

KTERÁ JE TA PRAVÁ?

Porovnání základních typů databází

Cílem tohoto článku je seznámit čtenáře se základními kategoriemi databází a připomenout historické okolnosti jejich vzniku a alespoň zhruba popsat silné stránky různých typů databází, jejich základní slabiny, aby si čtenář mohl udělat lepší představu o tom, po jaké z nich (kategorii) sáhnout při vývoji nových aplikací.

Již v prvopočátcích využití počítačů (2. světová válka) bylo potřeba někde uchovávat zpracovávaná data. Původními aplikacemi uchovávanými data byly vojenské, knihovní a medicínské systémy. Data byla uchovávána různým způsobem. Dalo by se říci, co aplikace, to jiný způsob uchování a manipulace s daty. Nicméně v 60. letech doznaly počítače již takového rozšíření, že si je mohly pro své potřeby dovolit i soukromé firmy, a s tím vyvstala potřeba nějak sjednotit způsob práce s daty.

V zásadě se vyvinuly dva směry: síťový datový model (CODASYL) a hierarchický datový model (IMS). K datům se přistupovalo nízkourovňově pomocí ukazatelů na (zřetězené) záznamy. Programátoři s daty pracovali na základě znalosti fyzické struktury databáze a měli prakticky neomezené možnosti libovolně navrhovat strukturu dat. To s sebou ovšem neslo nepříjemné důsledky:

» při změně struktury záznamů se muselo přepsat aplikační rozhraní tak, aby odráželo novou strukturu ukládaných dat,

» jelikož neexistovala metadata popisující záznamy, docházelo často k nesprávným interpretacím významů jednotlivých polí v záznamech anebo k jejich duplikaci, tak jak do aplikace zasahovali v průběhu času různí vývojáři.

Počátkem 70. let publikoval E. F. Codd¹ popis databáze založené na popisu vztahů mezi pevně danými typy záznamů – tabul-

kami. Tím položil základ k oddělení schématu popisujícímu data a jejich fyzického způsobu uložení. Jeho práce se stala standardem pro celou kategorii databází, které jsou dnes na trhu nejrozšířenější.

V roce 1976 navrhl P. Chen² entitně-relační model návrhu databázových struktur a zhruba od poloviny 80. let již všechny relační databáze používaly jednotný jazyk pro práci s daty Structured Query Language (SQL).

S masivním nástupem objektově orientovaných programovacích jazyků dochází v 90. letech minulého století k realizaci myšlenky, nač používat dva různé typy myšlení při psaní aplikací, když by bylo jednodušší pracovat objektově i s ukládanými daty. Objevily se první objektové databáze.

Aby toho nebylo málo, s masivním rozvojem internetu se zrodil i univerzální popisný jazyk XML (eXtended Markup Language) jako jednodušší derivát staršího, ale pro praktické použití nevhodného (neboť příliš složitého) jazyka SGML³.

Shrnuto a podtrženo: dnes se můžeme setkat s těmito hlavními typy databází (řazeno podle doby vzniku):

Hierarchické databáze

Data jsou v těchto databázích ukládána ve formě datových stromů, tedy ve strukturách podobných např. adresářovým strukturám souborových systémů, s nimiž pracuje drtivá většina dnes používaných operačních systémů. Datové položky jsou přístupné prostřednictvím identifikátorů – uzlů stromu.

+ Možno modelovat velmi složité struktury, neexistence formálních omezení.

Základní typy databází

Typ databáze	Hlavní představitelé
Hierarchické	ANSI M
Relační	Oracle, Ingres, Sybase, DB2, MySQL
Objektové	Poet, DB4O, O2, GemStone, Caché
Objektově - relační	Oracle, MSSQL
XML	Tamino
Jiné	Caché, Terradata, Universe

+ Velmi rychlé prohledávání datových stromů i na různých úrovních větvení.

- Neexistuje standardní dotazovací jazyk, každá databáze implementuje své nástroje pro dotazování a manipulaci s daty.

- Předsudky ze strany většiny databázových programátorů, považujících neprávem tento typ databází za zastaralý.

Relační databáze

Dnes nejrozšířenější typ databází. Data jsou ukládána v tabulkách, jež mohou být vzájemně provázané.

+ Velmi dobře matematicky popsané pomocí relační algebry.

+ Standardizovaný jazyk SQL (Structured Query Language) pro dotazování.

- Relační model je zjednodušeným pohledem na modelovanou skutečnost a někdy je kvůli jeho principiálním omezením nutno buď velmi zjednodušit popisovanou skutečnost, nebo tvořit složité a nepřehledné modely s pomocnými tabulkami.

- Jazyk SQL má velmi omezené možnosti pro manipulaci s daty (uložené procedury versus metody tříd).

- Různí výrobci implementují proprietární rozšíření SQL. Tím dochází k degradaci obecnosti SQL jazyka a závislosti programátorů aplikací na konkrétní implementaci databáze.

Objektové databáze

+ Velmi dobře jsou schopny postihnout modelovanou skutečnost.

+ Mají velmi dobré prostředky pro manipulaci s daty, jelikož k ní využívají

plnohodnotných objektově orientovaných programovacích jazyků typu Java, C++, .NET.

■ Neexistuje matematický popis jako mají relační databáze, tudíž neexistuje žádný pevný standard pro návrhy modelů⁴.

■ Objektové struktury jsou mnohem náročnější na paměť, na rozdíl od relačních databází, které pracují se skalárními proměnnými. Zatímco SQL pracuje pouze se sloupci, které jsou zadány v příkaze SELECT, objekty se otvírají vždy se všemi svými vlastnostmi. Z toho plyne, že objektové databáze jsou mnohem pomalejší při dotazování a prohledávání velkých sad záznamů.

Objektově-relační databáze

Tyto databáze jsou vlastně pouhými nepřeliš povedenými pokusy o křížence mezi objektovou a relační databází. V jádře jsou to relační databáze a objekty jsou implementovány pouze na úrovni datových typů sloupců.

➤ Možnost práce s abstraktními datovými typy (datové typy s komplexnější strukturou, např. XML dokumenty).

■ Svým proprietárním rozšířením SQL vnášejí nejednotnost a závislost tvůrců aplikací na konkrétní implementaci databáze.

XML Databáze

Jedná se o specializované databáze určené pro ukládání XML dokumentů a pro provádění dotazovacích operací nad hierarchicky prezentovaným datovým modelem XML dokumentů, zejména prostřednictvím XML jazyků XPath a XQuery. Z uvedeného je zřejmé, že jejich uplatnění v aplikacích je poměrně omezené.

Jiné typy databází

» Prostorové databáze – specializované databáze pro práci s geografickými údaji, optimalizované pro ukládání obrovských objemů dat.

» Temporální databáze – specializované databáze pro práci s časově (případně i prostorově, pak se mluví o spaciotemporálních databázích) proměnnými daty, používané například

ke studiu fyzikálních jevů. Viceméně jsou zatím pouze ve stádiu prototypů.

Caché

Caché představuje samostatnou kategorii databází. Jedná se o produkt spojující v sobě aplikační server a univerzální databázi, podporující plnohodnotnou práci s persistentními objekty, stejně jako dotazování pomocí SQL, přístup k datům pomocí XML nebo přímý přístup k hierarchickým datovým stromům. Technicky vzato, data jsou ukládána právě v hierarchických datových stromech, ale prostřednictvím manipulačního kódu, který se generuje automaticky při tvorbě datových modelů, lze k těmto datům přistupovat jako by to byly objekty, či relační tabulky nebo XML dokumenty. Vše záleží jen na rozhodnutí programátora: v jakém okamžiku je pro aplikaci lépe pracovat s daty relačně, či objektově. Zpravidla ovšem pro vyhledání cílové instance objektu poslouží rychlý SQL dotaz a poté, co je objekt nalezen (respektive jeho ID), pracuje se s ním již objektově. Caché tedy kombinuje „to nejlepší“, co nabízí jak relační, tak objektové databáze. Navíc, ve vícevrstvých aplikacích nedochází k nesouladu mezi technologií aplikačního serveru a databázového serveru (o tom pojednávají další odstavce).

Jistě by se daly vyjmenovat ještě i další typy databází, ale jejich kategorie je zpravidla dána způsobem jejich použití (např. OLAP analýzy). My se jimi zde zabývat nebudeme.

Databáze jako součást informačního systému

Každá firma, bez výjimky, potřebuje k efektivnímu řízení provozu informační systém. Nezáleží na tom, zda se jedná o výrobní podnik, nemocnici nebo obchodní organizaci. Každý den činnosti firmy vyprodukuje spousty dat, která je nutno ukládat a zpracovávat. K tomu slouží právě informační systémy, jejichž nedílnou součástí jsou databáze.

Informační systémy jsou z hlediska architektury zpravidla zastoupeny těmito konfiguracemi:

- » terminálové aplikace
- » aplikace klient – server
- » tří- a vícevrstvé aplikace (klient – aplikační server – databázový server).

Terminálové aplikace

Terminálové aplikace jsou nejstarší architekturou. Tvoří je znaková konzole s minimální inteligencí a server, provádějící veškerou aplikační logiku a ukládání dat. Tento typ řešení je stále životaschopný, neboť funkčně omezená konzole neodvádí pozornost obsluhy od práce a na druhé straně využívá výpočetní sílu centrálního serveru (nebo clusteru serverů), přičemž

je jedno, jaký typ databáze je na serveru provozován.

Nehodí se však příliš pro manažery, z důvodu absence grafického rozhraní. V posledních několika letech je však možno vidět masivní nástup terminálových řešení nové generace, kde původní konzole je nahrazena webovým prohlížečem, který nabízí plnohodnotné grafické prostředí a také pokročilé funkce pro správu uživatelského rozhraní.

Aplikace typu klient – server

Právě absence grafického prostředí terminálových řešení vedla k masivnímu rozvoji aplikací typu klient – server, kde klient (označovaný jako „tlustý“) obstarává jak uživatelskou interakci, tak aplikační logiku a server se stará pouze o ukládání dat. Tento typ aplikací vyhovuje právě relačním databázím.

Vícevrstvé aplikace

Aplikace klient – server mají jednu nectnost, a to je nutnost poměrně zdlouhavé instalace klienta na každou uživatelskou stanicí a také nezbytnost dostatečně tyto stanice hardwarově dimenzovat. Stejně tak při každé aktualizaci aplikace je nutno aktualizovat i všechny klienty. To byl jeden z důvodů, proč se od tohoto konceptu v případě větších instalací (řádově stovky klientů) přešlo na architekturu vícevrstvou. Klient (zvaný „tenký“) v takovém uspořádání obstarává pouze uživatelskou interakci, aplikační logiku má na starosti vyhrazený server (aplikační server) a o ukládání dat se stará databázový server. Aplikační server je typicky postaven na platformě Java nebo .NET, zatímco databázovým serverem je (z důvodu maximalizace výkonu pro dotazování a ukládání dat) relační databáze.

Toto uspořádání ale vnáší do aplikace jeden velmi rušivý element – paradigmatický nesoulad mezi objektovým aplikačním serverem a relačním datovým serverem. Tento nesoulad je zdrojem více práce nutné pro mapování objektů aplikačního serveru na relační tabulky a zpět a navíc je to možný zdroj problémů při aktualizaci verzí aplikačního/datového modelu.

Nevyčerpejme čtenáře

Tento článek není vyčerpávajícím popisem jednotlivých typů databází, nicméně snaží se alespoň stručně vyzdvihnout to, co jednotlivé typy databází nabízejí, a pro jaké konfigurace aplikací jsou vhodné. Zůstává na čtenáři, aby při posuzování vhodnosti té které databáze pro své účely vzal v úvahu všechna pro a proti jednotlivých řešení a zodpovědně vybral správnou technologii, která bude minimalizovat náklady na vývoj a údržbu a maximalizovat užitek pro koncové uživatele. □

Reference

- ¹ E. F. Codd: A Relational Model of Data for Large Shared Data Banks
- ² P. Chen: The Entity-Relationship Model – Towards a Unified View of Data
- ³ Standard Generalized Markup Language – ISO8879
- ⁴ Za defacto standard se bere F. Kaufman: The Object Database Standard: ODMG-V2.0.