SYBASE ADAPTIVE SERVER
JAVA V DATABÁZI

Vladimír Kyjonka

Sybase ČR, Kyjonka@sybase.com

Rozšíření možností databázových serverů o schopnost zpracovávat objekty a příkazy jazyka Java spojuje dva dosud oddělené světy: svět vytváření aplikačních programů a návrhu a tvorby databázové logiky. Představuje jednoduchý a vtipný, ale přitom velmi účinný krok směrem k vlastnostem objektivně relačních systémů bez nutnosti vytvářet a pracně ověřovat nové standardy. Popsané řešení umožňuje využít této extense pro definici universálních datových typů, obecné objektové logiky přímo v databázi i standardního rozhraní JDBC. V článku jsou popsány vlastnosti řešení, způsob instalace a využívání objektů Javy v databázi.

1. Úvod: Nové cesty pro vývoj IT

V rámci své Adaptivní komponentové architektury vyvíjí Sybase rovněž pro svůj relační databázový server novou architekturu na orientovanou na Javu™. Ta je základem pro rozšíření funkčních možností programování na straně serveru; otevřenost a pružnost při vývoji aplikací a jejich využívání tak dostává nový rozměr. Sybase Adaptive Server tím nabývá vlastností otevřeného objektově relačního DBMS.

V současné době je pro vývoj aplikací v IT charakteristická vysoká roztříštěnost. Na jedné straně se pro aplikační logiku vně serveru používá různých prostředí 3GL jako C/C++, Java atd., nebo 4GL jako PowerBuilder nebo Visual Basic. Na druhé straně je logika uvnitř serveru tradičně založena na procedurách na bázi SQL. V tomto smyslu dnes představují obě prostředí dva rozdílné světy, jež se jen těžce domlouvají.

Javovská iniciativa Sybase se chápe možností, jež Java slibuje - napsat aplikační kód jednou, a spustit jej kdekoliv - a rozšiřuje je na novou platformu - databázový server (DBMS). Pro vývojáře to znamená odstranění bariéry mezi programováním klientské a serverové části aplikací, což umožňuje zvýšení produktivity pro oba typy programátorů ("ryze aplikační" - využívající Javy, a "serverové" - orientované na SQL).

1.1 Hlavní rysy architektury

Javovská iniciativa Sybase otevírá nové možnosti pro vývoj aplikací:

- *Plnohodnotný programovací jazyk pro DBMS*
  Aplikační logika (ve formě javovských tříd) běží přímo na Adaptive Serveru, v němž je vestavěna Java Virtual Machine a interní rozhraní JDBC™. Tímto způsobem poskytuje Sybase plnohodnotné, ale přitom bezpečné, programovací prostředí přímo na serveru a překonává tak omezení SQL procedur.
- *Objektové datové typy*
  Objekty Javy mohou být uloženy přímo v relační databázi. To poskytuje možnost využívat bohatou škálu datových typů, což j e dnes snahou všech databázových systémů, ale s pomocí Javyje toho však dosaženo otevřenou neproprietární formou.
- *Konsistentní programovací model*

Aplikační komponenty mohou být přesunovány mezi jednotlivými vrstvami systému klienti, aplikační servery, databázové servery). Pro všechny vrstvy je k disposici jednotný konsistentní programovací model.

- *Nativní implementace*
  Všechny typy komponent jsou na serveru implementovány nativním způsobem - javovské objekty, databázová schémata, SQL se chovají tak, jak se od nich čeká a přitom umožňují vzájemnou interakci.

1.2 Otevřenost a návaznost na uznávané standardy

Architektura Javy pro relační databázi odstraňuje bariéry, bránící zvyšování produktivity při vývoji aplikací. Případné proprietární implementace nových technologií však mohou v budoucnosti vytvářet další bariéry.

Aby si Sybase zachovala otevřený charakter svých řešení i do budoucna, spolupracuje na vývoji standardů přímo s výborem pro ANSI SQL, JSQL konsorciem a JavaSoftem.

2. Shrnutí výhod - Proč Java ?

Pro svou schopnost významně zvýšit produktivitu při vývoji aplikací představuje Java programovací jazyk nové generace. V koncepci Sybase hraje klíčovou roli jako prostředek ke zvýšení produktivity při serverovém programování a rozšíření možností samotných databázových serverů.

Vhodnost Javy pro programování serverové logiky je dána následujícími vlastnostmi:

- *Silná aplikační funkcionalita*
  Pro vývoj serverové logiky je možno použít téhož programovacího jazyka jako pro vývoj aplikací. To umožňuje využívat i v serverovém prostředí celou šíři funkcí typicky "aplikačního" charakteru.
- *Zabudované bezpečnostní mechanismy*
  Pro programy, běžící uvnitř tak kritického systému, jako je databázový server, musí mít samo prostředí zabudovány účinné ochranné mechanismy. Java tuto ochranu poskytuje.
- *Objektová orientace*
  Java je od základu navržena jako objektově orientovaný jazyk. To je základem pro vytváření softwarových komponent a využívání celé škály objektových datových typů.

Použití Javy v databázi má dva hlavní aspekty:

- Jazyk pro programování serverové logiky, jako nástupce uložených procedur na bázi SQL.
- Prostředek pro vytváření objektových datových typů.

3. Java pro databázovou logiku

Dnešní databázové servery zpravidla vykonávají činnosti, jež je možno rozdělit do dvou skupin:

- Zpřístupňování dat.
- Serverová logika.

Zatímco SQL je stále považován za výborný jazyk pro manipulaci s daty, jeho rozšíření, umožňující vytváření procedurální serverové logiky, vykazuje některé zjevné nedostatky.

Možnosti uložených procedur na bázi SQL jsou omezeny zejména tím, že pro ně nejsou k disposici odpovídající vývojové nástroje, není možno je používat mimo databázi a hlavně skutečností, že postrádají řadu vlastností běžných pro moderní programovací nástroje jako je např. využívání externích knihoven, možnost zapouzdření a další objektové rysy, možnost vytvářet obecné komponenty atd.

3.1 Instalace tříd na server

Algoritmy Javy j sou vytvářeny ve formě tříd. Aby bylo možno používat Javu v databázích, umožňuje Adaptive server instalovat třídy přímo na server. Třída jsou kompilována do byte-kódu mimo server. Poté, co je nainstalována do serveru, může v něm být spouštěna i debugována.

3.2 Používání SQL v Javě

Pro implementaci javovské logiky do databáze je nezbytné rozhraní mezi Javou a SQL. Příkazy jazyka SQL pro manipulaci s daty musí být z javovských metod dostupné stejně jako z uložených procedur.

Pro klientské aplikace je disposici JDBC, programovatelné aplikační rozhraní (API), jež umožňuje začlenit SQL přímo do metod Javy. Bylo uvedeno v Java SDK verze 1.1.

Aby bylo možno odstranit bariéry mezi serverovou a klientskou aplikační logikou, je třeba, aby JDBC rovněž poskytovalo možnost používání SQL v javovských metodách v databázi. Interní JDBC interface Adaptive serveru je proto klíčovou součástí javovské iniciativy Sybase.

3.3 Podpora rychlého a snadného vývoje s JDBC

JDBC je podobně jako ODBC databázové rozhraní nízké úrovně. Pro efektivnější vývoj aplikací nad ODBC je dnes mnoho vývojových nástrojů vybaveno mnohem komfortněji, a potřeba podobného prostředí existuje i pro JDBC.

V architektuře Sybase jsou kompilované javovské třídy (v byte-kódu) instalovány přímo do databázového serveru. Pro jejich vytváření je možno použít jakýchkoliv vyšších programovacích nástrojů, jež generují kód Javy a JDBC.

Například:

- *JSQL* je alternativní metoda, umožňující začlenit volání SQL do kódu Javy. Je vyvinuta a spravována konsorciem firem IBM, Oracle, Sybase a Tandem. Poskytuje funkcionalitu obdobnou Embedded SQL v prostředí JDBC.
  Kód JSQL je výrazně jednodušší než JDBC a pro databázové administrátory je to nástroj, jehož použití je mnohem bližší způsobu vytváření uložených procedur na bázi SQL. Kód JSQL je před kompilací předzpracován do formy volání JDBC. Uživatelé Adaptive Serveru mohou přímo používat JSQL a výsledný kód instalovat na serveru.
- *RAD nástroje* pro Javu, jako např. PowerJ, produkují javovské třídy vybudované nad JDBC v uživatelsky mnohem přívětivějším prostředí pro vývojáře. Tyto třídy mohou být rovněž instalovány přímo na databázový server.

- *JavaBeans* - jsou komponenty, obsahující kolekce javovských tříd s přesně deformovaným rozhraním. JavaBeans mohou být instalovány na server stejným způsobem jako ostatní třídy.

Adaptivní komponentová architektura Sybase (Sybase Adaptive Component Architecture) vychází z toho, že vývoj aplikací na bázi komponent se dnes stává nosným způsobem vývoje aplikací a základní metodou umožňující zrychlit produkci software pomocí znovupoužitelného kódu.

Využitelnost javovských komponent (JavaBeans) ve všech aplikačních vrstvách včetně databází přenáší výhody Javy i do rozsáhlých enterprise aplikací.

3.4 Znovupoužitelnost - klíčová záležitost pro javovskou architekturu

Zabudovaný Java Virtual machine, jenž provádí javovské metody, umožňuje využívat vytvořené objekty v různých vrstvách systému.

Jeden javovský objekt, ať byl vytvořen přímo s použitím JDBC, pomocí RAD nástroje, nebo v JSQL,je možno využívat jak v klientských aplikacích a aplikačních serverech, tak uvnitř databázového serveru.

Pro maximální využití schopnosti Javy zvyšovat produktivitu je podstatné, že javovské třídy nemusejí být vytvářeny speciálně pro databázový server. Tato vlastnost je pro implementaci Sybase klíčovou.

3.5 Objektové datové typy v Javě

Kromě toho, že Java poskytuje podstatně širší funkcionalitu pro implementaci aplikační logiky, umožňuje její implementace do serveru překonat další omezení klasických relačních databází. Mimo jiné poskytuje rozšířenou škálu datových typů s mnohem všestrannějšími možnostmi.

Po instalaci javovské třídy na server ji je možno používat jako jiné datové typy pro sloupce tabulek. Každý takový sloupec se pak stává javovským objektem.

Jako jednoduchý příklad je možno uvažovat třídu obsahující adresy. Můžeme vytvořit třídu *Adresa* a instalovat ji na server. *Adresa* obsahuje název ulice, číslo domu, město, PSČ. Tabulky databáze pak mohou obsahovat sloupce s datovým typem *Adresa*. Jednotlivá pole adresy pak mohou být v dotazech použita samostatně. Výhody, jež využití javovských tříd přináší, je možno shrnout v následujících bodech:

- Zapouzdření vnitřní struktury adresy umožňuje snadno zajistit konsistenci informací v různých tabulkách (např. informace o zákaznících, zaměstnancích atd.).
- Příslušné třídy mohou obsahovat užitečné metody. V případě adresy to může být například dosazení názvu města podle hodnoty PSČ.
- Je možno využít dědičnosti tříd, například pro vytvoření různých metod pro definované podmnožiny řádků daného sloupce. Je možno např. vytvořit zvláštní třídy pro tuzemské adresy (CR_adresa) a zahraniční adresy (Int adresa), jež jsou odvozeny z obecnější třídy *Adresa*. Tyto třídy mohou například obsahovat další metody, kontrolující formální správnost PSČ atd. Do sloupce tabulky s datovým typem Adresa je pak možno vkládat objekty typu *Adresa, CR_adresa* i *Int adresa.*

3.5.1 Objektové datové typy a SQL

Pro přístup k databázím přes jazyk SQL existuje uznávaný standard JDBC. Pro přístup k javovským objektům ze SQL však obdobný standard zatím neexistuje. Sybase vyvinula pro tento účel pravidla vlastní a na obecné standardizaci spolupracuje s příslušnými autoritami.

Hlavním principem implementace Sybase je vývoj "no-surprises interface", tedy prostředí, jež garantuje, že všechny objekty pracují tak, jak se od nich očekává, i když byly použity v rámci příkazů SQL.

Příklady:

- Vložení nového řádku do tabulky:

```
INSERT INTO zaměstnanci (id, jméno, bydliště)
VALUES (1234,'Josef Novák',
            new Adresa (`Na kopečku 1163','182 00')
             )
```

- Vložení nového řádku pro mezinárodní adresu:

```
INSERT INTO zaměstnanci (id, jméno, bydliště)
VALUES (1235,'Joseph Newman',
           new Adresa(`3611 Hill Road','94608','US')
             )
```

- Dotaz, jenž zobrazí jméno zaměstnance a název ulice, ve které bydlí:

```
SELECT jmeno, Adresa.ulice
FROM zaměstnanci
WHERE id=1233
```

- Využití javovských objektů v jiných klausulích SQL; např. zobrazení zaměstnanců, bydlících v určité ulici:

```
SELECT jmeno
FROM zaměstnanci
WHERE Adresa.ulice='Tychonova'
```

3.6 Výkon a bezpečnost

Pro databázové systémy využívané v řešeních jsou stěžejní otázky výkonu a bezpečnosti. V prostředí Javy jsou tyto oblasti předmětem stálých diskusí.

3.6.1 Jak se vyrovnává Adaptive Server s otázkou bezpečnosti ?

Podniky a organizace využívají relačních databází mimo jiné pro to, aby jim zaručily vyšší bezpečnost dat. Produkty a řešení Sybase jsou tradičně zaměřeny na kritické systémy, mající v této oblasti vysoké nároky. Proto i při implementaci Javy do databázových serverů

byla problematika bezpečnosti řešena v tomto kontextu. Implementace zajišťuje stejnou míru bezpečnosti a robustnosti, na kterou jsme zvyklí z "klasických" relačních serverů. Pro ilustraci:

- Na serveru je možno používat pouze třídy nainstalované databázovým administrátorem
- Administrátor může stanovit, které metody a pole tříd je možno využívat v SQL
- Java Virtual Machine v Adaptive Serveru nepodporuje akce, jež mohou způsobit v databázovém serveru problémy se zabezpečením. (Např. file I/O).
- Java sama o sobě je navržena s ohledem na bezpečnostní potřeby. Je navržena jako ideální jazyk pro aplikace běžící v kritických systémech jakými jsou např. DBMS.

3.6.2 Jak se vyrovnává Adaptive Server s otázkou výkonu ?

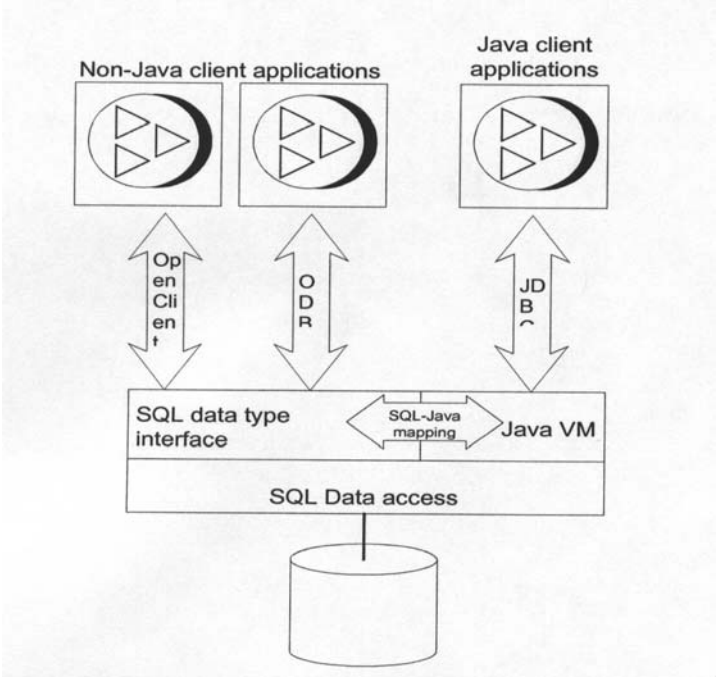
Otázka výkonu Javy je dnes rovněž s oblibou často diskutována. V tomto směru j e v obecné rovině očekáván zvrat v průběhu nejbližších měsíců.

V konkrétním případě implementace do prostředí Adaptive Serveru je však tento problém již v současné době minimalizován tím, že implementace Java Virtual Machine využívá všech zdrojů systému obdobným způsobem jako ostatní služby serveru. Třídy jsou v databázích ukládány ve zkompilované formě a administrátoři DBMS mohou využít všech obvyklých nástrojů k ladění výkonu.

3.7 Klientské aplikace a Java na serveru

Architektura Sybase je koncipována tak, aby v konkrétních řešeních respektovala již vložené úsilí a investice. Jednou z hlavních myšlenek Adaptive Serveru je zajištění provozu existujících aplikací v nezměněné podobě při současném využití nových možností a vlastností, jestliže jsou požadovány kvalitativně nové funkce.

Javovské funkcionality mohou využívat i stávající nejavovské klientské aplikace, jež komunikují se serverem prostřednictvím standardního ODBC nebo rozhraní Open Client. Klientské aplikace napsané v Javě mohou využívat dalších rozšiřujících možností.

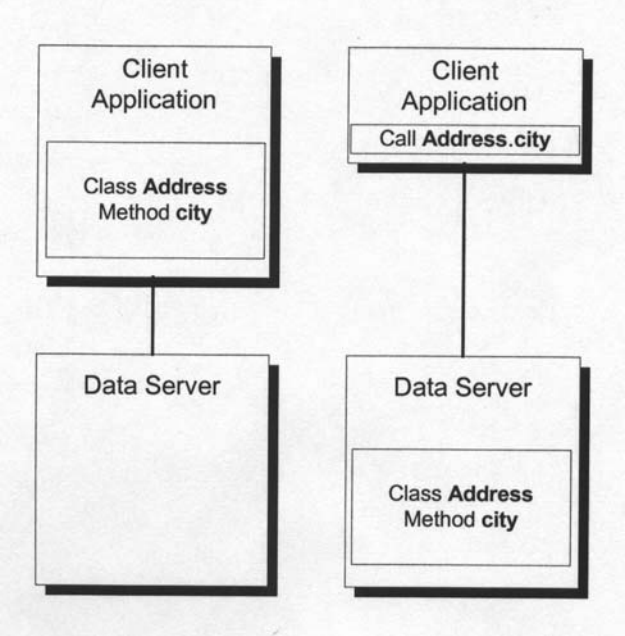


Podpora nejavovských klientů je zajištěna interním mapováním mezi Javou a SQL. Klienstká aplikace komunikuje se serverem pomocí standardního rozhraní, např. Sybase Open Client nebo ODBC. Podle javovských objektů mohou být zpřístupněna příkazy jazyka SQL a javovské metody spouštěny stejným způsobem jako uložené procedury.

3.8 Javovský klient a Java na serveru

Klientské aplikace napsané v Javě se k Adaptive Serveru mohou připojovat prostřednictvím rozhraní JDBC. Proti klasickým nejavovským aplikacím mají navíc možnost vyměňovat si se serverem objekty. Javovský objekt uložený na serveru může být vyžádán klientskou aplikací a serverem odeslán ke zpracování Java Virtual Machine na klientu. Jak již bylo zmíněno, javovská architektura Adaptive Serveru (na rozdíl od jiných implementací) nevyžaduje třídy vytvářené speciálně pro server. Třída může být použita v klientské aplikaci, na aplikačním serveru (ve střední vrstvě) i v databázi beze změny.

Jestliže má např. třída Adresa metodu město, která odvozuje název města z PSČ, může být tato metoda volána jak na klientské tak na serverovské straně.



4. Závěr

Cílem popsaného řešení je odstranit umělé bariéry, jež historicky oddělovaly různé domény architektury informačních systémů.

Java je dnes respektovaným a perspektivním prostředím pro vývoj aplikací a koncept JavaBeans a Enterprise JavaBeans představuje důležitý komponentový model pro budování javovských aplikací. Relační databáze jsou nepochybně jedním z nosných prvků řešení informačních systémů. Ukládání a využívání javovských objektů v relačních databázích otevírá nové možnosti pro efektivní budování a využívání informačních systémů.