Relační databáze

Z Wikipedie, otevřené encyklopedie

Tento článek z oblasti informatiky potřebuje úpravy. Můžete Wikipedii pomoci tím, že ho vhodně vylepšíte (http://cs.wikipedia.org/w/index.php?title=Rela%C4%8Dn%C3%AD_datab%C3%A1ze&action=edit) .
Jak by měly články vypadat, popisuje stránka Vzhled a styl.

Relační databáze je databáze založená na relačním modelu. Často se tímto pojmem označuje nejen databáze samotná, ale i její konkrétní softwarové řešení.

Relační databáze je založena na tabulkách, jejichž řádky obvykle chápeme jako záznamy a eventuelně některé sloupce v nich (tzv. cizí klíče) chápeme tak, že uchovávají informace o relacích mezi jednotlivými záznamy v matematickém slova smyslu.

Termín *relační databáze* definoval Edgar F. Codd v roce 1970.

Obsah

Historie

V roce 1890 vzniká na objednávku státních úřadů v USA první automat na bázi děrných štítků. U jeho zrodu stál Herman Hollerith, jehož firma při fůzi několika firem dává vzniknout IBM. Ta stojí v popředí i v roce 1951, kdy vzniká první digitální počítač pro komerční využití UNIVAC I. V roce 1960 vzniká předchůdce dnešních databázových jazyků COBOL. V 60. letech zakládá Charles Bachman spolu s dalšími výzkumníky seskupení Codasyl, které publikuje základní specifikaci pro programovací jazyky, především pro COBOL. Většina Codasyl kompatibilních databází byla postavena na síťovém modelu, zatímco firma IBM se vydala cestou

hierarchického modelu. V roce 1970 přichází Ted Codd s novým návrhem datového modelu, relačním modelem. Dle relační teorie lze pomocí základních operací (sjednocení, kartézský součin, rozdíl, selekce, projekce a spojení) uskutečnit veškeré operace s daty a ostatní operace jsou již jen kombinacemi těchto pěti. Zavádí tedy použití relačního kalkulu a algebry. Databáze mají být nezávislé na fyzickém uložení dat i na použitém jazyce. Pod tlakem událostí se do projektu vkládá i IBM, která přichází s jazykem SQL. První SQL databází byl v roce 1980 Oracle pro počítače VAX. Druhá v řadě přichází i firma IBM s produktem DB2. Osmdesátá léta lze považovat za zlatý věk databází[zdroj?].

Terminologie

Základem relačních databází jsou databázové tabulky. Jejich sloupce se nazývají atributy nebo pole, řádky tabulky jsou pak záznamy. Atributy mají určen svůj konkrétní datový typ - doménu. Řádek je řezem přes sloupce tabulky a slouží k vlastnímu uložení dat. Konkrétní tabulka pak realizuje podmnožinu kartézského součinu možných dat všech sloupců - relaci.

Primární klíč

Primární klíč je jednoznačný identifikátor záznamu, řádku tabulky. Primárním klíčem může být jediný sloupec či kombinace více sloupců tak, aby byla zaručena jeho jednoznačnost. Pole klíče musí obsahovat hodnotu, tzn. nesmí se zde vyskytovat nedefinovaná prázdná hodnota NULL. V praxi se dnes často používají umělé klíče, což jsou číselné či písmenné identifikátory - každý nový záznam dostává identifikátor odlišný od identifikátorů všech předchozích záznamů (požadavek na unikátnost klíče), obvykle se jedná o celočíselné řady a každý nový záznam dostává číslo vždy o jednotku vyšší (zpravidla zcela automatizovaně) než je číslo u posledního vloženého záznamu (číselné označení záznamů s časem stoupá).

Cizí klíč

Dalším důležitým pojmem jsou nevlastní/cizí klíče. Slouží pro vyjádření vztahů, relací, mezi databázovými tabulkami. Jedná se o pole či skupinu polí, která nám umožní identifikovat, které záznamy z různých tabulek spolu navzájem souvisí.

Integrita databáze

Integrita databáze znamená, že data v ní uložená jsou konzistentní vůči definovaným pravidlům. Lze zadávat pouze data, která vyhovují předem definovaným kriteriím (např. musí respektovat datový typ nastavený pro daný sloupec tabulky, či další omezení hodnot přípustných pro daný sloupec). K zajištění integrity slouží integritní omezení. Jedná se o nástroje, které zabrání vložení nesprávných dat či ztrátě nebo poškození stávajících záznamů v průběhu práce s databází. Například je možné zajistit mazání dat, která již ztratila svůj význam - například smažeme-li uživatele, odstraní se i zbytek jeho záznamů v ostatních databázových tabulkách.

Druhy integritních omezení

- *Entitní integritní omezení* – povinné integritní omezení, které zajišťuje úplnost primárního klíče tabulky;

zamezí uložení dat, která neobsahují všechna pole sdružená do klíče, nebo data, jež by v těchto polích byla stejná jako v nějakém jiném, již zapsaném, řádku tabulky
- *Doménová integritní omezení* – zajišťují dodržování datových typů/domén definovaných u sloupců databázové tabulky
- *Referenční integritní omezení* – zabývají se vztahy dvou tabulek, kde jejich relace je určena vazbou primárního a cizího klíče
- *Aktivní referenční integrita* – definuje činnosti, které databázový systém provede, pokud jsou porušena některá pravidla

Dodržování integritních omezení

V zásadě existují tři způsoby, jak zajistit dodržování integritních omezení.

1. umístění jednoduchých mechanismů pro dodržování integritních omezení na straně databázového serveru. Jedná se o nejlepší způsob z hlediska ochrany dat. Uživateli však obvykle přináší delší odezvu systému a nelze vždy zajistit jejich přenositenost na jiný databázový systém.
2. umístění ochranných mechanismů na straně klienta. Pro komfort a nezávislost na databázovém systému je nejlepší volbou. Nutnost kontrolních mechanismů pro každou operaci může způsobit chyby u aplikací a v případě většího počtu aplikací je potřeba je opravit na více místech.
3. samostatné programové moduly na straně serveru. V moderních databázových systémech jsou pro tento účel implementovány tzv. triggery. Jedná se o samostatné procedury, které lze spouštět automatizovaně před a po operacích manipulujících s daty. Tento způsob umožňuje implementaci i složitých integritních omezení. Nevýhody opět přináší provádění na serveru, i velmi omezená možnost přenesení na jiný databázový systém.

Ideálním řešením je kombinace předchozích v závislosti na konkrétních podmínkách.

Kontroly integritních omezení se zpravidla provádějí po každé provedené operaci, což snižuje nároky na server. Není nutno nijak zaznamenávat, které kontroly mají být provedeny později. Složitější integritní omezení však vždy nelze takto ověřit, proto je možné kontrolovat dodržení pravidel až po dokončení celé transakce.

Vztahy mezi tabulkami

Vztahy, neboli relace, slouží ke svázání dat, která spolu souvisejí a jsou umístěny v různých databázových tabulkách. V zásadě rozlišujeme čtyři typy vztahů.

1. mezi daty v tabulkách není žádná spojitost, proto nedefinujeme žádný vztah.
2. 1:1 používáme, pokud záznamu odpovídá právě jeden záznam v jiné databázové tabulce a naopak. Takovýto vztah je používán pouze ojediněle, protože většinou není pádný důvod, proč takovéto záznamy neumístit do jedné databázové tabulky. Jedno z mála využití je zpřehlednění rozsáhlých tabulek. Jako ilustraci je možné použít vztah řidič - automobil. V jednu chvíli (diskrétní časový okamžik) řídí jedno auto právě jeden řidič a zároveň jedno auto je řízeno právě jedním řidičem.
3. 1:N přiřazuje jednomu záznamu více záznamů z jiné tabulky. Jedná se o nejpoužívanější typ relace, jelikož

odpovídá mnoha situacím v reálném životě. Jako reálný příklad může posloužit vztah autobus - cestující. V jednu chvíli cestující jede právě jedním autobusem a v jednom autobuse může zároveň cestovat více cestujících.

4. M:N je méně častým. Umožňuje několika záznamům z jedné tabulky přiřadit několik záznamů z tabulky druhé. V databázové praxi bývá tento vztah z praktických důvodů nejčastěji realizován kombinací dvou vztahů 1:N a 1:M, které ukazují do pomocné tabulky složené z kombinace obou použitých klíčů (třetí resp. tzv. vazební tabulka). Příkladem z reálného života by mohl být vztah výrobek - vlastnost. Výrobek může mít více vlastností a jednu vlastnost může mít více výrobků. V reálném životě nicméně existuje velké množství vztahů M : N, mimo jiné také proto, že často existuje praktická potřeba zachovávat i údaje o historii těchto vztahů z časového hlediska (jeden řidič v delším časovém období řídí více rozličných aut a jedno auto v delším časovém období může mít více různých řidičů).

Normální formy

Pod pojmem *normalizace* rozumíme proces zjednodušování a optimalizace navržených struktur databázových tabulek. Hlavním cílem je navrhnout databázové tabulky tak, aby obsahovaly minimální počet redundantních dat. Správnost navržení struktur lze ohodnotit některou z následujících normálních forem.

1. Nultá normální forma (0NF) - tabulka v nulté normální formě obsahuje alespoň jeden sloupec (atribut), který může obsahovat více druhů hodnot.
2. První normální forma (1NF) - tabulka je v první normální formě, pokud všechny sloupce (atributy) nelze dále dělit na části nesoucí nějakou informaci neboli prvky musí být atomické. Jeden sloupec neobsahuje složené hodnoty.
3. Druhá normální forma (2NF) - tabulka je v druhé normální formě, pokud obsahuje pouze atributy (sloupce), které jsou závislé na celém klíči.
4. Třetí normální forma (3NF) - tabulka je ve třetí normální formě, pokud neexistují žádné závislosti mezi neklíčovými atributy (sloupci).
5. Čtvrtá normální forma (4NF) - tabulka je ve čtvrté normální formě, pokud sloupce (atributy) v ní obsažené popisují pouze jeden fakt nebo jednu souvislost.
6. Pátá normální forma (5NF) - tabulka je v páté normální formě, pokud by se přidáním libovolného nového sloupce (atributu) rozpadla na více tabulek.

Externí odkazy

- Článek na Root.cz - historie databází (http://www.root.cz/clanky/historie-relacnich-databazi/)
- Úvod do relačních databází (http://programujte.com/index.php?akce=clanek&cl=2007110801-teoreticky-uvod-do-relacnich-databazi)
- Jiří Kosek: Co je to databáze (http://www.kosek.cz/clanky/iweb/12.html)
- en:Edgar F. Codd (anglicky)
- en:Charles Bachman – en:Navigational database (anglicky)

Kategorie: Databáze