

Nebojme se objektových databází

MICHAL TOMEK

Vývoj databází a systémů pro zpracování dat si vynutily velké báze dat vznikající například při sčítání lidu nebo při vědeckotechnickém výzkumu. V druhé polovině 60. let šlo o převážně síťové systémy řízení báze dat (DBMS) pro sálové počítače a od 70. let i hierarchické databázové systémy.

Kolem roku 1975 se objevuje dotazovací jazyk SEQUEL (Structured English Query Language), později přejmenovaný na SQL. Relační databáze byly nakonec dotaženy do stadia vhodného pro komerční použití a staly se dominantní platformou. V roce 1979 byl uveden na trh systém Oracle, v roce 1983 DB2 od IBM, následované Sybase, Microsoft SQL serverem a dalšími.

pují pouze jako k množině bitů, které nemají pro databázi význam a neumějí s nimi nakládat.

Databáze relační, objektové a postrelační

V relačních databázích (RDBMS) jsou data uspořádána do tabulek, pro něž jsou definovány povolené operace. Součástí databáze tvoří také relace mezi jednotlivými tabulkami, na něž je odkazováno pomocí cizího klíče, což však při odkazování vede k nutnosti ošetřovat shodu dat (referenční integritu). Relační systémy jsou navrženy pro ukládání dat podle neefektivnější metody datové katalogizace definované teorií množin. Dobře si vedou tam, kde se ukládají do tabulek jednoduché datové typy s několika vztahy k datům v jiných tabulkách a jejich výhodou je přirozená reprezentace dat typu seznam a relativně snadné definování

témy. Ještě před nástupem a dominancí relačního modelu, existoval předrelační model databází, který ukládal data do vnořených tabulek a dalších struktur, které na sebe odkazovaly. Na tento přístup navázaly postrelační databáze, které se také vyznačují komplexní podporou objektů.

Objektové databázové systémy (ODBMS), vycházející z postupů objektivě orientovaného programování, který charakterizuje popis reality pomocí objektů a jejich metod a dalšími charakteristikami. Každý objekt je instancí nějaké třídy, jež je jeho abstrakcí určující datovou strukturu a operace (metody), které lze nad objektem provádět.

Klíčovou charakteristikou objektového modelu je dědičnost, umožňující dědění vlastností i metod mezi třídami. Zapouzdření pak umožňuje skrývat okolí vlastností a metody objektů, a tím zvyšují nezávislost objektu a brání nekonzistenci dat. Polymorfismus zajišťuje, že se jednotlivé operace mohou lišit podle třídy objektu, nad kterým jsou prováděny. S polymorfismem úzce souvisí možnost pozdní vazby, tedy takové volání operací, kdy aplikace dynamicky za chodu programu zvolí metodu pro zpracování na základě třídy objektu.

Kromě těchto charakteristik, převzatých z objektového programování, objektové databázové systémy speciálně řeší uchování dat, a to jak dočasně při běhu programu, tak trvale. Na rozdíl od relačních databází mohou objektové databáze perzistentně uchovávat libovolné objekty a přirozeně tak ukládat strukturovaná data i s jejich vztahy. Některé objektové databáze (Cached) podporují ukládání perzistentních objektů v libovolných strukturách a podle strategie specifikované uživatelem, případně také přes SQL bránu i do tabulek externích relačních databází.

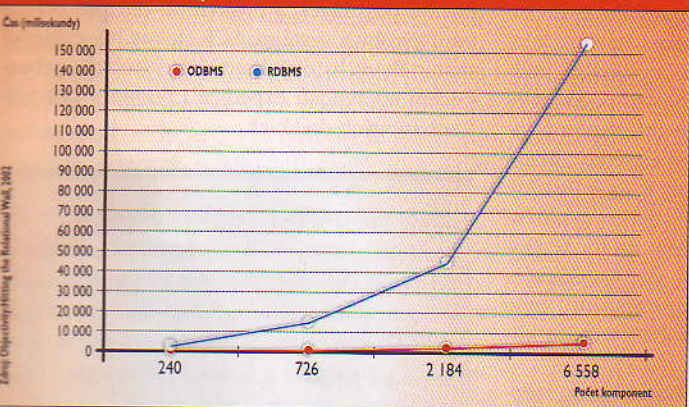
Relační zed'

Relační databáze byly a jsou velmi úspěšné. RDBMS přinesly uživatelům mnoho výhod, včetně ad hoc dotazování, nezávislosti dat na logických aplikacích, a velký výběr front-endových GUI nástrojů. Sloužily velkému počtu podnikových aplikací, a rozrostly se do rozsáhlého průmyslového odvětví s mnohamiliardovým ročním obrátem. Ale s rostoucí složitostí podnikových i jiných systémů dochází k tomu, že narážejí na pomyslnou hranici, za níž nejsou tak výkonné a funkční, jak by bylo potřeba. To se děje v okamžiku, kdy má databáze modelovat hlubší vztahy, zpracovávat a ukládat různé datové typy. Problémy se vyskytnou také tehdy, jde-li o distribuovaná prostředí s komplexními operacemi.

Pokusy zvládnout tuto složitost s relačními technologiemi, vedou k nárůstu počtu tabulek a jejich propojení, poklesu výkonu, malé škálovatelnosti a ztrátě integrity. Hranici pro použití relačních databází není jednoduché přesně načrtnout, ale projevuje se s nárůstem složitosti aplikací, kdy dochází k nesouladu mezi DBMS a aplikacemi. Dalším projevem nedostatečnosti RDBMS je ztráta flexibility, tedy obtíže při změnách aplikací, tak databází. Stejný problém nastává i při rozšiřování aplikací a datových modelů.

Limity relačního modelu ukazují i srovnávací testy. Výkonnostní testy mezi RDBMS a ODBMS

Graf 1: Vytváření komponent



V té době bylo hlavní oblastí použití DBMS udržování relativně jednoduchých podnikových dat, postupem doby se ale podnikové aplikace stávaly stále složitějšími a zpracovávané informace se stávaly strukturovanější, rostl počet vzájemných vztahů mezi daty (datové struktury skladů, výroby, správy součástek apod.). Od 90. let minulého století se v návaznosti na koncepci objektově orientovaných programovacích jazyků objevují také první objektové databáze, které si postupně našly své uplatnění především ve zdravotnictví a finančním sektoru, popř. výzkumu. Dochází i k částečnému sblížení obou platform, kdy relační databáze přejímají některé objektové postupy (i když povaha ukládání dat je fundamentálně odlišná).

V posledních letech, kdy dochází k masivnímu nárůstu informací i jejich typům, se projevuje další nedostatek relačních databází, a to nedostatečná podpora zpracování a ukládání velkých binárních objektů (BLOB – Binary Large Objects), jako jsou například audio a videosoubory nebo geografické informace. Relační databáze k nim někdy přistupují

vazeb. Dojde-li ale na data ukládaná v komplexních a vzájemně provázaných strukturách, nebo když musejí být data rychle obnovována a mění se jejich vzájemné vztahy, nelze vystačit s jednoduchým prohledáváním seznamů a relační databáze se musejí vypořádat s vícenásobnou správou indexů a komplexních schémat struktur.

Další komplikací je, že většina aplikací je dnes vytvářena v objektově orientovaných jazycích, a při práci s daty se tedy potýkají s problémem impedance nesouladu (impedance mismatch). Vzniká proto, že v relačních databázích nelze objekty ukládat přímo v jednoduchých tabulkách. Je proto nutné použít techniku objektově-relačního mapování (objektově-SQL), tedy provést dekompozici objektů a jejich konverzi do relačních tabulek, což vyžaduje značně úsilí a bývá to také zdrojem chyb. Součástí objektů však nejsou jen jejich vlastnosti, ale také metody, což situaci ještě více komplikuje.

Vzhledem k těmto a jiným nedostatkům relačního modelu vznikaly a rozvíjely se alternativní databázové modely. Patří mezi ně objektové (ODBMS) postrelační (PDBMS) databázové sys-

(databáze Oracle vs. Objectivity) odhalily, že zatímco v případě jednoduchých dat jsou obě databáze srovnatelné, při růstu složitosti se zřetelně projevují výhody objektového technologie. Při nárůstu komplexity se v případě RDBMS počet operací a tedy i výpočetní doba zvyšuje exponenciálně, kdežto ODBMS zachovávají lineární trend, hodnoceny přitom byly různé operace vytváření komponent, jejich aktualizace a mazání. V případě SQL dotazů jsou databáze srovnatelné pro jednoduchá data, ale pro složitá objektové jednoznačně předčí relační.

Přednosti a nevýhody

Mezi často zmiňované klady použití objektových databází patří menší náklady a časová náročnost. V závislosti na aplikaci může programový kód snadno narůst do více než poloviny aplikace. Aplikáčnický programátor musí psát kód, který převádí složité datové struktury do jednoduchých tabulek, což s sebou nese vyšší náklady a zabírá čas nutný pro programování, ztrátu integrity a výkonu. Při použití objektové technologie se naopak program aplikace zjednoduší, třeba o 30 až 40 procent. To má samozřejmě výhodu ve větší přehlednosti a menší chybovosti.

Výhody objektové technologie se ale neprojevují automaticky – vždy záleží na vhodném návrhu a dekompozici problému. Tento efekt je ještě výraznější v případě údržby. Studie ukazují, že až 80 procent nákladů na aplikace po dobu jejich životnosti tvoří náklady na údržbu – tedy přidávání nových funkcí, ladění, změna funkcí, což také vyžaduje změny datových modelů, a tím růst objemu práce a rizika chyb kvůli nutnosti mapování.

Dalším rizikem je ztráta integrity. Mapování z jednoduchých relačních dat do aplikace se uskutečňuje v aplikaci, a hrozí nebezpečí, že každý aplikační programátor bude mapovat odlišně, což vede k narušení integrity. Problém je i to, že mapování dělá aplikace. Naproti tomu v objektových databázích není mapování potřeba, protože databázový systém rozumí a přímo operuje s jednoduchými i složitými strukturami. Komplexita struktury je obsažena a zachycena přímo v databázi a je tudíž dostupná všem uživatelům bez rozdílu a je zaručena kompatibilita na všech úrovních.

Podpora SQL

Poněvadž SQL je de facto standardem, bylo pro objektové databáze nezbytné nabídnout také podporu tomuto dotazovacímu jazyku, který po-

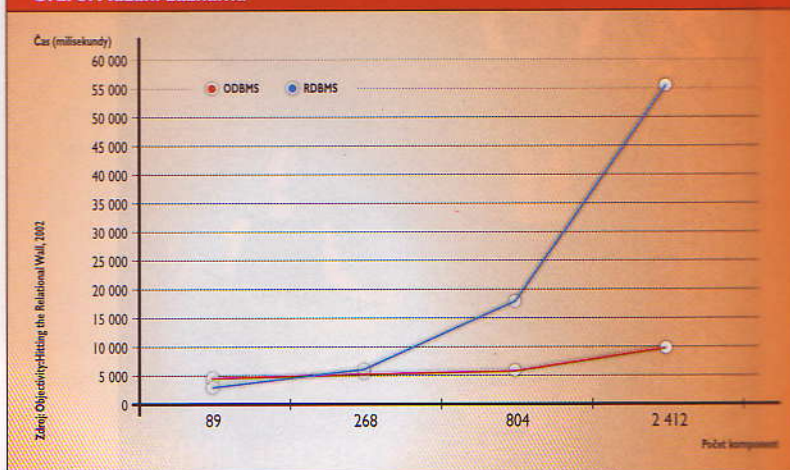
užívá mnoho aplikací a nástrojů. Objektové databáze podporují SQL automaticky, třída nebo typ se v SQL objeví jako tabulka, instance jako řádka tabulky, a vlastnost, operace nebo vztah se ukážou jako sloupce. Podpora stávajících aplikací vyžaduje plnou podporu SQL a všech jeho funkcí, ne jen dotazování, ale například bezpečnosti pomocí přidělování přístupových práv (s využitím zapouzdření atd.). V objektových databázích jsou možné standardní dotazy, používají se stejné indexovací techniky a operace. Většinou lze využít všech aspektů SQL, včetně náhledů, vkládání, aktualizace, smazání, triggerů a uložených procedur.

Objektové databázové systémy propojují objektový přístup s dalšími technologiemi jako je perzistence, transakční zpracování, SQL apod., takže je možné jimi v mnoha oblastech relační databáze nahradit. Relační databáze však dobře poslouží pro mnoho aplikací, hlavně tam, kde se často nemění struktura databáze a na vznik nových datových typů. Relační databáze mohou bez problémů zpracovávat velká množství nestrukturovaných dat uložených v jednoduchých tabulkách. Objektové databáze se uplatní při potřebě robustních a škálovatelných databází, kde vyniknou jejich výhody.

Z globálního pohledu jsou objektové databáze zatím málo rozšířené, a to především z následujících důvodů. Prvním z nich je konzervativní myšlení, které favorizuje tradiční a rozšířené relační systémy, a dosud malá obeznámenost s objektovými databázemi mezi vývojáři. Nelze rovněž opomenout nejednotné specifikace a chybějící standardy, i když i o tom se už jedná. Třeba standard ODMG 3.0 není všemi výrobci respektován. Chybí rovněž například kompletní podpora všech konstrukcí pro modelování objektové databáze v UML apod.

Objektové databáze zatím nacházejí uplatnění hlavně ve výzkumných střediscích, zdravotnictví a finanční oblasti. Mnohé se však využívají nejen jako systémy pro ukládání dat, ale ve spojení s dalšími aplikacemi slouží jako komplexní integrační platforma v organizaci, která je schopná přes různá dodávaná rozhraní přistupovat ke zděděným datům a aplikacím a sloužit jako univerzální datová sběrnice organizace. Impulzem pro rozvoj objektové databázové platformy by se také mohla stát open source komunita a její aktivity.

Graf 3: Mazání záznamů



Obavy z objektových databází mohou rozptýlit příklady jejich nasazení. Objektovou technologii používá třeba CERN, mezinárodní středisko provádějící náročné fyzikální experimenty. Jeho urychlovač nové generace vyžadoval nejrozsáhlejší databázi na světě, která je dimenzována až na 100 PB informací (sto milionů GB), a po výběru došel CERN k závěru, že jeho požadavky může splnit jen objektová technologie (tedy Objectivity). Hlavním důvodem pro použití objektové databáze byla v tomto případě potenciální velikost a distribuce.

Dalším příkladem živé implementace je zdravotní organizace Partners Healthcare v USA, jež nasadila sérii více než 100 aplikací Caché a jednu terabytovou produkční databázi obsluhující 33 000 uživatelů. Důvodem použití bylo, že když v Partners Healthcare analyzovali datový model lékařské objednávky, zjistili, že relační model by vyžadoval více než 700 tabulek. ■

OBJEKTOVÉ DATABÁZE

- db4objects (db4o)
- GemStone (GemStone/S)
- Intersystems (Caché)
- Matisse (Matisse)
- ObjectDB (ObjectDB)
- Objectivity (Objectivity/DB)
- Progress (ObjectStore, PSE Pro)
- Versant (merged with Poet; VDS, FastObjects)
- W3apps (Jeevan)

OPEN SOURCE

- db4o
- Ozone
- Perst

Michal Tomek je ředitelem InterSystems ČR. Zastihnout ho můžete na adrese tomek@intersystems.cz

Zdroj: ODBMS.ORG

Graf 2: Aktualizace

