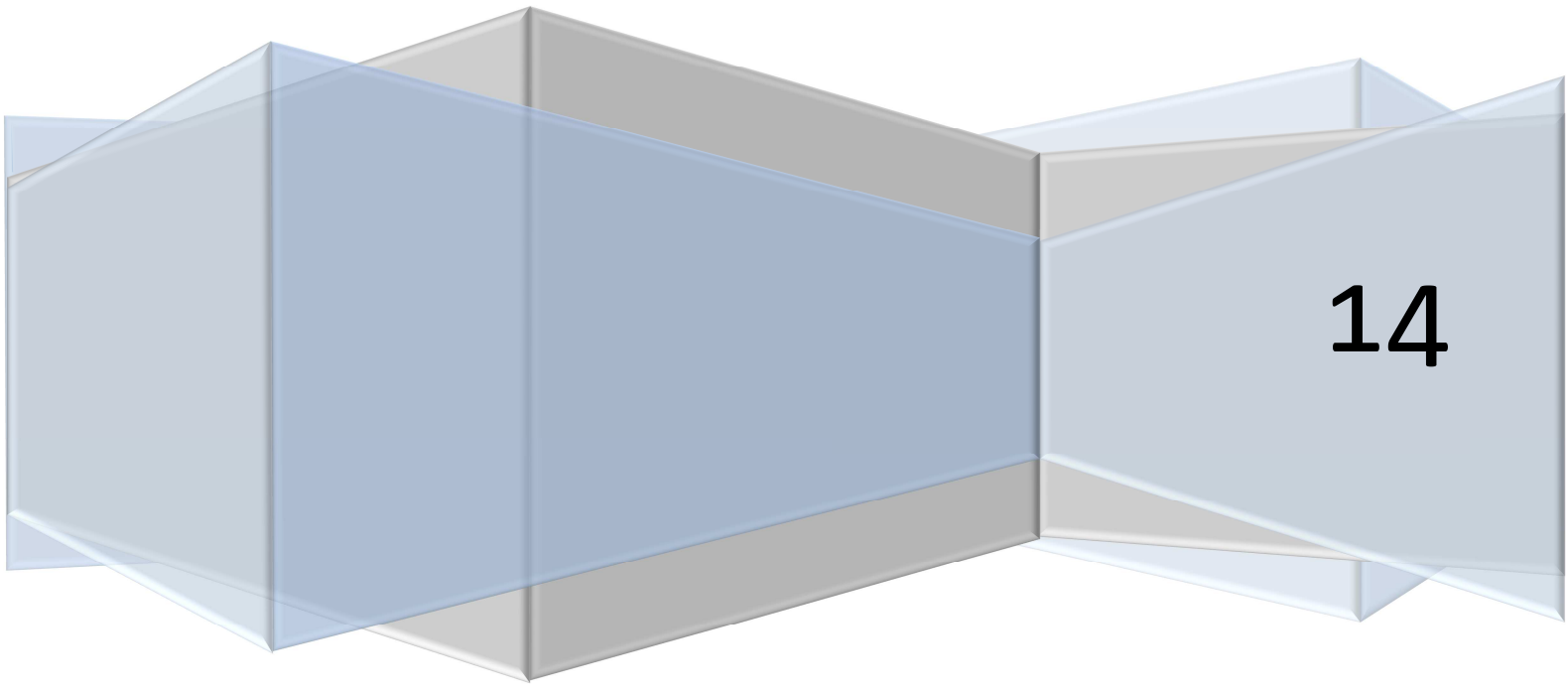
SQL

Seznámení s SQL

HP



14

Úvod

Tato práce je zaměřena na seznámení s SQL a jeho popisem. Seznámíme se zde se základními funkcemi a jednoduše si je popíšeme pro potřeby učení nebo školení.

Obsah

1. Principy relační databáze

Co je databáze

Databáze je kolekce vzájemně souvislých datových položek, které jsou spravovány jako jediná jednotka. Jednotlivý výrobci databází existují mnohé odchylky v implementaci. Jednou z největších výhod relačních databází je však to, že podrobnosti o fyzické implementaci jsou odděleny od logické definice databázových objektů, takovým způsobem, že většina uživatelů nemusí vědět kde (nebo jak) jsou databázové objekty v systému souborů počítače uloženy.

Co je to databázový řídící systém

Systém řízení báze dat (SŘBD) je software, který dodávají výrobci databází. SŘBD poskytuje všechny základní služby vyžadované pro databáze. Uspořádání a údržbu databáze a to včetně následujících:

- Přesouvání dat z a do fyzických datových souborů podle potřeby.
- Řízení souběžných přístupů k datům více uživateli včetně opatření k zamezení kolizí souběžných aktualizací.
- Řízení transakcí tak, aby se všechny změny databáze způsobené transakcí provedly zcela nebo vůbec.
- Podpora databázových jazyků
- Opatření pro zálohování databáze a obnova databáze po selhání
- Bezpečnostní mechanismy.

Co je relační databáze

Relační databáze je databáze založena na relačním modelu. Který vyvinul Dr. E. F. Codd. Relační model prezentuje data v dobře známých dvourozměrných tabulkách. Na rozdíl od tabulkových procesů však není nutné ukládat data v tabulkovém tvaru. Model dovoluje také kombinovat tabulky tak, aby mohly vytvářet pohledy.

Komponenty relační databáze

Základní součástí relačních databází jsou:

Tabulky

V relační databázi je hlavní jednotkou pro ukládání dat **tabulka**, což je dvourozměrná struktura, která se skládá z řádků a sloupců. Každá tabulka představuje entitu, což je osoba, místo, věc nebo událost, která je v databázi znázorněna.

Relace

Relace představuje souvislosti mezi tabulkami relační databáze. Zatímco každá relační tabulka může existovat samostatně, databáze jsou především o ukládání souvisejících dat.

Omezení

Omezení je pravidlo aplikované na databázový objekt, které nějakým způsobem omezuje přípustné hodnoty dat tohoto databázového objektu. Jakmile se jednou nastaví, je toto nastavení vynucováno. Každé omezení má své jedinečné jméno. Existuje několik typů omezení:

- **Omezení NOT NULL** – pokud je definováno pro sloupce databáze, znemožní použít hodnotu null.
- **Omezení primárního klíče** – jeho definice pro sloupce primárního klíče zajistí, že jsou hodnoty primárního klíče v rámci tabulky vždy jedinečné.
- Omezení jedinečnosti – definuje se ve sloupci nebo skupině sloupců tabulky, které musí obsahovat jedinečné hodnoty.
- **Referenční omezení** - toto omezení vynucuje relaci mezi dvěma tabulkami v relační databázi. Vynucuje, znamená že ŘSBD automaticky kontroluje, zda pro všechny hodnoty cizího klíče existují odpovídající hodnoty primárního klíče v nadřazené tabulce.
- **Omezení CHECK** – ověřuje hodnotu sloupce pomocí jednoduchého logického příkazu.

Pohledy

Pohledy je uložený databázový dotaz, který uživateli databáze poskytuje přizpůsobenou podmnožinu dat z jedné nebo více databázových tabulek. Jedná se o virtuální tabulku. Pohledy plní několik funkcí:

- Skrývají sloupce, které uživatel nepotřebuje
- Skrývají z tabulky řádky, které uživatel nepotřebuje
- Skrývají komplexní databázové operace
- Zvyšují výkon dotazu.

Postup návrhu relační databáze

Zde si popíšeme jednotlivé pohledy na proces návrhu databáze:

Normalizace

Je proces, při kterém se jednotlivá data v tabulce upravují vzhledově, aby byla přehledné jejich zpracování.

- **Anomální vkládání** - Je situace, kdy nelze do databáze vložit data kvůli umělé závislosti mezi sloupci v tabulce.
- **Anomální odstranění** - jedná se o přesný opak anomálie vkládání. Jedná se o situaci, kdy odstranění data způsobí neúmyslnou ztrátu dat.
- **Aktualizační anomálie** – se vztahuje k situaci, kdy aktualizace jediného údaje vyžaduje aktualizaci více řádků.

Použití normalizačního procesu

Při normalizaci se obvykle vychází z libovolného zobrazení dat, jak jsou (nebo budou) poskytována uživateli. Aplikuje se na každý uživatelský pohled. Je nutné dbát na to ,aby všechny ukázková data, na jejich základě se při normalizaci rozhoduje. Skutečné reprezentovat typ hodnot, které se budou vykytovat v reálných datech. Zde je popis:

- Volba jedinečného identifikátoru
- První normální forma: vyloučení opakovaných dat
- Druhá normální forma: vyloučení částečných závislostí
- Třetí normální forma: vyloučení tranzitivních závislostí

2. Koncepce jazyka SQL

Tato kapitola obsahuje základní informace o jazyku SQL.

Co je SQL

SQL je standardní jazyk pro komunikace s relačními databázemi. Dotaz je požadavek, který se odesílá databázi. Na základě tohoto požadavku databáze poskytne odpověď. SQL se řadí do skupiny neprocedurálních neboli deklarativních jazyků. To znamená, že počítači sdělíte, jaké výsledky požadujete.

Kategorie příkazů jazyku SQL

Příkazy SQL se v závislosti na své funkci dělí do několika kategorií:

- **Jazyk DDL (Data Definition Language),**
- **Jazyk DQL (Data Query Language),**
- **Jazyk DML (Data Manipulation Language),**
- **Jazyk DCL (Data Control Language).**
- **Příkazy řízení transakcí.**

Jazyk DDL (Data Definition Language)

Jazyk DDL zahrnuje příkazy SQL, které umožňují uživatelům databáze vytvářet databázové objekty (např. tabulky, pohledy a indexy) a upravovat je. Patří, jsem příkazy CREATE, ALTER A DROP.

Jazyk DQL (Data Query Language)

Jazyk DQL obsahuje příkazy SQL, které načítají data z databáze. Zde existuje pouze jediný příkaz SELECT.

Jazyk DML (Data Manipulation Language)

Součástí jazyku DML jsou příkazy, které umožňují uživatelům přidávat data do databáze, odebírat a měnit stávající data. Patří, jsem INSERT, UPDATE a DELETE.

Jazyk DCL (Data Control Language)

Jazyk DCL patří příkazy, které správcům dovolují řídit přístup k datům v databázi a používat různá systémová oprávnění SŘBD, jako funkce spuštění nebo vypnutí databáze. Patří jsem příkazy GRANT A ALTER.

Příkazy řízení transakcí

Databázová transakce je sada příkazů, které databázový uživatel požaduje zpracovat jak nedělitelnou jednotku. To znamená, že transakce musí být kompletně úspěšná nebo neúspěšná.

3. Definování databázových objektů

Zde se seznámíme s příkazy SQL, které umožňují definovat a spravovat databázové objekty v relační databázi. Nyní se seznámíme s jednotlivými způsoby ukládání dat.

Datové typy

Datový typ je kategorie, která určuje formát příslušného sloupce. Mají několik zásadních výhod:

- Omezují data v sloupci na znaky, které mají smysl z hlediska daného datového typu.
- Poskytují sadu vlastností, které jsou užitečné pro uživatele databází.
- Pomáhají relačními systému řízení báze dat při efektivním ukládání dat sloupců

Jazyk SQL podporuje tři základní datové typy: předem definované, složené a uživatelsky definované typy. Předem definované datové typy jsou k dispozici jako nativní součást SŘBD. Složené datové typy, které se také označují jako typy kolekcí, uchovávají pole nebo sady hodnot předem definovaných konstrukcí SŘBD. Uživatelsky definované datové typy, které umožnují uživatelům databáze definovat vlastní datové typy.

Standardní datové typy

- **Znakové datové typy –** ukládají řetězce znaků. Znakem může být libovolné písmeno, číslice nebo jiný symbol.
- **Číselné datové typy –** tyto datové typy ukládají pouze čísla.
- **Časové datové typy –** uchovávají údaje, které nějakým způsobem určují čas.

- **Typy pro velké objekty –** ukládají údaje, jejichž velikost značně přesahuje možnosti dosud popsaných typů.
- **Další datové typy –** Jednotlivý dodavatelé mají vlastní rozšíření pro své databáze.

Hodnoty NULL a tříhodnotová logika

Při definování sloupců v databázových tabulkách můžeme určit, zda budou v daném sloupci povoleny hodnoty **null**. Hodnota **null** v relační databázi je speciální kód, který lze umístit do sloupců. Tento kód znamená, že hodnota sloupce v daném řádku není známa.

Příkazy jazyku DDL (Data Definition Language)

Tito příkazy definují databázové objekty, ale neumožnují vkládat ani aktualizovat data, která jsou v těchto objektech uložena. Jazyk má tři základní klíčová slova:

- **CREATE** – vytvoří nový databázový objekt,
- **ALTER** – změní definici existujícího databázového objektu,
- **DROP** – odstraní (zruší) existující databázový objekt.

4. Načítání dat pomocí jazyk DQL

Jazyk DQL obsahuje pouze jeden příkaz, který však velmi důležitý a je to příkaz SELECT. Pomocí něho lze načítat data z databáze. Data se pak dále zpracovávají. Nejjednodušší příkaz SELECT obsahuje dvě klauzule:

- SELECT (DISTINCT) – uvádí sloupce, které mají být součástí sady výsledků.
- FROM – obsahuje seznam tabulek nebo pohledů, z kterých budou data vybrána.

Klauzule FROM není ve skutečnosti tak snadná jak se zdá. Většina relačních systémů řízení báze dat poskytuje funkce, které vracejí data na systémové úrovni.

Řízení výsledků

Výsledky dotazů jsou často užitečnější, pokud určíte takové pořadí, které vyhovuje příjemci informací. Chcete-li zaručit konkrétní pořadí řádků v sadě výsledků, musíte požadované pořadí uvést v syntaxi. Ve jazyku SQL to lze provést přidáním klauzule ORDER BY do

příkazu SELECT. Klauzule obsahuje seznam jednoho nebo více sloupců, pomocí nichž budou datové hodnoty ve sloupcích řazeny ve vzestupném nebo sestupném pořadí.

Filtrování řádků pomocí klauzule WHERE

Klauzule WHERE jazyku SQL umožnuje vybrat řádky, které budou zobrazeny. Pokud uvedeme klauzuli WHERE, vyhodnotí se tato klauzule pro každý řádek dat na základě pravidel Booleovské algebry.

Operátory porovnání

Operátory porovnání v klauzuli WHERE umožnují porovnat dvě hodnoty. Výsledkem je logická hodnota „**true**“ nebo „**false**“. Lze porovnat dvě konstanty zadané v klauzuli WHERE, hodnoty sloupců v databázi, nebo kombinaci konstanty a hodnoty. V následující tabulce jsou uvedeny operátory porovnání:

Operátor	Popis
=	Rovná se
<	Menší než
<=	Menší než nebo se rovná
>	Větší než
>=	Větší než nebo se rovná
!=	Nerovná se
<>	Nerovná se (standard ANSI)

Spojovací operátory

Někdy je potřeba zúžit sadu výsledků dotazů pomocí více podmínek. Použijeme-li více podmínek, musíme je v klauzuli WHERE logicky zkombinovat. K tomu souží spojovací operátory. Jsou dva:

- **AND** – výsledkem vyhodnocené klauzule WHERE je hodnota „true“, pokud jsou pravdivé všechny podmínky spojené operátorem AND.
- **OR** – Klauzule WHERE se vyhodnotí jako „true“, jestliže má hodnota“ true“ libovolná z podmínek spojených operátorem OR.

Pokud kombinujeme operátory AND a OR ve stejné klauzuli WHERE, je situace poněkud složitější.

Logické operátory

Logické operátory používají při porovnání místo symbolů klíčová slova:

- **IS NULL** – pomocí tohoto operátoru lze zjistit, zda je hodnota null. Je důležité si uvědomovat, že hodnoty null v databázi se nerovnají ničemu jinému, ani jiným hodnotám null
- **BETWEEN** – tento operátor dovoluje zjistit, zda hodnota patří do určitého rozsahu. Rozsah se určuje pomocí minimální a maximální hodnoty a tento rozsah je inkluzivní. To znamená, že minimální a maximální hodnoty jsou součástí rozsahu.
- **LIKE** – umožnuje porovnat znakové hodnoty se vzorem. Pokud znaková hodnota odpovídá vzoru, vrátí logickou hodnotu „true“, opačném případě „false“.
- **IN** – umožnuje zjistit, zda hodnota patří do seznamu hodnot. Seznam hodnot může obsahovat hodnoty laterálu, jejichž seznam je oddělen čárkami a uzavřen do uvozovek, nebo lze hodnoty vybrat z databáze pomocí příkazu subselect.
- **EXISTS** – umožnuje určit, zda příkaz subselect obsahuje nějaké řádky. Pokud zde nějaké najde, zobrazí se logická hodnota „true“ a pokud ne tak „false“.

Aritmetické operátory

Aritmetické operátory slouží v jazyce SQL k matematickým výpočtům podobně jako ve vzorcích tabulkového procesoru nebo v programovacích jazycích. Existují čtyři aritmetické operátory:

Operátor	Popis
+	Sčítání
-	Odečítání
*	Násobení
/	Dělení

Základní funkce SQL

Funkce je speciální typ programu, který při každým spuštění vrací jedinou hodnotu. Termín vychází z matematické koncepce funkce. Funkce v jazyku SQL přijímají pouze výraz, který často obsahuje název sloupce.

Znakové funkce

Znakové funkce získaly svůj název protože, pracují se znakovými daty.

- **Řetězení řetězců** – spojuje více znakových řetězců dohromady a vytvářejí ve výsledcích dotazu jedinou hodnotu sloupce-
- **UPPER** – změní znakový řetězec tak, aby obsahovat výhradně velká písmena. Čísla a speciální znaky zůstávají beze změn.
- **LOWER** – funguje přesně opačně než funkce UPPER. Převede všechna písmena ve znakovém řetězci na malá.
- **SUBSTR** – funkce vrací část řetězce na základě parametrů, které definují název sloupce, první (počáteční) pozici vrácených dat ve sloupci a počet vrácených znaků (délka řetězce).
- **LENGTH** – vrací délku znakového řetězce.

Matematické funkce

Matematické funkce manipulují s číselnými hodnotami podle matematických pravidel. Seznam podporovaných matematických funkcí:

- **ROUND** - Funkce zaokrouhluje číslo na určený počet desetinných čísel.

Další tabulka obsahuje nejobvyklejší matematické funkce.

Funkce	Popis
ABS	Absolutní hodnota daného čísla
COS	Trigonometrický kosinus daného úhlu (v radiánech)
EXP	Exponenciální hodnota daného čísla
POWER	Umocní číslo na příslušnou mocninu (číslo i mocnina se uvádí jako parametry)
SIN	Trigonometrický sinus daného úhlu (v radiánech)
TAN	Trigonometrický tangens daného úhlu (v radiánech)

Konverzní funkce

Konverzní funkce převádějí data na jiný datový typ.

- **CAST** – převádí data z jednoho typu na jiný
- **CONVERT TO** – jedná se o stejnou funkci jako CAST.

Agregační funkce a seskupení řádků

Agregační funkce kombinuje více řádků dat do jediného řádku. Viz následující tabulka

Název funkce	Popis
AVG	Vypočítá průměrnou hodnotu sloupce nebo výrazu.
COUNT	Zjistí počet hodnot nalezených ve sloupci. Chcete-li místo celkového počtu hodnot (řádků) ve sloupci zjistit počet jedinečných hodnot, zadejte klíčové slovo DISTINCT.
MAX	Vyhledá maximální hodnotu ve sloupci.
MIN	Vyhledá minimální hodnotu ve sloupci.
SUM	Sečte hodnoty ve sloupci.

Klauzule GROUP BY

Klauzule GROUP BY, zajistí, že SŘBD sestaví řádky vybrané dotazem do skupin na základě hodnot v jednom nebo více sloupcích. Poté aplikuje agregační funkci na každou skupinu a vrátí v sadě výsledků jeden řádek pro každou skupinu.

Operátory složených dotazů

Někdy se hodí integrovat výsledky více dotaz do jediné sady výsledků.

- **UNION –** připojí všechny řádky výsledků jednoho dotazu k sadě výsledků jiného dotazu.
- **UNION ALL –** funguje stejně jako UNION, al v tomto případě nejsou ze sady výsledků eliminovány duplicitní řádky.
- **INTERSECT –** vyhledá hodnoty vybrané jedním dotazem, které se zároveň vyskytují ve výsledcích druhého dotazu.
- **EXCEPT –** je operátor standardu ANSI/ISO, který nalezne rozdíly mezi dměna sadami výsledků.

5. Kombinace dat z více tabulek

Nyní se seznámíme s kombinací dat z více tabulek.

Spojení

Spojení je operace relační databáze, která kombinuje sloupce ze dvou nebo více tabulek do jediného výsledku dotazu.

Ekvivalentní spojení

Ekvivalentní spojení nebo vnitřní spojení páruje jeden nebo více sloupců z jedné tabulky s podobnými sloupci ve druhé tabulce pomocí podmínky rovnosti.

- Spojení pomocí klauzule WHERE – lze přirovnat k vyloučení nežádoucích řádků ze sady výsledků.
- Spojení pomocí klauzule JOIN – se zapisuje jako odkaz na tabulky v klauzuli FROM.

Přirozené spojení

Přirozené spojení je založeno na všech sloupcích v obou tabulkách, jejichž názvy se shodují. V zásadě platí, že mezi přirozené spojení patří i ekvivalentní spojení.

Vnější spojení

Vnější spojení zahrnuje do výsledku dotazu nespárované řádky alespoň z jedné z tabulek. Pro nespárované řádky platí, že datové hodnoty vybrané z tabulky, která neobsahuje odpovídající řádek, jsou nastaveny na hodnotu null. Existují tří základní typy vnějších spojení:

- Levé vnější spojení vrátí všechny řádky v tabulce na levé straně spolu s případnými řádky tabulky na pravé straně, které lze spárovat.
- Pravé vnější spojení vrátí všechny řádky v tabulce na pravé straně spolu s případnými řádky tabulky na levé straně, které lze spárovat.
- Úplné vnější spojení vrátí všechny řádky z obou tabulek.

Vlastní spojení

Vlastní spojení je spojení tabulky s toutéž tabulkou.

Další spojení

Většina spojené patří mezi ekvivalentní. Někdy však není nutné párovat spojované řádky pomocí ekvivalentní podmínky. Na tomto místě je ale vhodně upozornit, že připojení, která nejsou ekvivalentní, přinášejí více výkonnostních problémů.

Křížové spojené

Křížové spojení není nic jiného než standardní syntaxe pro kartézský součin.

Příkazy subselect

Velmi silnou funkci jazyka SQL představují příkazy subselect (neboli pod dotazy). Příkazy subselect se obvykle používají v klauzulích WHERE, kde omezují počet řádků vrácených v sadě výsledků vnějšího dotazu. Tímto způsobem se vybírají data.

Nekorelované příkazy subselect

Nekorelovaný příkaz subselect se vyznačuje tím, že vnitřní příkaz SELECT neodkazuje na vnější příkaz, ve kterém je vložen.

Korelované příkazy subselect

Korelovaný příkaz subselect je takový příkaz subselect, ve kterém vnitřní výběr odkazuje na hodnoty poskytované vnějším výběrem.

6. Složitější dotazy

Nyní se seznámíme s jazykem DML.

Pokročilé funkce SQL

Seznámíme se s dalšími funkcemi.

Znakové funkce

Znakové funkce zpracovávají znaková data. Nyní se s nimi seznámíme.

- **REPLACE** – funkce prohledá znakový řetězec a nahradí nalezené znaky za znaky, které jsou uvedeny v řetězci pro nahrazení.
- **LTRIMM** – odstraní ze znakového řetězce všechny úvodní mezery. Funkce odebere pouze úvodní mezery.
- **RTRIMM** – stejná jako funkce LTRIMM, ale odstraňuje koncové mezery.
- **ASCII** – funkce ASCII vrátí hodnotu ze znakové sady ASCII pro znakové řetězce, který obsahuje jediný znak.
- **CHAR** – vrátí znak přidružený k hodnotě ASCII.

Matematické funkce

Matematické funkce vrátí výsledek matematické operace. Obvykle jako vstupní parametr požadují číselný výraz, což může být hodnota laterálu, číselná hodnota sloupce tabulky nebo libovolný jiný výraz, který poskytuje číselnou hodnotu.

- **SIGN** – funkce přijímá číselný výraz a na základě znaménka vstupního čísla vrátí jednu z následujících hodnot:

Vrácená hodnota	Význam
-1	Vstupní číslo je záporné
0	Vstupní číslo je rovno nule
1	Vstupní číslo je kladné
null	Na vstupu je hodnota null

- **SQRT** – přijímá jeden číselný výraz a vrátí jeho druhou odmocninu.
- **CEILING (CEIL)** – vrátí nejmenší celé číslo, které je větší nebo rovno hodnotě číselného výrazu uvedeného jako vstupní parametr.
- **FLOOR** – je logickým protikladem funkce CEILING. Vrátí celé číslo, které je menší nebo rovno hodnotě číselného výrazu zadaného jako vstupní parametr.

Datové a časové funkce

V implementaci datových a časových funkcí se různí dodavatelé značně rozcházejí. Důvodem bylo že tito datové a časové funkce existovali dříve než vznikl příslušní standard.

Výraz CASE

Výraz CASE byl do standardu SQL doplněn teprve nedávno, ale je velice důležitý. Poprvé lze části příkazů SQL spouštět podmínečně. Existují dva druhy příkazu:

- **Jednoduchý výraz CASE** -
- **Prohledávaný výraz CASE** – poskytuje při porovnání vyšší pružnost, protože všechny podmínky v rámci příkazu jsou kompletní a obsahují i operátor porovnání.

7. Jazyk DML

Zde se seznámíme s jazykem DML. Tato část jazyku SQL umožnuje správu dat, která jsou uložena v relačních tabulkách databáze. DML zahrnuje tři příkazy jazyku SQL:

- **INSERT** – přidává do databázové tabulky nové řádky.
- **UPDATE** – aktualizace existující řádky databázové tabulky.
- **DELETE** – odstraňuje řádky z databázové tabulky.

Příkazy jazyku DML mohou pouze manipulovat pouze s daty jedné tabulky. V příkazu DML lze odkazovat na pohled, včetně takového pohledu, který obsahuje data z více tabulek. Většina systému (SŘBD) nabízí určitý typ podpory transakcí, kdy lze posloupnost příkazů SQL DML považovat za skupinu, která musí být provedena kompletně nebo vůbec.

Příkaz INSERT

Příkaz INSERT v jazyku SQL umožňuje přidávat nové datové řádky do tabulky. Má dvě základní formy: jedna z nich obsahuje hodnoty sloupců ve vlastním příkazu a v druhé formě se hodnoty vybírají z tabulky nebo pohledu pomocí příkazu subselect.

Příkaz Update

Příkaz UPDATE v jazyku SQL umožňuje aktualizovat datové hodnoty ve sloupcích tabulky, které jsou v příkazu.

Příkaz DELETE

Příkaz DELETE odebere jeden nebo více řádků z tabulky. Příkaz může také odkazovat na pohled, ale musí se jednat o pohled založený na jediné tabulce. Příkaz DELETE nikdy neodkazuje na sloupce, protože odebírá celé řádky včetně datových hodnot na každém zpracovávaném řádku.

8. Zabezpečení pomocí příkazu Jazyk DCL

V této kapitole si představíme některé obecné koncepce zabezpečení.

Bezpečnostní architektura databází

Jeden z problémů správců databází od různých dodavatelů spočívá v tom, že s výjimkou produktů Microsoft SQL Server a Sybase Adaptive Server nemají žádné dvě databáze stejnou bezpečnostní architekturou.

Implementace zabezpečení přístupu k databázi

Účelem zabezpečení přístupu k databázi je chránit data před neoprávněným použitím. Předpokladem je, aby správce databáze určil, jaké akce mohou jednotliví uživatelé provádět databázovými objekty.

Databázová oprávnění

Oprávnění uživatelům databáze určité činnosti s databází. Obecně platí, že když uživatel databáze dostane oprávnění, aby se mohl k databázi připojit, nemůže provádět žádné akce, dokud nedostane určitá dodatečná oprávnění.

- **Systémové oprávnění** – jsou obecné oprávnění k funkcím pro správu serveru a databáze či databází.
- **Objektové oprávnění** – objektové oprávnění povolují provádět určité činnosti s konkrétními databázovými objekty.

Příkazy SQL používané při správě zabezpečení

Popisuje příkazy jazyku SQL, které slouží ke správě zabezpečení přístupu k datům.

- Příkaz CREATE USER – mnozí dodavatelé databází umožňují spravovat zabezpečení pomocí grafického uživatelského rozhraní.
- Příkaz GRANT – příkaz GRANT dovoluje udělit uživateli databáze jedno nebo více oprávnění.

Účty vlastníka schématu

Nezávisle na použité databázi je žádoucí neposkytovat uživatelům databází více oprávnění, než kolik potřebují k plnění svých úkolů. Tím se zabraňuje poškození dat kvůli lidským chybám, ale navíc díky tomu uživatelé nemohou podlehnout pokušení.

Zjednodušení správy pomocí rolí

Role je pojmenovaná kolekce oprávnění, které lze následně udělit jednomu nebo více uživatelům. Většina systému RSŘBD se standardně dodává s předem definovanými rolemi a uživatelé databáze, kteří mají oprávnění CREATE ROLE, mohou vytvářet své vlastní role.

- Role mohou existovat dříve než uživatelské účty.
- Role šetří správci mnoho rutinní práce
- Role přetrvají i odstranění uživatelského účtu.

Implementace zabezpečení na úrovni sloupců a řádků pomocí pohledů

Při zabezpečení se běžně řeší, jal umožnit uživatelům databáze přístup k některým řádkům a sloupcům tabulky, ale zároveň v přístupu k jiným řádkům a sloupcům. Lze toho dokonale dosáhnout pomocí pohledů. Některé výhody:

- Z pohledu je možné vynechat sloupce, které uživatel databáze nepotřebuje. Pokud uživatelům udělíte přístup k pohledu místo výchozí tabulky, kompletně jim zabráníte zobrazit informace ve sloupcích, které jste z pohledu vypustili.
- Do pohledu lze zahrnout klauzuli WHERE, která omezí počet vrácených řádků. Řádky můžete také odfiltrovat pomocí spojení, které je spáruje s jinou tabulkou.
- Pomocí spojení s „vyhledávacími“ tabulkami lze nahradit hodnoty kódů v tabulce odpovídajícími popisy.

9. Zajištění integrity databáze pomocí transakcí

Zde si ukážeme, jak lze jazyk SQL integrovat do aplikaci vytvořených například v jazyku Java a dalších.

Co je databázová transakce

Transakce je oddělená sada akcí, které je nutné zpracovat jako celek, nebo vůbec. Transakce se někdy označuje jako pracovní jednotky, což zdůrazňuje jejich nedělitelnou povahu. Jednotlivé vlastnosti transakcí si můžete zapamatovat pomocí anglického akronymu ACID (Atomicity, Consistency, Isolation, Durability – atomicita, konzistence, izolace, trvanlivost):

- Atomicita Transakcí musí zůstat vcelku. Tj. musí být buď kompletně úspěšná, nebo kompletně neúspěšná.
- Konzistence Transakce by měla převést databázi z jednoho konzistentního stavu do jiného.
- Izolace Každá transakce by měla plnit svou funkci nezávisle na všech ostatních transakcích, které mohou existovat současně.
- Trvanlivost Změny provedené dokončenou transakcí by měly být trvalé, i po následném vypnutí nebo chybě databáze či jiné klíčové součásti systému.

Podpora transakcí v relačních systémech řízení báze dat (RSŘBD)

Podporu transakcí nabízí většina relačních systémů báze dat s výjimkou databázových systémů pro osobní počítače. Jsou příkazy SQL pro označení začátku a konce transakce a také funkci protokolování všech změn prováděných jednotlivými transakcemi, aby je bylo možné v případě potřeby zrušit. Problém u transakcí je ten, že byly navrženy dříve, než byl vytvořen standard.

Zamykání a uváznutí transakcí

Souběžné sdílení dat mezi mnoha uživateli databází sice přináší zásadní výhody, ale má také vážnou nevýhodu, která může způsobit ztrátu aktualizovaných dat.

Problém souběžné aktualizace

Problém souběžné aktualizace, ke které také dochází, když lze stejná data aktualizovat z více databázových relací. Problém souběžné aktualizace nejčastěji nastává u dvou různých databázových uživatelů, kteří nevědí o tom, že provádějí konfliktní aktualizaci stejných dat.

Mechanismus zamykání

Zámek je kontrolní mechanismus, kterým databáze vyhradí data, aby je mohl aktualizovat pouze jeden uživatel. Když jsou data zamčena, nemůže je aktualizovat žádná jiná databázová relace, dokud není zámek uvolněn. K tomu se obvykle používá příkaz COMMIT nebo ROLLBACK jazyka SQL.

Jednotlivé úrovně zámků:

- Databáze je uzamčena – tj. že celá databáze je zamčena a možnost aktualizace má pouze jedna databázová relace.
- Soubor je uzamčen – tj. že je uzamčen celý databázový soubor (nevyužívá se).
- Tabulka je uzamčena – tj. tato úroveň se používá, pokud provádíte změny celé tabulky.
- Blok nebo stránka je zamčena – blok je nejmenší jednotka dat, kterou může operační systém načíst ze souboru nebo ji do souboru uložit.
- Řádek je uzamčen – tato úroveň zamykání se používá nejčastěji a podporují jí téměř všechny moderní databázové systémy.
- Sloupec je uzamčen – tato metoda není příliš praktická vzhledem ke spotřebě prostředků, které je potřeba k tomuto úkonu.

Uváznutí

Pojem uváznutí označují situaci, kdy dvě nebo více databázových relací uzamknou určitá data a každá z těchto relací poté požádá o zámek, dat, která jsou již uzamčena jinou relací.

10. Integrace jazyka SQL do aplikace

Nyní se podíváme, jak se jazyk SQL používá v aplikacích.

Zpracování kurzorů

Procedurální programovací jazyk jsou navrženy tak, aby postupně zpracovávaly jednotlivé záznamy. Při použití jazyka SQL spolu s těmito programovacími jazyky nastávají potíže, protože dotazy SQL obvykle poskytují sady výsledků, které zahrnují více datových záznamů.

Z tohoto důvodu relační databáze podporují koncepci kurzoru, což není nic jiného než ukazatel na jediný řádek v sadě výsledků. Seznamy jednotlivých příkazů:

- Příkaz DECLARE CURSOR – je nutné nejdříve deklarovat, aby bylo možné se na něj odkazovat z jiných příkazů SQL.
- Příkaz OPEN CURSOR – kurzor je možné použít teprve poté, co byl otevřen.
- Příkaz FETCH – pokaždé když program požádá o nový řádek ze sady výsledků, odešle kurzor příkaz FETCH.
- Příkazy kurzoru UPDATE a DELETE – SQL umožnuje data vybraná kurzorem snadno aktualizovat, případě odstraňovat řádky z tabulky, na které kurzor odkazuje.
- Příkaz CLOSE – příkaz CLOSE uzavírá kurzor.

Integrace jazyka SQL do aplikací

Většina připojení mezi aplikacemi a databázemi je založena na standardním rozhraní API. Rozhraní API je sada konvencí volání, pomocí nichž aplikace přistupuje ke službám. Tyto služby mohou poskytovat operační systémy ale i jiné softwarové aplikace.

Připojení ODBC

ODBC je standardní rozhraní API pro připojení aplikací k systému SŘBD. Rozhraní ODBC, vychází z rozhraní CLI, jehož definici vytvořili v září 1992. Rozhraní ODBS nezávisí na konkrétním jazyku, operačním systému ani databázovém systému. Aplikaci napsanou v souladu s rozhraním ODBC API lze přenést na jinou databázi nebo operační systém. Ovládač ODBC odpovídá za vazbu rozhraní API na konkrétní databázi a platformu. Informace, které příslušná aplikace potřebuje pro připojení k databázové službě, jsou součástí definice označované jako zdroj dat ODBC.

11. Problematika výkonu a ladění SQL

V poslední části se seznámíme s hlediskem výkonu a ladění, abychom dokázaly zvýšit rychlost a efektivitu svých příkazů SQL.

Obecné hlediska ladění relačního systému řízení báze dat (RSŘBD)

Většina výkonnostních problémů RSŘBD je důsledkem špatně napsaných příkazů SQL. Správce databáze však může provést některé kroky, které zlepší efektivitu celého databázového systému. Seznámíme se s několik pravidly:

Minimalizace čtení z disku a zápisu na disk

Tato rada je sice banální, ale nejpomalejší operace ve většině počítačových systémů je vstup a výstup, např. čtení dat z úložného systému a jejich zápis atd. z toho vyplývá, že správce databáze udělá nejlépe, když efektivně využije dostupnou paměť k minimalizaci vstupně-výstupních operací a času, který je nutní k čekání na jejich dokončení zde je několik způsobů jak to urychlit:

- Přiřazení vyrovnávací paměti se správnou velikostí
- Rozložení vstupně-výstupních diskových operací

Ladění počítačového systému a prostředí

Rozumí se, že počítačový systém, ve kterém funguje SŘBD , by měl být co nejrychlejší a nejlépe nakonfigurovaný: toto jsou základní pokyny:

- Výběr rychlého a spolehlivého hardwaru
- Ladění operačního systému

Efektivní návrh tabulek

Návrh relačních tabulek může mít značný dopad na výkon. K většině oblíbených systémů RSŘBD jsou k dispozici příručka nebo dokumenty white paper, které popisují optimální postupy při návrhu efektivních tabulek. Zde jsou nejdůležitější části, na které se zaměřit:

- CHAR vs. VARCHAR – u pěti nebo méně znaků se používá datový typ CHAR. Nad pět znaků se používá VARCHAR.

- Číselné datové typy ve sloupci – použijte nejmenší datový typ, do kterého se data vejdou.
- Shodné datové typy – pro sloupce primárního a cizího klíče.
- Opatrné používání spouštění – spouštění umístění v databázových tabulkách někdy usnadňují práci nebo představují jedinou možnost, jak řešit specifické problémy typy vynucení složitých omezení

Ladění dotazů SQL

Přibližně 80 procent výkonnostních problémů databázových dotazů lze vyřešit úpravou příkazů SQL. Abyste dokázaly SŘBD správně upravit musíte znát, jak daný systém funguje. Umístěním příkazů SQL do uložených procedur můžete například značně zvýšit výkon databází Microsoft SQL a Sybase.

Exekuční plán dotazu popisuje, jakým způsobem bude systém SŘBD zpracovávat konkrétní dotaz. Plán zahrnuje použití indexu, logiku spojení a odhadované náklad na prostředky.

Obecné hlediska RSŘBD

V této části se zaměříme na hlediska návrhu a ladění, které platí pro většinu implementací SQL.

Seznámení s optimalizátorem

Optimalizátor dotazů je softwarová komponenta systému RSŘBD, která analyzuje příkaz SQL a určuje optimální způsob, jak jej provést. Většina moderních optimalizátorů je založena na nákladech. To znamená, odhaduje náklady všech možných způsobů, jak příkaz realizovat, a na základě analýzy zvolí způsob s nejnižšími náklady. Následuje několik hledisek, která se týkají optimalizátorů dotazů:

- Pořadí názvů tabulek
- Pořadí predikátů hledání
- Chybějící statistika
- Předpis dotazů
- Sloučení definic pohledů
- Jiná kritéria

Návrh efektivních dotazů

Mnozí vývojáři aplikací píší příkazy SQL bez přípravy a příliš se nesnaží o to, aby je navrhli s ohledem na efektivní zpracování. Do této pasti snadno spadnete, protože jazyk SQL je neprocedurální a poskytuje falešný dojem, že pokud příkaz poskytuje správnou sadu výsledků, nezáleží na jeho tvaru. Několik hledisek, která se týkají návrhu databází:

- Seznámení s daty
- Minimalizace počtu vrácených řádků
- Vyloučení skenování velkých tabulek
- Vyloučení zbytečných sloupců
- Vyloučení řazení velkých sad výsledků
- Sladění datových typů v predikátech

Rozumné použití indexů

Indexy mohou značně urychlit přístup k datům. Vždy je třeba si pamatovat, že indexy zabírají místo a vyžadují údržbu. Zde je několik hledisek pro zvýšení výkonu pomocí indexu:

- Vyloučení indexů pro často aktualizované sloupce
- Vytváření pouze selektivních indexů
- Indexy cizího klíče zlepšují výkon spojení
- Indexování sloupců, které se často používají v predikátech
- Nadměrné indexování není žádoucí
- Vyloučení překrývajících se indexů
- Možnost jedinečných indexů
- Odstranění indexů pro nárazovém zatížení

Ladění příkazů DML

Příkazy jazyka DML zpravidla přenášejí méně výkonnostních problémů než dotazovací příkazy.

U příkazu INSERT je nutné mít na paměti dvě hlediska:

- Zajištění dostatku volného místa pro nové řádky
- Údržba indexu

S příkazy UPDATE souvisejí následující hlediska

- Údržba indexu
- Rozšíření řádků

Příkazy DELETE způsobují výkonnostní problémy nejméně často. Pokud se však nadřízená tabulka účastní relace, která je definována s možností ON DELETE CASCADE, může poklesnout její výkon, jestliže je nutné z podřízených tabulek odstranit hodně řádků.

12. Literatura

SQL bez předchozích znalostí průvodce pro samouky, Cpress 2005, ISBN 978-80-251-1707-1