Úvodem

Cílem je seznámit studenty se zajímavou databázovou technologií firmy InterSystems.

Caché je velmi vyspělý nástroj pro tvorbu komplexních aplikací založených na práci s persistentními (trvale dostupnými) daty. Na rozdíl od klasických relačních databází i čistě objektových databází je **Caché díky své architektuře uložení dat** schopna pracovat s daty **jak z pohledu objektového tak relačního**.

Kombinací těchto dvou základních přístupů Caché dává vznik samostatné kategorii databází, a to **databází postrelačních**. Tedy takových, které **umí pracovat s daty pomocí** jak **SQL jazyka** tak pomocí **objektové syntaxe**.

Pro Caché je velmi příznačné, že na rozdíl od např. objektově-relačních databází nedochází ke ztrátě výkonu při přístupu k datům pomocí objektové nadstavby neboť Caché nic takového nemá. Má jen 2[1] stejně výkonné projekce dat.

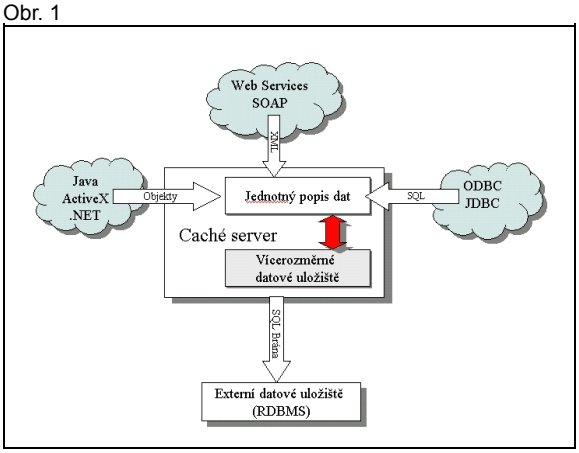
Unifikovaná architektura databáze

Základem modelování komponent aplikací jsou objekty. Objekty jsou organizovány do tříd ve kterých jsou popsány vlastnosti (data) a metody (chování) objektů.

Definice tříd jsou uloženy spolu s ostatními daty v jednotném uložišti zvaném Caché Class Dictionary (slovník tříd). Tento slovník tříd je sám o sobě databází a ostatní objekty k němu mohou objektově přistupovat.

V okamžiku zkompilování definice třídy dojde k vytvoření dvou různých, navzájem synchronizovaných sad kódu, které zajistí optimální přístup k instancím objektů třídy buď pomocí objektového nebo relačního přístupu.

Obr. 1



[1] Ve skutečnosti je počet projekcí dán jen počtem prakticky využívaných technologií, např. třetí projekcí je XML.

Definice tříd je možno v Caché vytvořit několika různými způsoby:

- "ručně" pomocí Caché Studia
- "relačně" pomocí DDL jazyka při zavolání DDL příkazu pomocí SQL. Caché automaticky vytvoří na základě definice tabulky definici třídy.
- Pomocí XML, Caché je schopno načíst XML dokument, má-li dokument správnou strukturu, Caché vytvoří definici třídy
- Programově, pomocí objektů. Caché obsahuje systémové třídy pro práci se slovníky tříd.
- Pomocí UML, Caché umí importovat a exportovat datové modely vytvořené v Rational Rose a Microsoft Visual Modelleru.

Přístup k definici uložení dat

Objektový model používaný Caché je v porovnání s objektovými modely jiných programovacích jazyků rozšířen o prvky z relačního prostředí. Lze tedy například definovat indexy, omezení (constrains) a strukturu uložení dat.

Definice fyzické stuktury uložení dat objektů je nezávislé na popisu třídy. Vyvojáři mají možnost vybrat si přednastavenou strukturu používanou kompilátorem tříd, nebo zvolit vlastní strukturu a ručně ji nadefinovat s ohledem na maximální výkon při dotazech nebo transakcích apod.

Objekty

Definování třídy

Caché disponuje plně vyvinutou objektovou databází pro práci v prostředích náročných na výpočetní výkon a množství zpracovaných transakcí.

Objektový model Caché zahrnuje mj:

Třídy: to jsou základní kameny objektového programování. Caché umožnuje definovat třídy uchovávající aplikační data a vykonávající aplikační logiku.

Vlastnosti: těmito jsou popisována data ukládaná a zpracovávaná třídami.

Metody: popisují chování objektů a vzájemnou interakci.

Relace: popisují vztahy mezi jednotlivými třídami v aplikaci.

Dědičnost: třídy definované v Caché mohou být odvozeny od jiných tříd a přejímat jejich vlastnosti a metody a další (např. parametry)

Různorodost (**polymorfismus**): třídy mohou modifikovat zděděné vlastnosti a metody.

Řetězení referencí: Caché automaticky při otevření instance třídy otevírá instance všech tříd které jsou ve vztahy k otevřené instanci (reference/relace)

SQL

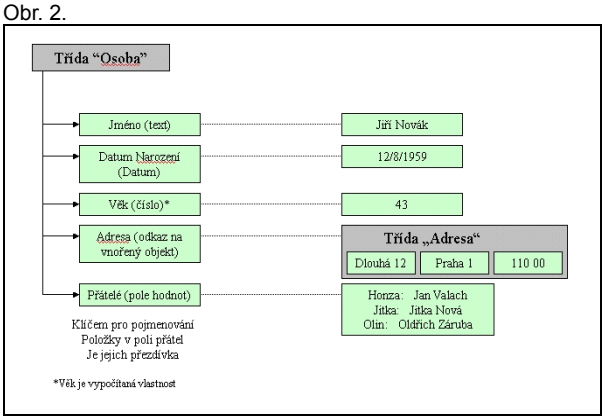
Vztah objektů a relačních tabulek

Jak již bylo uvedeno dříve, při kompilaci definice třídy se automaticky vytváří dvě sady kódu. První z nich je pro potřeby objektového přístupu (jak jinak, že) a druhá sada rutin definuje přístup k vlastnostem třídy jako ke sloupcům v tabulce.

Jedním z projevů nezávislosti objektové a relační projekce datového modelu je i možnost přidělit třídě alternativní název jakožto název tabulky. Totéž platí i pro vlastnosti třídy. Důvod je nasnadě. SQL obsahuje klíčová slova, která nemohou být použita jako názvy tabulek nebo sloupců, zatímco svět objektů je daleko svobodnější. Jen pro ilustraci, slovo USER je klíčovým slovem SQL a tedy jej nelze v názvu tabulky či sloupce použít, zatímco v objektovém datovém modelu je poměrně často použito pro název vlastnosti či třídy popisující např. uživatele aplikace.

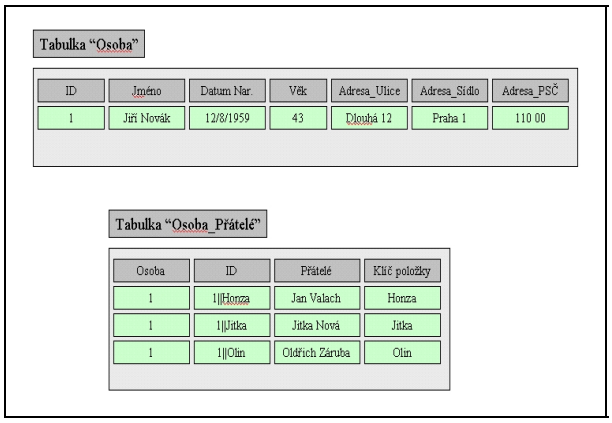
Každá vlastnost třídy která má jednu hodnotu je automaticky transformována na sloupec relační tabulky. Vlastnost typu pole, které nabývá libovolného počtu hodnot, je transformována na samostatnou tabulku obsahující odkaz na původní tabulku.

Obr. 2.



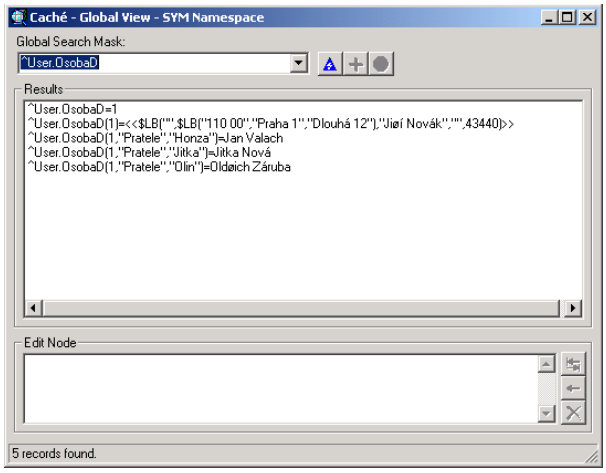
stejný datový model ale v relační projekci:

Obr. 3



Z uvedených obrázků je zřejmé, že mezi objektovou a relační projekcí téhož datového modelu mohou být značné rozdíly.

Obr. 4



Na tomto obrázku je ukázáno fyzické uložiště dat.

Fyzické uložiště dat se v Caché nazývá **globál**. Jedná se o vícerozměrné řídké pole.

Zatímco k zobrazení dat v klasických 2-rozměrných polích je nejvhodnější použít tabulkový pohled, k zobrazení vícerozměrných datových struktur je vhodný pohled stromový; tímto způsobem jsou prezentována data na předchozím obrázku. Vlevo od znaménka „=" jsou vidět jednotlivé „dimenze", vpravo pak data příslušná instanci objektu v dané dimenzi . Pole přátel je uloženo o úroveň níže než detaily osoby. Na druhé straně, detaily adresy jsou uloženy v rámci datových položek osoby ale v samostatném bloku ($lb).

Při kompilaci návrhu třídy Caché sama vytvoří kód, který zajistí přístup k fyzické datové vrstvě, programátor se vůbec o tyto vrstvu nemusí zajímat.

Nicméně, a tím se Caché výrazně liší od ostatních databázových nástrojů, vývojáři mohou sami definovat způsob uložení dat ve fyzické vrstvě. Díky této otevřenosti Caché je možno provádět optimalizaci / modifikaci uložení v případech kdy původní zvolený způsob ukládání dat není pro náročnost aplikace optimální. Je to velmi zřídka, ale stává se to. Nicméně s potřebou “ručně” optimalizovat se vývojáři setkají jen u opravdu rozsáhlých databází (GB) provádějících statisíce a více transakcí denně.

Návrh datového modelu pomocí Caché Studia

K návrhu složitých datových modelů je vhodné používat některý z profesionálních CASE nástrojů, například firmy Rational Rose, ze kterého je možno model importovat přímo do Caché.

U jednodušších datových modelů je možno začít definici třídy přímo v Caché Studiu. Stejně tak implementace kódu metod tříd vytvořených v CASE nástrojích se provádí v tomto studiu.

Při návrhu definice třídy má vývojář možnost použít průvodce pro jednotlivé dílčí úkoly nebo může psát definici přímo pomocí jazyka CDL (Class Definition Language)

Následující text obsahuje definice návrhu třídy User.Osoba.

Obr. 5

```
Class User.Osoba Extends %Persistent [ ClassType = persistent, ProcedureBlock ]
{

Property DatumNarozeni As %Date;

Property Jmeno As %String;

Property Vek As %Integer [ Calculated ];

Property Pratele As %String [ Collection = array ];

Property Adresa As User.Adresa;

/// vypočítá věk z data narození
Method VekGet() As %Integer
{
 // přibližný výpočet ...
 set vek=+$h-..DatumNarozeni\365
 Quit vek
}

/// Pozdraví
Method Pozdrav()
{
		write "Dobrý den!",!
		quit
}

}
```

Těla metod mohou být napsána pomocí Caché ObjectScriptu nebo CachéBasicu, což jsou skriptovací jazyky pro práci s daty a pro provádění aplikační logiky. Skriptovacím jazykům Caché bude věnován samostatný díl seriálu.

Detaily implementace objektů

Typy tříd

Objektová technologie implementovaná systémem Caché zahrnuje následující typy tříd:

- Abstraktní třídy
- Registrované třídy
- Persistentní třídy
- Vnořené třídy
- Odvozené třídy
- Datové typy

Nyní k popisu jednotlivých typů tříd:

Abstraktní třídy slouží k popisu obecných tříd, od nichž se v rámci aplikace odvozují třídy představující konkrétní modelované objekty. Abstraktním třídám nelze vytvářet instance objektů.

Registrované třídy implementují plnohodnotné objekty ve smlyslu OOP. Vzhledem k tomu, že nepodporují persistenci, využívají se zpravidla pro tvorbu kódu aplikační logiky.

Persistentní třídy slouží k popisu dat, která jsou trvale k dispozici, tedy dat ukládaných na disk nebo jiné záznamové médium. Každá persistentní třída je odvozena od třídy registrované, při kompilaci vytváří kód který zajišťuje komunikaci mezi aplikační vrstvou (objekt) a fyzickou vrstvou (globály). Příkladem persistentní třídy je třída *Osoba*.

Vnořené třídy jsou podobné třídam persistentním ale na rozdíl od nich nemají vlastní uložiště dat. Data obsažená ve vnořených třídách jsou vždy uložena spolu s daty persistentní třídy jež na vnořenou třídu odkazuje. Příkladem vnořené třídy je *Adresa*.

Odvozené třídy jsou obecně třídy odvozené od jedné nebo více uživatelských tříd (**Dědičnost**). Odvozené třídy obecně mohou modifikovat vlastnosti a metody svých nadtříd (**Polymorfismus**).

Datové typy jsou třídy které slouží jako popis dat. Datové typy mohou být libovolně složité. Úlohou datových typů je mj. zajistit správnou konverzi dat mezi zobrazovanou (externí) podobou a jejich uloženou (vnitřní) podobou, dále zajistit validaci vstupních hodnot aj.

Zvláštní kategorií tříd jsou CSP stránky, což jsou třídy sloužící pro tvorbu dynamických webových aplikací a Web Services jež implementují webové služby.

Vlastnosti, metody, vnořené dotazy

Vlastnosti tříd mohou být:

- Datový typ
- Jednoduchá reference na jinou třídu
- Jedna strana relace typu: 1-N, rodič-dítě

Vlastnosti mohou být privátní, vyžadující neprázdnou hodnotu, indexované, vyžadující unikátní hodnotu, transientní, vypočítané, vícerozměrné. Vlastnost může bát označena jako finální, takovou vlastnost nelze v odvozených třídách modifikovat.

Metody mohou být metodami instance nebo metodami tříd, (pak pro jejich volání není nutno otevírat instanci třídy). Je-li metoda metodou třídy může být zároveň označena jako uložená procedura SQL. Metody také mohou být, podobně jako vlastnosti, privátní a finální.

Třídy mohou definovat též vložené dotazy. Vložené dotazy mohou být definovány buď pomocí jazyka SQL nebo pomocí Caché ObjectScriptu. Výhodou vložených procedur je to, že jsou zkomilovány spolu s definicí třídy a tím se šetří čas při běhu programu. Vložené dotazy dovolují zadávání parametrů.

Pomocí parametrů tříd a vlastností lze modifikovat například způsob validace hodnot při ukládání třídy na disk.

Zdroje : Daniel Kutáč – firma InterSystems, Caché Databáze postrelačního typu a tvorba aplikací ISBN 80-251-0491-5