADOExample.PRG) načte z převedené databáze Northwind všechny brazilské zákazníky a zobrazí ID zákazníka a název firmy. Je zde vidět, že objekt Connection se stará o spojení a objekt Recordset má na starosti data. Program obsahuje odkaz na ADOVFP.H, hlavičkový soubor s konstantami užitečnými pro práci s ADO.

```
#include ADOVFP.H
local loConn as ADODB.Connection, ;
  loRS as ADODB.Recordset, ;
  lcCustomers

* Connect to the ADS database.

loConn = createobject('ADODB.Connection')
loConn.ConnectionString = 'provider=Advantage.OLEDB.1;' + ;
  'data source=c:\adstest\northwind\northwind.add'
loConn.Open()

* Create a Recordset and set its properties.

loRS = createobject('ADODB.Recordset')
loRS.ActiveConnection = loConn
loRS.LockType         = 3  && adLockOptimistic
loRS.CursorLocation   = 3  && adUseClient
loRS.CursorType       = 3  && adOpenStatic

* Execute a query and display the results.

loRS.Open("select * from customers where country='Brazil'")
lcCustomers = ''
do while not loRS.EOF
  lcCustomers = lcCustomers + ;
    loRS.Fields('customerid').Value + chr(9) + ;
    loRS.Fields('companyname').Value + chr(13)
  loRS.MoveNext()
enddo while not loRS.EOF
messagebox(lcCustomers)
loRS.Close()
loConn.Close()
```

Všimněte si, jak tento příklad přistupuje k sadě záznamů pomocí objektově orientovaného kódu. Vlastnost EOF je ekvivalentní foxovské funkci EOF() a metoda MoveNext funguje jako SKIP. Hodnotu určitého pole v aktuálním záznamu vrací Recordset.Fields('FieldName').Value.

Parametrické dotazy dají v ADO trochu víc práce než v ODBC. Parametr se v dotazu zadává pouhým otazníkem (bez názvu proměnné), ale zato musíte jeho název a hodnotu předem zadat pomocí objektů Command a Parameter.

```
* Connect to the ADS database.

loConn = createobject('ADODB.Connection')
loConn.ConnectionString = 'provider=Advantage.OLEDB.1;' + ;
  'data source=c:\adstest\northwind\northwind.add'
loConn.Open()
```

```
* Create a Command object and define the command type and connection.

loCommand = createobject('ADODB.Command')
loCommand.CommandType      = adCmdText
loCommand.ActiveConnection = loConn

* Create a Parameter object, set its properties, and add it to the Command
* object.

loParameter = loCommand.CreateParameter('Country', adChar, adParamInput, 15)
loParameter.Value = 'UK'
loCommand.Parameters.Append(loParameter)

* Execute a parameterized query and display the results.

loCommand.CommandText = 'select * from customers where country = ?'
loRS = loCommand.Execute()
* same code as above to display the results
```

Výhody

Výhody ADO jsou:

- Mnohé z výhod jsou stejné jako u přímého předávání příkazů SQL: přístup k datům je flexibilnější než u vzdálených pohledů, parametry spojení lze nastavovat za běhu podle potřeby, příkaz SQL SELECT se dá upravovat podle potřeby, můžete si sami spravovat svá vlastní spojení, obejdete se bez databázového kontejneru.
- Ačkoliv rozdíly ve výkonu nejsou u jednoduchých scénářů nijak podstatné (ve skutečnosti je obecně ODBC rychlejší než ADO), přizpůsobí se ADO lépe vysokému zatížení u aplikací, jako jsou webové servery.
- ADO je objektově orientované, takže lze s daty zacházet jako s objekty.
- Při správném nastavení umožňují sady záznamů ADO aktualizaci automaticky, bez nějaké práce navíc, stačí zavolat metodu Update nebo UpdateBatch. Zpracování transakcí a ošetření případných konfliktů jsou vestavěné.
- Recordset lze snadno uložit do lokálního souboru, později ho znovu načíst a pokračovat v práci, a nakonec aktualizovat datový zdroj ADS. U aplikací určených „obchodním cestujícím“ je to mnohem lepší volba než vzdálené pohledy nebo SPT.

Nevýhody

Nevýhody ADO jsou:

- Je to pracnější, protože si musíte sami naprogramovat úplně všechno: vytváření a zavírání spojení, příkazy SQL SELECT, které se mají vykonat, atd. Nemáte k dispozici žádný pěkný vizuální nástroj jako Návrhář pohledů (View Designer), aby vám ukázal, která pole jsou v kterých tabulkách.
- Sada záznamů ADO není foxovský kurzor, takže se nedá použít jako zdroj dat tam, kde je kurzor nutný, např. pro grid nebo sestavu. Utilita VFPCOM (která se dá stáhnout z domovské stránky VFP, http://msdn.microsoft.com/vfoxpro) obsahuje funkce pro konverzi sady záznamů do kurzoru a naopak, ale jednak mohou snížit výkon, zvláště u velkého objemu dat, jednak je o nich známo, že mají potíže s určitými datovými typy. Chcete-li používat ADO, je lepší volbou CursorAdapter (popsaný v dalším odstavci).
- Sady záznamů ADO se nedají zpracovávat vizuálně. Místo toho si musíte sami naprogramovat jejich vytváření a otevírání, ručně vytvořit ovládací prvky a nastavit jim vlastnosti, které je svážou s daty (třeba ControlSource). Je to ještě víc práce než u SPT, protože tu není syntaxe CURSOR.FIELD, ale Recordset.Fields('FieldName').Value.
- Je to nejobtížnější technologie pro použití ve vývojovém prostředí, protože si musíte sami naprogramovat úplně všechno: navázání spojení, načtení dat, pohyb mezi záznamy. Nemůžete ani

zavolat BROWSE a podívat se, jak vypadá výsledek (ledaže použijete VFPCOM nebo CursorAdapter a překonvertujete Recordset do kurzoru).

- Naučit se pracovat s ADO je náročnější než používat kurzory vytvořené pomocí ODBC.

Přístup k ADS pomocí CursorAdapteru

Nejspíš jste si všimli, že každý z popsaných způsobů přístupu se naprosto liší od těch ostatních. To znamená, že u každého z nich se toho budete muset hodně učit a že převést existující aplikaci z jednoho typu přístupu na jiný rozhodně není triviální záležitost.

Naštěstí existuje ve VFP technologie, která nabízí společné rozhraní pro ODBC i OLE DB: třída CursorAdapter. Tato novinka verze VFP 8 je opravdu úžasná:

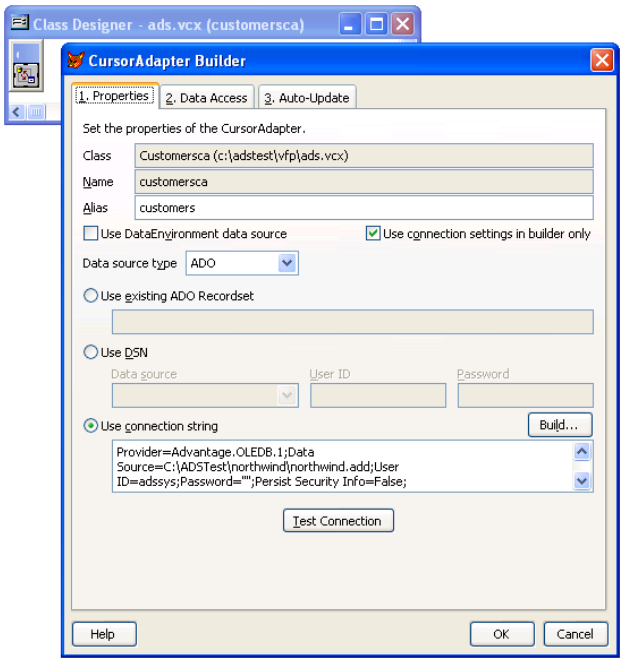
- Umožňuje snadno používat technologie ODBC, ADO nebo XML, i když se v nich zatím moc nevyznáte.
- Poskytuje konzistentní rozhraní pro přístup ke vzdáleným datům bez ohledu na to, který mechanismus si vyberete.
- Umožňuje snadné přepínání mezi jednotlivými mechanismy.

Poslední bod si ukážeme na příkladu. Dejme tomu, že vaše aplikace přistupuje k datům ADS pomocí ODBC v CursorAdapterech a z nějakého důvodu to potřebujete změnit na ADO. Úplně stačí změnit DataSourceType v CursorAdapterech a změnit připojení k databázi ADS, a máte hotovo. Ostatní součásti aplikace se o to vůbec nemusí starat; vidí pořád stejný kurzor, ať už je mechanismus přístupu k datům jakýkoliv.

Následující příklad (CursorAdapterExample.PRG) načte z tabulky Customers v databázi Northwind určitá pole pro všechny brazilské zákazníky. Kurzor dovoluje aktualizace, takže provedete-li v okně Browse nějaké změny, zavřete ho a spustíte program znovu, uvidíte, že změny byly uloženy.

```
local lcConnString, ;
  lnHandle, ;
  loCursor as CursorAdapter, ;
  laErrors[1]
close tables all

* Connect to ADS.

lcConnString = 'driver=Advantage StreamlineSQL ODBC;' + ;
  'DataDirectory=C:\ADSTest\Northwind\Northwind.add;'
lnHandle     = sqlstringconnect(lcConnString)

* Create a CursorAdapter and set its properties.

loCursor = createobject('CursorAdapter')
with loCursor
  .Alias              = 'Customers'
  .DataSourceType     = 'ODBC'
  .DataSource         = lnHandle
  .SelectCmd          = "select CUSTOMERID, COMPANYNAME, CONTACTNAME " + ;
    "from CUSTOMERS where COUNTRY = 'Brazil'"
  .KeyFieldList       = 'CUSTOMERID'
  .Tables             = 'CUSTOMERS'
  .UpdatableFieldList = 'CUSTOMERID, COMPANYNAME, CONTACTNAME'
  .UpdateNameList     = 'CUSTOMERID CUSTOMERS.CUSTOMERID, ' + ;
    'COMPANYNAME CUSTOMERS.COMPANYNAME, CONTACTNAME CUSTOMERS.CONTACTNAME'
  if .CursorFill()
    browse
  else
    aerror(laErrors)
    messagebox(laErrors[2])
  endif .CursorFill()
endwith
```

Není nutné vytvářet CursorAdapter programem. V návrháři tříd (Class Designer) lze založit podtřídu třídy CursorAdapter a buď nastavit její vlastnosti v okně Properties nebo k tomu využít pěkný vizuální CursorAdapter Builder (viz **Obrázek 6**). Pozor, CursorAdapter Builder nespolupracuje s ADS úplně správně, pokud se připojíte k databázi prostřednictvím ODBC; v odstavci „Jak opravit CursorAdapter Builder VFP“ najdete podrobnosti a návod, jak tento problém opravit.



Obrázek 6. CursorAdapter Builder nabízí vizuální nástroj k vytváření podtříd třídy CursorAdapter.

CursorAdapter má celou řadu vlastností a metod, nejdůležitější z nich jsou:

- DataSourceType: určuje způsob přístupu k datům. Možnosti jsou „Native“, „XML“, „ODBC“ a „ADO“, pro přístup k ADS lze použít jen poslední dvě.
- DataSource: hodnota této vlastnosti závisí na nastavení DataSourceType. V případě ODBC to musí být handle otevřeného spojení ODBC. Pro ADO je to objekt Recordset s vlastností ActiveConnection nastavenou na objekt Connection (i zde musí být spojení otevřené). Mějte na paměti, že v obou případech za otevírání a správu spojení zodpovídáte vy sami.
- SelectCmd: příkaz SQL, jehož vykonáním se načtou data.
- Alias: alias vytvořeného kurzoru.
- KeyFieldList, Tables, UpdatableFieldList a UpdateNameList: tato pole jsou klíčová pro přepisovatelnost kurzoru. Jsou-li jejich hodnoty nastaveny správně, bude CursorAdapter automaticky zapisovat změny do vzdáleného zdroje dat. Podrobné informace o těchto vlastnostech najdete v dokumentaci k VFP.
- CursorFill: zavoláním této metody vznikne kurzor a vykoná se příkaz uvedený v SelectCmd, který tento kurzor naplní daty.
- CursorDetach: implicitně je kurzor vytvořený CursorAdapterem „připoután“ k objektu CursorAdapter. Při zániku CursorAdapteru (například když skončí obor jeho platnosti) se kurzor automaticky zavře. Chcete-li, aby kurzor zůstal otevřený, zavolejte metodu CursorDetach, která ho od CursorAdapteru „odpoutá“.

CursorAdapter může pro připojení k ADS použít buď ODBC, nebo ADO. V případě ODBC stačí otevřít spojení s databází příkazem SQLCONNECT() nebo SQLSTRINGCONNECT(), nastavit hodnotu

vlastnosti DataSource CursorAdapteru na handle tohoto spojení a hodnotu vlastnosti DataSourceType na „ODBC“. ADO dá trochu víc práce: vytvořit instanci objektu Connection a otevřít spojení, vytvořit instanci objektu Recordset, nastavit hodnotu jeho vlastnosti ActiveConnection na objekt Connection, nastavit hodnotu vlastnosti DataSource CursorAdapteru na objekt Recordset a hodnotu vlastnosti DataSourceType na „ADO“. U parametrických dotazů se používá v příkazu SQL SELECT syntaxe „?VariableName“, a to i pro ADO. Nicméně v případě ADO musíte také vytvořit instanci objektu Command a předat ji jako čtvrtý parametr metodě CursorFill (o objekt Parameter se nemusíte starat; VFP ho ošetří interně).

Abyste všechnu tu práci kolem ADO nemuseli dělat ručně, udělá jí kus za vás podtřída SFCursorAdapterADO z knihovny SFCursorAdapter.VCX, která je součástí příloh k tomuto článku. Její vlastnost DataSourceType má hodnotu „ADO“ a hodnotu vlastnosti DataSource nastavuje metoda Init.

```
local loRS as ADODB.Recordset
loRS = createobject('ADODB.RecordSet')
loRS.CursorLocation = 3  && adUseClient
loRS.LockType       = 3  && adLockOptimistic
This.DataSource     = loRS
```

Po vytvoření a otevření objektu Connection ho můžete předat metodě SetConnection.

```
lparameters toConnection
This.DataSource.ActiveConnection = toConnection
```

Nemusíte předávat metodě CursorFill objekt Command; SFCursorAdapterADO ho použije automaticky, pokud příkaz SQL SELECT obsahuje otazník.

```
lparameters tlUseCursorSchema, ;
  tlNoData, ;
  tnOptions, ;
  toSource
local loSource as ADODB.Command, ;
  lnOptions, ;
  llUseCursorSchema, ;
  llNoData, ;
  llReturn, ;
  laError[1]

* If we have a parameterized query, we need an ADO Command object. Create one
* if it wasn't passed.

if '?' $ This.SelectCmd and vartype(toSource) <> 'O'
  loSource = createobject('ADODB.Command')
  loSource.ActiveConnection = This.DataSource.ActiveConnection
  lnOptions = adCmdText
else
  loSource  = toSource
  lnOptions = tnOptions
endif '?' $ This.SelectCmd ...

* If the first two parameters weren't specified, we don't want to explicitly
* pass .F., so use the default values. If CursorSchema is empty, we'll, of
* course, pass .F. for the first parameter.

do case
  case pcount() >= 2
    llUseCursorSchema = tlUseCursorSchema
    llNoData          = tlNoData
  case pcount() = 1
    llUseCursorSchema = tlUseCursorSchema
    llNoData          = This.NoData
  case pcount() = 0
    llUseCursorSchema = This.UseCursorSchema
    llNoData          = This.NoData
endcase
if empty(This.CursorSchema)
  llUseCursorSchema = .F.
endif empty(This.CursorSchema)
```

```
llReturn = dodefault(llUseCursorSchema, llNoData, lnOptions, loSource) and ;
  used(This.Alias)

* If something went wrong, find out why.

if not llReturn
  aerror(laError)
  This.cErrorMessage = laError[2]
endif not llReturn
return llReturn
```

Následující příklad ADOCursorAdapterExample.PRG využívá SFCursorAdapterADO:

```
local loConnection as ADODB.Connection, ;
  loCA as SFCursorAdapterADO of SFCursorAdapter.vcx
private pcCountry

* Create and open an ADO Connection object.

loConnection = createobject('ADODB.Connection')
loConnection.ConnectionString = 'Provider=Advantage.OLEDB.1;' + ;
  'Data Source=C:\ADSTest\northwind\northwind.add;' + ;
  'User ID=adssys;Password=""'
loConnection.Open()

* Create an SFCursorAdapterADO object and set it up.

loCA = newobject('SFCursorAdapterADO', 'SFCursorAdapter.vcx')
with loCA
  .SetConnection(loConnection)
  .SelectCmd = 'select * from customers where country = ?pcCountry'
  .Alias     = 'customers'

* Do the query and either show the result set or an error.

  pcCountry = 'Germany'
  if .CursorFill()
    browse
  else
    messagebox(loCa.cErrorMessage)
  endif .CursorFill()
endwith

* Close the connection.

loConnection.Close()
```

Výhody

CursorAdaptery v sobě v podstatě kombinují výhody ostatních technologií.

- V závislosti na nastaveních daného objektu (například je-li zcela samostatný [soběstačný?]) může být otevření kurzoru z podtřídy CursorAdapter skoro stejně jednoduché jako otevření vzdáleného pohledu: stačí vytvořit instanci podtřídy a zavolat metodu CursorFill. Můžete to dokonce udělat celé v jednom kroku, pokud toto volání umístíte do metody Init.
- Převod existující aplikace je jednodušší, bude-li používat CursorAdaptery a ne kurzory vytvořené přímým předáváním příkazů SQL.
- Stejně jako vzdálený pohled, i CursorAdapter se dá přidat do datového prostředí formuláře nebo sestavy a pak těžit z výhod vizuálních postupů, které datové prostředí nabízí: můžete přetahovat myší jednotlivá pole a tak automaticky vytvářet ovládací prvky, můžete snadno svázat ovládací prvek s polem, které vyberete ze seznamu v okénku Vlastnosti (Properties), a podobně.
- Je snadné přenést aktualizovaná data na vzdálené úložiště: za předpokladu, že byly vlastnosti pohledu [??] nastaveny správně, stačí zavolat TABLEUPDATE().

- Protože výsledná sada záznamů vytvořená CursorAdapterem je foxovský kurzor, dá se ve VFP použít kdekoliv: v gridu, v sestavě, procházet v cyklu SCAN atd. To platí i v případě, že zdrojem dat je ADO nebo XML, neboť CursorAdapter se automaticky postará o konverzi do a z kurzoru za vás.
- Přístup k datům je velmi flexibilní, lze využívat uložené procedury nebo objekty střední vrstvy.
- Parametry spojení můžete nastavovat za běhu podle potřeby, příkaz SQL SELECT se dá upravovat podle potřeby, obejdete se bez databázového kontejneru a můžete si sami spravovat svá vlastní spojení.
- Musíte si to sice naprogramovat sami, máte však větší kontrolu nad tím, jak se provádějí aktualizace. Můžete například naplnit kurzor příkazem SQL SELECT, ale k aktualizaci obsahu tabulek ADS volat uloženou proceduru.

Nevýhody

CursorAdaptery mají jen málo nevýhod:

- Pro vytváření CursorAdapterů není k dispozici takový hezký vizuální nástroj jako návrhář pohledů (View Designer), ale CursorAdapter Builder ho zdatně dohání.
- Jako u všech nových technologií, i zde se toho musíte spoustu naučit.

Licencování

Advantage Database Server vyžaduje licenci pro každého připojeného uživatele. Každá pracovní stanice může mít neomezený počet databázových připojení. Pro několik aplikací přistupujících k ADS z téže pracovní stanice stačí jedna licence. Sybase iAnywhere nezveřejňuje ceník licencí. Uvádí sice, že existují různá cenová zvýhodnění a že jsou ceny ADS srovnatelné s jinými databázovými servery, je však nutné vyžádat si konkrétní nabídku pro požadovaný počet licencí.

Zdroje

Na webu Advantage Developer Zone, http://devzone.advantagedatabase.com, jsou četné zdroje, z nichž se můžete o ADS dozvědět více: diskusní fóra (včetně jednoho zaměřeného přímo na VFP), online dokumentace, podrobné články, příručky a ukázky kódu. Skvěle vám ADS představí i kniha *Advantage Database Server: A Developer's Guide* (ISBN 978-1-4259-7726-9), kterou napsali Cary Jensen a Loy Anderson. Žádný z příkladů sice není ve VFP, ale foxaři nebudou mít problémy s tím, aby ukázky pochopili a přeložili si je.

Na adrese http://akselsoft.libsyn.com/index.php?post_id=302994 je ke stažení podcast The FoxShow #49, v němž Andrew MacNeill hovoří s J.D. Mullinem, R & D Managerem pro ADS. Toto interview přináší základní informace o vztahu ADS a VFP a zmiňuje se o designérských funkcích ADS.
www.Sybase.com/foxpro

Shrnutí

Advantage Database Server je vynikající databázový stroj, který nabízí programátorům ve Visual FoxPro lepší podporu než kterýkoliv jiný databázový stroj typu klient/server. Může se stát základnou pro převod aplikací ze souborově orientovaného přístupu k datům na opravdovou technologii klient/server.

V době vzniku tohoto článku (duben 2008) byla na světě jen beta verze ADS 9, je tedy možné, že v konečné verzi ještě došlo k nějakým změnám oproti tomu, co je zde popsáno. Základní pojetí však zůstává stejné, zejména pokud jde o přínosy ADS a způsoby, jak k němu mohou aplikace VFP přistupovat.

Jak opravit utilitu ADS pro převod databáze VFP

V ostré verzi už to asi bude vyřešeno, ale beta verze ADS má několik zádrhelů v utilitě ADSUpsize.PRG:

- Neošetřuje správně autoinkrementální pole.
- U některých tabulek dochází k chybě v metodě AddDatabaseToTable.
- Nelze ji spustit víc než jednou bez toho, že byste po ní ručně uklidili.
- Neošetřuje správně situaci, kdy se nezdařilo připojení k OLE DB provideru ADS.

- Dává relacím mezi tabulkami nic neříkající názvy jako Relation_1 a Relation_2.
- Při převádění pohledů neošetřuje rozdíly mezi syntaxí VFP a syntaxí ADS, zejména co se týká filtrovacích podmínek pro pole typu Logical, Date nebo DateTime a odkazů na databázový kontejner (tj. DatabaseName!TableName).
- U tabulek bez indexů dochází k chybě.
- Nepřevádí správně tabulky s poli typu Varchar nebo Varbinary.

Naštěstí se to všechno dalo snadno opravit a součástí příloh k tomuto článku je náhradní ADSUpsize.PRG, který se o to postará.

Jak opravit CursorAdapter Builder VFP

CursorAdapter Builder má problém s ODBC driverem ADS (s OLE DB providerem ADS funguje správně). Tlačítkem Build u příkazu Select na kartě 2 CursorAdapter Builderu se vyvolá dialog Select Command Builder. V něm dochází s ODBC driverem ADS k chybě, protože driver vrací funkcím SQLTABLES() a SQLCOLUMNS() jinou výslednou sadu než SQL Server a řada dalších driverů. Pole se nejmenují TABLE_NAME a COLUMN_NAME, nýbrž TABLENAME a COLUMNNAME.
Naštěstí Microsoft poskytuje zdrojový kód CursorAdapter Builderu a oprava byla jednoduchá; součástí příloh k tomuto článku je náhradní DEBuilder.APP, který se o to postará. Zkopírujte tento soubor do podadresáře Wizards domovského adresáře VFP, původní soubor jím přepište. Zdrojový kód opravy je zahrnut v souboru DECABuilder.VCX, který obvykle najdete v podadresáři Tools\XSource\VFPSource\Wizards\DEBuilder domovského adresáře VFP poté, co rozbalíte Tools\XSource\XSource.ZIP.

O autorovi

Doug Hennig je společníkem firem Stonefield Systems Group Inc. a Stonefield Software Inc. Vytvořil vynikající nástroje Stonefield Database Toolkit (SDT) a Stonefield Query a některé součásti Microsoft Visual FoxPro (aplikace MemberData Editor a Anchor Editor, dialog pro novou vlastnost nebo metodu a tvůrce CursorAdapteru a datového prostředí) a projektu Sedna (My namespace a aktualizovaný Upsizing Wizard). Doug je spoluautorem řady *What's New in Visual FoxPro* a knihy *The Hacker's Guide to Visual FoxPro 7.0*. Byl technickým redaktorem knih *The Hacker's Guide to Visual FoxPro 6.0* a *The Fundamentals*. Všechny tyto knihy pocházejí z vydavatelství Hentzenwerke Publishing (www.hentzenwerke.com). Doug napsal během 10 let přes 100 článků pro měsíčník FoxTalk a také řadu článků pro časopisy FoxPro Advisor a Advisor Guide. Přednášel na každé Microsoft FoxPro Developers Conference (DevCon) od roku 1997 a na různých konferencích a uživatelských fórech po celém světě. (V roce 2006 jsme se s ním setkali i u nás.) Je jedním ze správců webové stránky VFPX, nabízející rozšíření VFP vytvořená ve foxařské komunitě, (www.codeplex.com/VFPX) a jedním z organizátorů konference Southwest Fox (www.swfox.net). Má titul Microsoft Most Valuable Professional (MVP). V roce 2006 získal ocenění FoxPro Community Lifetime Achievement Award (fox.wikis.com/wc.dll?Wiki~FoxProCommunityLifetimeAchievementAward~VFP). Web: www.stonefield.com a www.stonefieldquery.com, E-mail: dhennig@stonefield.com, Blog: doughennig.blogspot.com