Mobilní databázové systémy

Petr Voborník
UHK – FIM – dTDDS
petr.vobornik@uhk.cz

Abstrakt

Trendem moderní doby je mimo jiné miniaturizace, mobilita a konektivita. Všechny tyto aspekty jsou nakloněny rozvoji mobilních databází, jež se stávají stále nezbytnějším vybavením většiny terénních pracovníků. Online přístup k datům společnosti s možností je přímo aktualizovat zlepšuje produktivitu, šetří čas i peníze a snižuje chybovost těchto dat.

V tomto článku se tedy zaměříme na jednu z hlavních součástí umožňujících tento komfort – mobilní databázové systémy (DBMS). Nejprve bude nastíněna teorie jejich fungování, následovat bude přehled a srovnání současných komerčních řešení a na závěr budou přiblíženy dvě teorie, které by v budoucnu mohly být aplikovány v praxi.

Klíčová slova

Mobilní databázové systémy, distribuované databáze, modely odpojování mobilních databází, DBMS, offline řešení, Oracle lite.

Úvod

Jedním z trendů poslední doby se stávají takzvané mobilní kanceláře, umožňující být schopen odvádět práci stejně rychle, kvalitně a pohodlně, nejenom ze své skutečné kanceláře, ale v podstatě odkudkoli, kde se dotyčný bude zrovna nacházet (od klienta, z domova, ze služební cesty, ze zahraničí, z dovolené, ze zahrady, z automobilu, z letadla, z čekárny u lékaře, ...). Důvodem samozřejmě není pouze pohodlnost, ale schopnost lepšího využití času, větší mobilita a vyšší stupeň pohotovostní způsobilosti.

Další skupinou pak jsou zaměstnanci, jimž potřeba práce v terénu přímo vyplývá z její podstaty (například pošťáci, prodejci, pojišťovací agenti a likvidátoři, inspektoři, auditoři, servisní technici, lékaři, pracovníci realitních kanceláří, policisté, vojáci, ...).

Rozvoj mobilních technologií v posledních letech dosahuje stále lepších a lepších výsledků, a je tedy tomuto trendu příznivě nakloněn. Zařízení jako jsou notebooky, PDA[1], MDA[2], mobilní telefony a další přenosná zařízení s přístupem na internet se stávají stále nezbytnějším vybavením většiny terénních pracovníků.

Aby byla mobilní zařízení jejich uživatelům co platná, dochází taktéž k rozšiřování data přenášejících služeb. Nejedná se však pouze o připojení se k internetu přes operátora mobilních telefonů, ale také o satelitní spojení, wireless, GPRS, G4 a další. Ideálním cílem by pak byl stav, kdy by bylo možné odkudkoli navázat datové spojení s kýmkoli, co nejvyšší rychlostí a za co nejnižší náklady.

Spojení, která tato zařízení a technologie umožňují pak mohou sloužit k různým účelům (telefonování, chatování, videokonferencím, hledání informací na internetu, stahování či nahrávání dat, navigaci, atd.). Co je však nejdůležitější z pohledu „mobilních zaměstnanců" je možnost bezpečného a stabilního připojení do podnikové sítě a především k firemní databázi. K tomuto je krom hardwarových prostředků ovšem nezbytná i softwarová část. Z té se zde přímo zaměříme na databázové

[1] PDA – Personal Digital Assistant – malý kapesní počítač, ovládaný obvykle dotykovou obrazovkou a perem [10].
[2] MDA – Mobile Digital Assistant – mobilní digitální asistent.

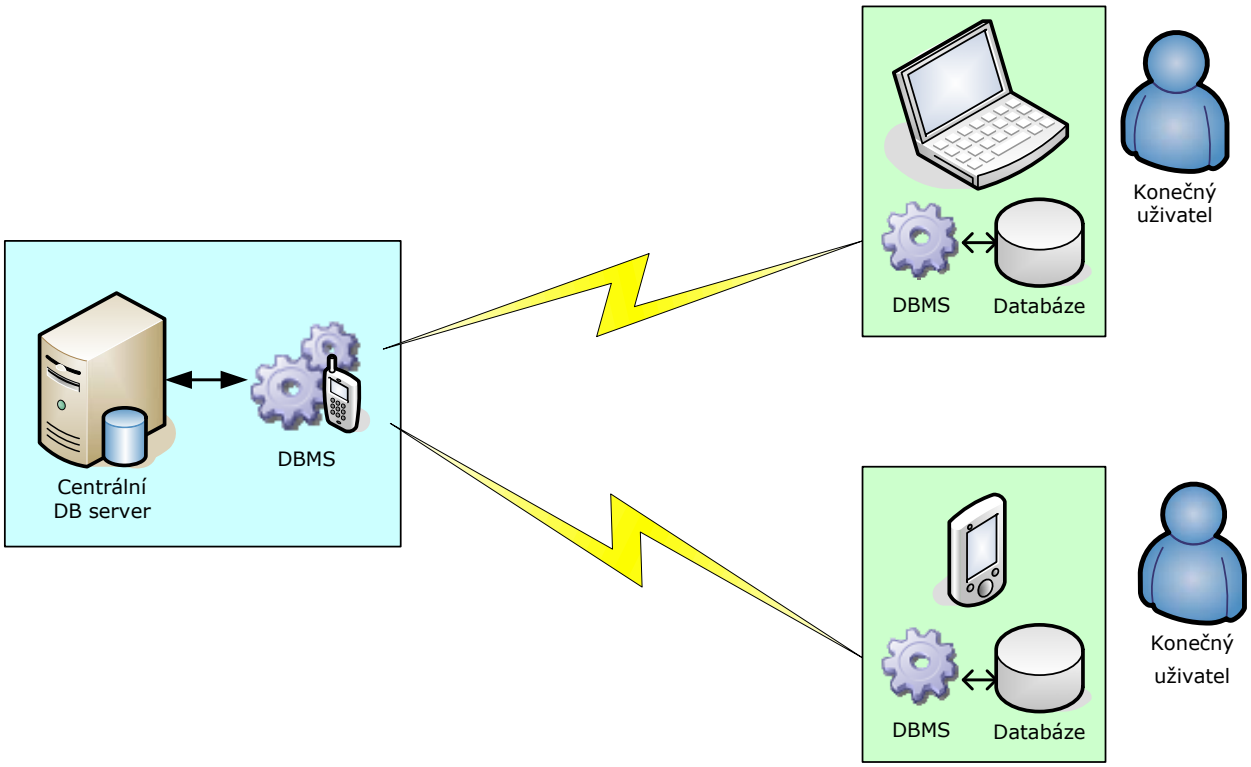
systémy, umožňující přístup mobilních zařízení k datům (DBMS).

Teorie DBMS

Mobilní databáze je přenosná a fyzicky oddělená od společného databázového serveru a je schopna komunikovat s tímto serverem ze vzdálených míst umožňujících sdílet společná data. [1]

Tím je rozuměn HTML případně WML[3] prohlížeč, nebo tzv. terminálové služby, které přenášejí pouze „obrazovky" mezi mobilním zařízením a centrální části. [3]

V softwarových vrstvách (viz Obr. 2) pak vrstva DBMS zcela chybí a aplikaci tvoří klasický webový prohlížeč, nebo program pro připojení ke vzdálené ploše.



Obr. 1 – Klasická architektura pro mobilní databázové prostředí [1]

Možnosti mobilního řešení

U mobilních řešení přístupu k databázi existují dvě základní pojetí, jak zajistit tento přístup: *online* a *offline* řešení.

Online řešení

Online řešení je založeno na práci při navázaném aktivním spojení mobilního zařízení s centrálním systémem. Tento princip umožňuje práci bez nutnosti instalovat aplikační software na mobilní zařízení, přičemž aplikace je nasazena na centrální serverové části a mobilní zařízení pak funguje jako terminál resp. přístupový kanál.

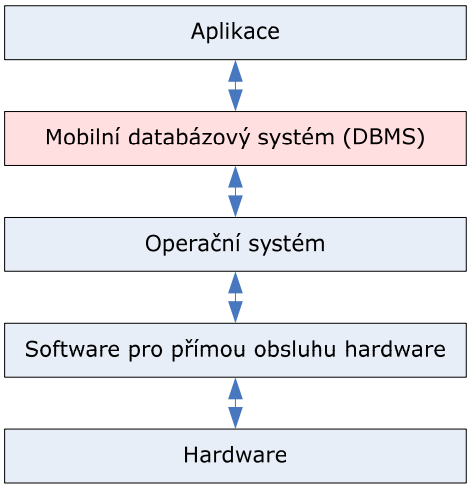
Offline řešení

Druhou možností je pak offline řešení, které nevyžaduje pro práci s aplikacemi trvalé připojení k centrální serverové části systému. Aby toto bylo možné je však nezbytné, aby na mobilním zařízení byla nainstalována jak mobilní aplikace, tak mobilní databázový systém včetně dat, představující „zmenšený obraz" plnohodnotné centrální databáze a aplikace. Mobilní pracovníci pak mohou kdykoli a kdekoli pracovat s tímto systémem bez přímého spo-

[3] WML - Wireless Markup Language – značkovací jazyk založený na jazyce XML umožňující tvorbu online dokumentů pro mobilní zařízení [10].

jení s centrálou a stačí jim se pouze čas od času „připojit" k centrálnímu systému a provést synchronizaci, jenž obnáší především aktualizaci dat na obou stranách, tedy v centrální i v mobilní databázi. [3]

Ač jsou tedy oba přístupy funkční o plnohodnotném mobilním databázovém systému lze hovořit pouze v případě offline řešení, případně kombinaci obou. Online řešení tedy v dalším textu zcela opustíme a budeme se věnovat výhradně offline řešení. Ze softwarových vrstev (viz Obr. 2) se pak zaměříme především na vrstvu DBMS.

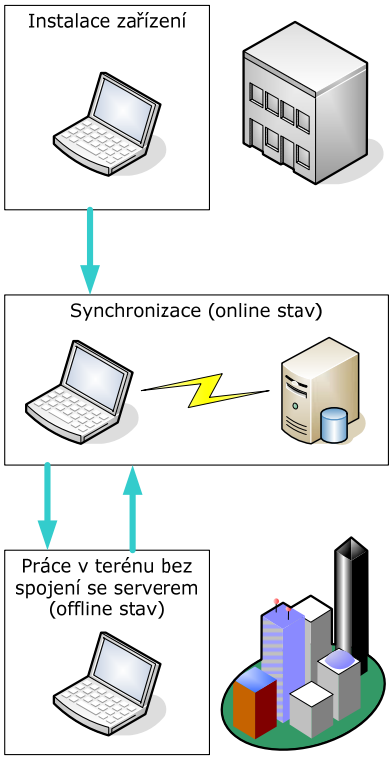


Obr. 2 – Softwarové vrstvy mobilního zařízení

Princip offline řešení

Offline řešení pracuje na bázi replikace dat. Nereplikuje se však celá databáze, ale pouze její pečlivě zvolená, co nejmenší část, kterou daný uživatel potřebuje pro svou práci v terénu.

Kromě části databáze (zvolených entit a z nich příslušných dat) dochází v některých případech k automatickému nahrávání aplikace pracující s daty na mobilní zařízení. Ta se samozřejmě také nepřenáší na zařízení celá, ale dovoluje-li to její architektura, jsou i z ní vybrány pouze části potřebné pro práci konkrétního uživatele. Jedná se tedy o jakousi aktualizaci aplikace, aby každý z uživatelů měl vždy poslední verzi, bez toho, aby se o to musel on sám starat.



Obr. 3 – Princip offline řešení

Celý postup práce s databází na mobilním zařízením je tedy následující: Nejprve je na něj nainstalována mobilní verze databázového systému a aplikace pro obsluhu dat. Poté je provedena synchronizace. Do mobilní databáze se nahrají vybraná data, určená jako relevantní pro práci daného uživatele. Pracovník poté odvádí svou práci v terénu a postupně mění a přidává data do databáze na svém mobilním zařízení. Poté se tímto zařízení připojí k centrální databázi, přičemž je celkem jedno jak a odkud to provede, jestli přímo v centrále firmy, na její pobočce, či z domu přes internetu. Databázový systém provede synchronizaci, tedy uloží změny, které pracovník provedl s daty na svém mobilním zařízení do centrální databáze a jemu zase aktualizuje data, která byla mezitím změněna někým jiným. Konfliktními situacemi (data změnil on a zároveň někdo jiný) se budeme zabývat dále (viz Konflikty, str. 4). Pokud to databázový systém podporuje, dojde zároveň k aktualizaci aplikace na uživatelově mobilním zařízení, byla-li na server nahrána nová verze. Proces synchronizace pak probíhá podle potřeb, možností a vnitřních pravidel organizace. Její frekvence se mů-

že pohybovat v řádu minut až dnů, avšak většinou jsou upřednostňovány co nejčastější intervaly.

Verze DBMS pro mobilní zařízení

Verze databázového systému pro mobilní zařízení musí splňovat několik podmínek oproti standardnímu databázovému systému nasazenému na serveru. Především je tu požadavek na co nejmenší nároky na prostředky systému, na kterém běží. Tím je myšleno zabrání co nejmenšího prostoru na „pevném" disku zařízení a to jak pro samotný DBSM, tak pro spravované databáze. Dále pak co nejnižší požadavky na operační paměť a výkon procesoru.

Oproti tomu stojí naopak požadavek, aby tyto DBMS poskytovaly co možná největší komfort a funkce, jako plnohodnotné databázové systémy. Základem je tedy správa přístupu k datům a udržování relační databáze v konzistentním stavu. Ne vždy je pak vyžadováno druhotných funkcí jako jsou triggry, uložené procedury apod. Co je ovšem u těchto systémů nezbytnou samozřejmostí, je schopnost synchronizace se svým protějškem na centrálním serveru.

DBMS na centrálním serveru

DBMS na centrálním serveru je nadstavbou klasického databázového systému, většinou ve formě samostatné aplikace. Ta může být dokonce heterogenní, čili databázový systém na serveru nemusí být tentýž (ani od stejného výrobce) jako DBMS který komunikuje s DBMS na mobilních zařízeních.

Hlavní funkcí tohoto systému je tedy především zajišťovat komunikaci s mobilními zařízeními. To obnáší zejména synchronizaci dat (viz Proces synchronizace), čili jejich získání z databázového serveru, odeslání na mobilní zařízení a zároveň i jejich přijetí z těchto zařízení a uložení (aktualizaci) na server. Jak již bylo řečeno, někdy krom dat zajišťuje DBMS i aktualizaci aplikace, DBMS na zařízeních, případně i další služby.

Proces synchronizace

Proces synchronizace zajišťuje tedy sladění dat v centrální databázi a na mobilním zařízení. Je třeba aby počítal s extrémními podmínkami za kterých může probíhat, což je například krátké, pomalé či nestabilní spojení pomocí veřejné sítě.

Vzhledem k možné pomalosti spojení a požadavku na co nejrychlejší průběh synchronizace je třeba co nejvíce redukovat objem přenášených dat. Toho lze docílit omezením se na skutečně měněná data, nikoli tedy na přenos celé databáze (tzv. kompletní replikaci). K tomu pomáhají příznaky změny dat, časová razítka a centrální primární klíče. Jazyk SQL pak umožňuje s minimálními přenosovými náklady případné požadavky na výběr dalších dat.

Tato opatření lze kombinovat ještě s kompresí přenášených dat. Ty tedy odesílající DBMS zkomprimuje (případně ještě zašifruje), odešle a přijímající DBMS data opět dekomprimuje (či nejdříve dešifruje) a poté s nimi teprve pracuje dále.

Zabezpečení dat

Pokud je spojení mobilního zařízení a centrální databáze během synchronizace realizováno přes veřejnou síť internet, je samozřejmě nezbytné přenášená data nějakým způsobem zabezpečit proti odposlechu. Samozřejmostí je autentizace a autorizace, samotný přenos je pak realizován buď pomocí vlastního kryptovacího nástroje v rámci DBSM nebo přes zabezpečený protokol (např. SSL či VPN), případně je použita kombinace obojího.

Další možností zabezpečení je zajištění dat na samotném mobilním zařízení pro případ jeho ztráty či odcizení. Jelikož je totiž mobilní, není tato možnost nikdy vyloučena. Nejčastějším způsobem této ochrany je šifrování dat uložených na mobilním zařízení. Jde buď o šifrování přímo dat ukládaných do databáze nebo šifrování databázového souboru v rámci operačního systému. Pro přečtení dat je pak třeba autentizace nejčastěji pomocí přístupového hesla.

Kromě zabezpečení přenášených dat proti odposlechu je třeba tato data ještě zabezpečit proti jejich možnému narušení během tohoto přenosu. Tuto funkci zajišťují obvykle kontrolní součty přenášených dat, aby v případě přerušení či narušení spojení nedošlo k jejich chybné interpretaci na cílovém DBMS. O toto se sice starají přímo komunikační protokoly (TCP), nicméně je obvyklá i kontrola dat na bázi DBMS.

Konflikty

Jak již bylo řečeno, konflikty v DBMS vznikají při synchronizaci, pokud dva uživatelé změnili stejná data (shodný řádek téže tabulky). Tomu, který poté provede synchronizaci jako první, proběhne vše bez problémů, ovšem tomu druhému již nikoli, protože by svým updatem přepsal data zapsaná tím prvním.

Existuje řada postupů, jak se s konflikty vyrovnávat. U mobilních zařízení je však celá záležitost o mnoho složitější. Nejde zde totiž aplikovat klasické postupy uzamykání záznamů, protože by prostě k jejich uzamčení docházelo z pohledu databázového systému na nepřiměřeně dlouhou dobu a práce s databází by pak byla téměř nemožná.

Nejjednodušší možností řešení či spíše neřešení konfliktů, kterou bohužel některé systémy skutečně používají je, konflikty zcela ignorovat a uložit prostě vždy poslední data jako ta platná.

Lepším přístupem, jež ovšem neeliminuje všechny případy, je uchovávání více verzí všech atributů každého synchronizovaného řádku a jejich následná individuální kontrola, došlo-li v nich skutečně ke změně. To může například vyřešit případ, že jeden pracovník v tabulce zákazníků aktualizuje jeho rodinný stav a druhý jeho příjmení. Ač tedy došlo ke dvěma změnám stejného řádku shodné tabulky, jedná se o nekonfliktní update. Toto řešení však má nemalé paměťové požadavky na straně serveru. Krom toho v případě že by oba (či dokonce i více) uživatelé změnilo stejný atribut téhož řádku shodné tabulky na rozdílnou hodnotu, bude nezbytné tyto případy řešit individuálně.

Ještě lepším přístupem, bohužel ne vždy zcela aplikovatelným, je rozdělení dat uživatelům tak, aby každý měl pouze ta potřebná pro svou práci, která by již nikdo další neměl, případně k nim měl přístup pouze pro čtení. Například by tedy každý z pracovníků měl v databázi pouze ty zákazníky, které má na starost a nikoliv už ty, jež zajišťuje jeho kolega. Kompletní tabulku například s číselníkem pohlaví by pak měli samozřejmě oba, ovšem nemohli v ní provádět žádné změny.

Současné DBMS

Nyní se podíváme na existující databázové systémy pro mobilní zařízení několika největších firem a jaké možnosti jejich produkty nabízejí. Zaměříme se především na produkt firmy Oracle, která je v oblasti databází jedničkou na trhu a pak v krátkosti zmíníme i produkty dalších firem.

Oracle lite

Oracle Database Lite je samostatný produkt (doplněk klasické Oracle databáze), který se používá pro vývoj aplikací pro offline mobilní zařízení a zařízení s limitovaným výpočetním výkonem. Lite systém není pouze samotná databázová platforma, která běží na koncovém zařízení, ale zahrnuje i synchronizační systém, který sídlí na serveru a umožňuje udržovat distribuované databáze na mobilních zařízeních v konzistentním stavu s produkční podnikovou databází. Mobilní pracovníci pak mohou pracovat s vymezenou množinou dat i bez stálého připojení k podnikové databázi.

Celý Oracle Lite systém lze rozdělit na dvě hlavní části – serverovou a klientskou. Serverová část se označuje jako *Mobile Server* a zajišťuje synchronizaci dat mezi databázovým serverem a mobilními klienty, správu uživatelů a jejich práv a správu klientských zařízení. Na mobilních zařízeních je pak několik malých aplikací, které

slouží pro řízení synchronizace se serverem, provádění úloh, které souvisí se vzdálenou správou zařízení (aktualizace aplikací apod.) a jednoduchý SQL klient. Dále je zde pak samozřejmě umístěn datový soubor, který obsahuje samotná aplikační data. [5]

Oracle Lite na mobilním zařízení

Oracle Lite podporuje v současnosti koncová zařízení s operačními systémy typu Windows32, WindowsCE, PalmOS, Linux a Symbian. Databázový systém je nenáročný na systémové prostředky, takže jej lze nasadit i na výkonnostně slabší zařízení. Systému má velice snadnou instalaci, minimální potřebu administrace a velké množství podporovaných funkcí ve srovnání s klasickou Oracle platformou.

Pro přístup k datům se používá standardní Oracle dialekt SQL s několika drobnými omezeními oproti plnohodnotné platformě. Mezi podporovaná datová rozhraní patří ODBC, JDBC, SODA (Stateless Object Database Access – knihovna pro přístup k datům z C++), ADOCE a ADO.Net.

Vývoj aplikací je tedy díky mnoha podporovaným rozhraním relativně nenáročný. Aplikační logiku je na mobilních zařízeních však vhodné umístit přímo do kódu aplikace místo do uložených procedur.

Bezpečnost dat na zařízení je možné zajistit šifrováním datového souboru pomocí hesla. Data jsou poté přístupná pouze po zadání tohoto hesla a nebezpečí úniku dat při ztrátě či odcizení zařízení je tedy minimální. [5]

Oracle Mobile Server

Na straně serveru běží *Oracle Mobile Server*, což je java aplikace, poskytující přístup k administraci celého Oracle Lite systému přes webové rozhraní a stará se o zpracování dat při synchronizaci a přípravu nových nebo změněných dat produkční databáze pro mobilní zařízení. Dále také umožňuje spravovat mobilní uživatele, jejich zařízení a jejich přístup k distribuovaným datům a aplikacím.

Mobile server může běžet samostatně nebo jako součást Oracle Application Server.

Kromě samotného webového rozhraní, které umožňuje provádět některé základní operace, existuje ještě několik dalších prostředků, kterými lze definovat synchronizaci dat a aplikací.

Součástí mobile serveru je plánovač úloh, který v pravidelných intervalech spouští takzvaný MGP (Message Generator and Processor), což je nástroj, který připravuje změněná data pro jednotlivé uživatele mobilních zařízení a zpracovává data, která byla na server z klientských zařízení přenesena. Plánovač samozřejmě může spouštět i další naplánované úlohy.

Bezpečnost přenosu dat mezi klienty a serverem je možné zajistit pomocí SSL protokolu a samotná data je možné synchronizovat až po zadání platného uživatelského jména a hesla. V případě odcizení zařízení je také možné deaktivovat přístup k synchronizaci dat a případně i přidat do fronty příkaz pro vymazání databáze a odinstalování celého Oracle Lite systému – při pokusu o synchronizaci pak dojde automaticky k odstranění Oracle Lite včetně dat. [5]

Synchronizace

Základní koncept Oracle Lite je postaven na aplikaci. Ta definuje synchronizovaná data a aplikační soubory a to vždy výhradně pro jednu platformu.

Chceme-li distribuovat jednu množinu dat na různé typy zařízení, která nemají shodnou platformu, je třeba vytvořit pro každou z nich jednu aplikaci. Základní myšlenkou je spojení dat s aplikací a distribuce tohoto jednoho celku.

Datová část je založená na takzvaných snapshots – zjednodušeně řečeno: každá přenášená databázová tabulka rovná se jeden snapshot, který se definuje pomocí klasického SQL dotazu.

U každého snapshotu lze definovat, jestli se má synchronizace provádět vždy úplná nebo přenést pouze změny. Definice přenášených tabulek má pak za následek vytvoření sady triggerů a pomocných tabulek na databázovém serveru, které evidují změny jednotlivých řádků.

Kromě synchronizace samotných datových tabulek podporuje Oracle Lite také spouštění SQL scriptů na koncových zařízeních a definici sekvencí. V praxi je potom nezbytné definovat sekvenci pro potřeby Oracle Lite tak, aby nekolidovala s již existujícími sekvencemi v produkční databázi. Dalším prvkem, který lze definovat, jsou indexy nad tabulkami.

Aplikační soubory se synchronizují na základě instalačních scriptů, které umožňují přesně specifikovat, kam se mají soubory na klientském zařízení umístit, jaké programy se mají při instalaci spustit atd. Aktualizace aplikací pak probíhá opět pomocí těchto scriptů a lze stejným způsobem poměrně detailně ošetřit. [5]

Sybase SQL Anywhere

Sybase SQL Anywhere podporuje kromě jiného i zařízení postavená na operační systémy typu Windows Mobile nebo PalmOS. Balík SQL Anywhere obsahuje dva produkty, které lze použít na zařízeních typu PDA – Adaptive Server Anywhere a UltraLite.

První jmenovaný produkt poskytuje řadu funkcí, které známe z velkých databázových serverů (triggery, uložené procedury v SQL nebo Javě, zatímco druhý UltraLite je produkt s minimálními hardwarovými nároky (velikost je přibližně 50 KB) a poskytuje pouze základní databázovou funkcionalitu. K databázím lze pak přistupovat mnoha způsoby z různých prostředí, takže vývojář není z hlediska vývojového prostředí nijak omezován.

Synchronizace je postavena na technologii MobiLink, která umožňuje synchronizaci s databázovými servery Adaptive Server Anywhere, Sybase Adaptive Server Enterprise, Oracle Dataabse, Microsoft SQL Server a IBM DB2. Samotná synchronizace klientů pak probíhá přes standardní protokoly HTTP nebo HTTPS (SSL), případně přímo přes Palm HotSync nebo MS ActiveSync.

Bezpečnost dat je u tohoto produktu zajištěna jak během komunikace zařízení se serverem (šifrovaný přenos dat a autentizace), tak i na samotném zařízení, kde jsou data šifrována a pro přístup k nim je nutná autentizace uživatele.

Výhodou tohoto produktu je poměrně široká podpora klientských zařízení, databázových systémů na straně serveru a v neposlední řadě také podpora širokého spektra vývojových prostředí a nástrojů. Výhodou je také možnost implementace vlastní synchronizační logiky buď přímo pomocí SQL nebo v prostředí Javy či .Net. Nevýhodou je absence synchronizačních mechanismů pro klientské aplikace, což je nutné řešit externími nástroji. [5], [9]

Microsoft – MS SQL CE

Microsoft SQL Server CE je databázový stroj, který je určen pro zařízení založená na operačních systémech Microsoft Windows CE, Microsoft Windows XP Tablet PC Edition, Windows Mobile 2003 Software for Pocket PC a Windows Mobile 5.0.

Výhodou tohoto řešení je velká podpora z hlediska vývoje (přímá podpora v MS Visual Studio a .Net Compact Framework) i provázání s operačním systémem mobilních zařízení na MS platformě. Velikou výhodou oproti konkurenci je možnost pracovat jak s offline zařízeními při občasné synchronizaci s podnikovou databází, tak zároveň i se zařízeními přímo připojenými, kdy se datové operace provádí místo nad lokální kopii dat přímo nad podnikovou databází. Přístup je z hlediska aplikace transparentní.

Bezpečnost dat při přenosu mezi serverem a klientskými zařízeními je zajištěna prostředky IIS serveru – je tedy možné využít protokol SSL s nutností zadání uživa-

telského jména a hesla a případně i autentizaci uživatele v rámci domény Windows.

Jednou z hlavních nevýhod je omezení pouze na zařízení s operačním systémem z dílny Microsoftu, propojení výhradně s velkou MS SQL databází a pak také fakt, že toto řešení nepodporuje distribuci aplikačních souborů na mobilní zařízení. [5]

IBM – DB2 Everyplace Database

Společnost IBM nabízí svým klientům na mobilní zařízení DB2 Everyplace Database, která se skládá ze tří základních komponent — vlastní databázové jádro (handheld database engine), synchronizační server (SyncServer) a Personal Application Builder (PAB).

Databázové jádro je klasická relační databáze zajišťující ukládání a poskytování dat. Samozřejmostí je integritní mechanismus bránící ztrátě konzistentnosti dat. Jádro zabírá na mobilním zařízení přibližně 100–150 KB a dovoluje připojení více aplikací současně.

Synchronizaci dat s centrální databází, která nemusí být nutně DB2, zajišťuje SyncServer. To znamená, že data, která se mají synchronizovat, procházejí právě přes SyncServer. Synchronizace dat spočívá buďto v přenesení změn z mobilní databáze na hlavní server nebo naopak změny, ke kterým došlo v centrální databázi, se přenesou na handheld[4].

Personal Application Builder slouží k vytváření vizuálních databázových aplikací pro DB2 Everyplace na mobilní zařízení. Lze jej také začlenit do testovacích a ladících nástrojů. Například testování je možné provádět díky nástroji PalmOS Emulator bez fyzického připojení k handheldu. [9]

Extended Systems OneBridge

Tento produkt je zaměřen spíše všeobecně a podporuje jednak přenos a řízení distribuce aplikací a dalších souborů a kromě jiného také synchronizaci databází, která je založena na rozšiřujících pluginech (conduits), jež umožňují synchronizovat mobilní zařízení prakticky s jakýmkoliv databázovým zdrojem.

Mezi podporované operační systémy a datové zdroje patří PalmOS (PDB, DB2e databáze), Windows Mobile (databáze dostupné přes ADOCE, ObjectStore = nativní WM databáze, DB2e), Symbian a desktopové Win32 operační systémy (databáze podporující ADO).

Bezpečnost tohoto řešení na straně klienta závisí zejména na konkrétním použitém úložišti dat, většina z nich však umožňuje šifrovat data pomocí hesla. Další možností je využití bezpečnostních technologií iAnywhere, které umožňují šifrovat kompletní obsah paměti klientského zařízení a po každém startu zařízení vyžadují heslo pro „odemknutí" obsahu paměti.

Komunikace mezi OneBridge serverem a klienty je zabezpečena velice důsledně za pomoci kombinace asymetrické a symetrické šifry (RSA a AES) a je možné vyžadovat zadání uživatelského jména a hesla při každé synchronizaci.

Výhodou tohoto řešení je tedy podpora více operačních systémů, podpora synchronizace aplikací, správa mobilních zařízení a velká univerzálnost celého řešení, která umožňuje synchronizovat data prakticky z jakéhokoliv zdroje. Také zabezpečení celého systému je na vysoké úrovni. Díky dostupnosti ODBMS pro platformu PocketPC(db4o) je dokonce možné na rozdíl od ostatních systémů implementovat vše jak na straně serveru, tak i na straně klienta zcela na základě objektových principů. Nevýhodou je větší náročnost při implementaci. [5]

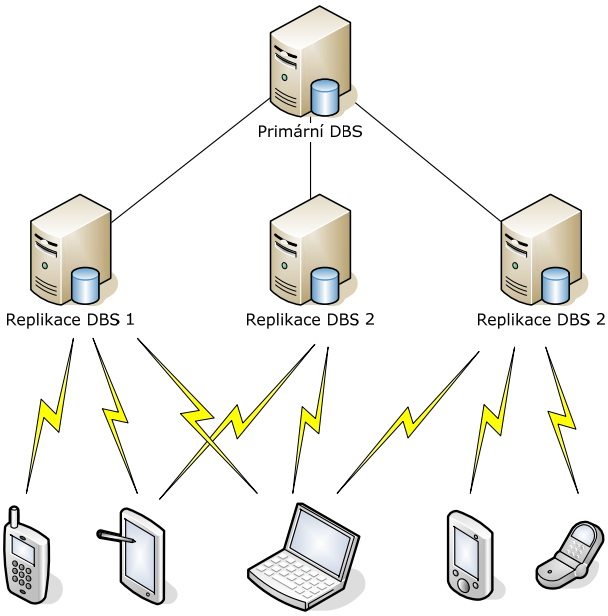
[4] Handheld - kapesní počítač s fyzickou klávesnicí.

Nové teorie v DBMS

V této části si přiblížíme a zhodnotíme dvě teorie, které teprve aspirují na svou realizaci.

Konzistentnost u více přístupových replikovaných míst

Jedná se o návrh řešení pro distribuované databázové prostředí, kde kromě primárního serveru existují i jeho kompletní replikované kopie, k nimž se dle potřeby připojují mobilní zařízení a provádí synchronizaci dat. Není tedy vyloučeno, že určité mobilní zařízení při první synchronizace načte data z jednoho replikovaného serveru a při druhé synchronizaci aktualizuje tato data na serveru zcela jiném (viz Obr. 4).



Obr. 4 – Přístupy mobilních zařízení k různým replikacím jedné databáze

Tato architektura může být v některých aspektech, výhodná či dokonce nezbytná, nicméně sebou přináší samozřejmě i celou řadu komplikací. Jedno z možných řešení navrhli José Monteiro, Ângelo Brayner a Sérgio Lifschitz ve svém článku *„A Mechanism for Replicated Data Consistency in Mobile Computing Environments*" [6], na kteréžto se podíváme podrobněji v následujícím textu.

Podmínky

Jednotlivé replikované databázové servery nemusí být vždy dosažitelné jak pro mobilní zařízením, tak pro spojení s primárním serverem. Připojení mobilních zařízení pak může být také nestabilní a během synchronizace není vyloučeno, že dojde k jeho přerušení. Přitom se však počítá s tím, že je-li replikovaný server dostupný pro mobilní zařízení, existuje zároveň jeho stabilní spojení s primárním serverem.

Dočasně může nastat situace, že se na různých replikovaných kopiích nacházejí rozdílná data. Server určený jako primární je pak odpovědný za synchronizaci těchto dat na všech replikovaných databázích. [6]

Zajištění konzistentnosti replikovaných dat

Popisovaný přístup pro zajištění konzistentnosti dat je založen na jednoduché strategii využívající obvyklý synchronizační graf. Jeho dynamickým monitorováním a řízením je poté zajišťováno, aby v něm nevznikala zacyklení (kružnice). Navíc tento graf oproti klasickému využívá informaci o čase ve kterém byla příslušná operace provedena. Kontrolní funkce pak svou částí probíhá na všech úrovních (na mobilních zařízeních, replikovaných serverech i primárním serveru).

Jakmile je někde úspěšně zakončena a potvrzena transakce (commit), primární server rozešle data aktualizovaná touto transakcí na ostatní servery, které jsou zrovna aktivní, společně s odpovídající časovou značkou (verzí). Číslo verze je definováno podle následujících pravidel:

Každá datová jednotka x v každé replice databáze má přiděleno číslo své verze (určitá obdoba časové značky), označme ji třeba $C(x)$. Každá operace (čtení či zápis) transakce T_i má také přiřazeno unikátní číslo verze $P_i(x)$.

Číslo verze se skládá ze dvou hodnot (z,y). První z nich (z) je číslem plnohodnotné verze, druhé (y) pak dočasným čís-

lem podverze. Ve výchozím stavu jsou obě tyto hodnoty rovny nule, na všech serverech.

Číslo verze (z) je automaticky navýšeno pokaždé, když je úspěšně zakončena transakce (commit), jenž provedla alespoň jednu změnu datové jednotky (write). Po každém commitu transakce primární server rozešle nové číslo verze datové jednotky na všechny ostatní aktivní servery.

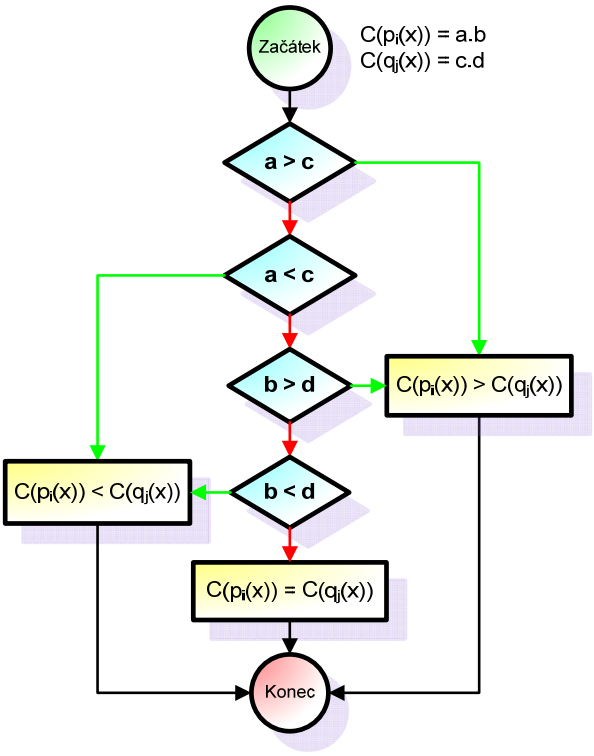
Číslo podverze (y) je zvyšováno při provedení každé operace měnící datovou jednotku x během dosud neukončené transakce na kterémkoli serveru.

Při každém čtení datové jednotky (read) r_i transakcí T_i z repliky R_j je uchován identifikátor těchto dat (ID) a číslo verze, která byla přečtena. Při každé změně této jednotky (write) w_i během transakce T_i je taktéž uchován identifikátor dat, jejich nová hodnota a nové číslo verze zatím na lokální kopii $C(x)_{Rj}$.

V pravidelných intervalech jsou na primární server každým z replikovaných serverů posílány informace obsahující seznam operací provedených na lokální kopii každého serveru spolu s časovými značkami (čísly verzí) přiřazenými k těmto operacím nad jednotlivými daty. Tyto informace jsou přijaty plánovačem na primárním serveru a použity pro synchronizaci operací jednotlivých transakcí určujících správné proložení, což zajistí konzistentnost replikovaných dat.

Pokud klient (mobilní zařízení) provádějící synchronizaci přes některý ze serverů požaduje potvrzení (commit) nebo zrušení (abort) transakce T_i, musí tento server požadavek předat primárnímu serveru. V případě commitu pak musí klient počkat, bude-li jeho požadavek schválen či zamítnut. Při prodlení odpovědi (překročení timeoutu) pak klient může transakci sám zrušit a zkusit ji zopakovat. Pokud se však jedná o transakci, která data pouze četla a neměnila, může být i tak zakončena, ovšem s vědomím, že přečtená data nemusí být konzistentní.

Proces schvalování a provádění operací na primárním serveru je pak založen na porovnávání časových značek (verzí) jednotlivých operací. Postup srovnání dvou značek prezentuje Obr. 5. [6]



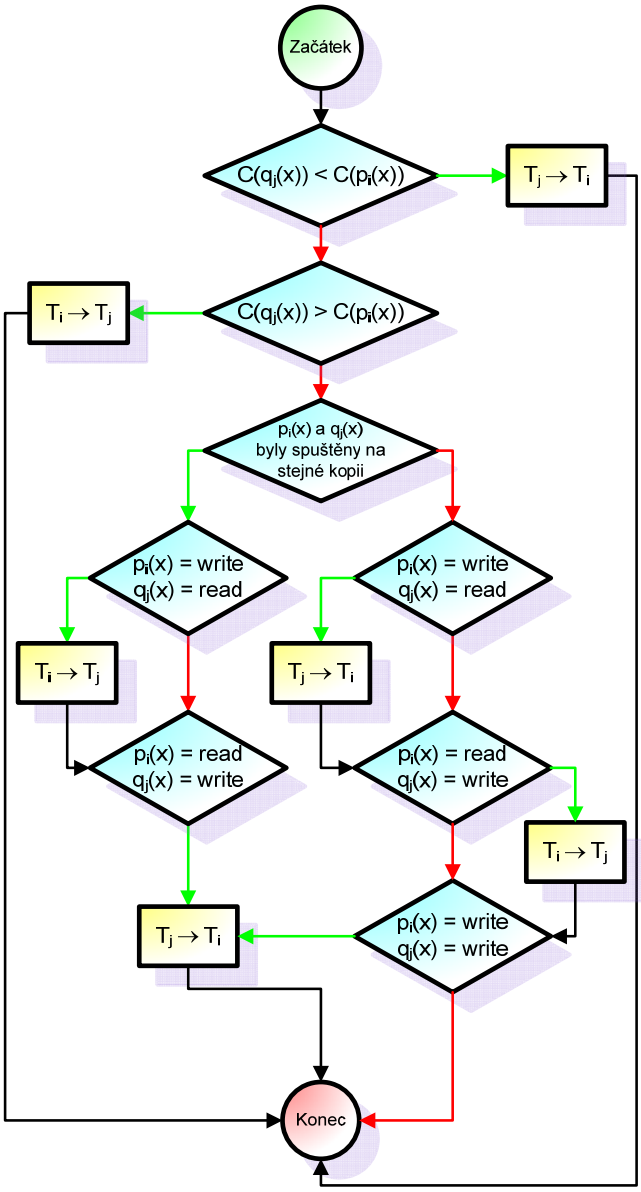
Obr. 5 – Postup porovnání dvou časových značek (verzí) operace

Postup kontroly konzistentnosti

Nyní si popíšeme funkčnost plánovače, umístěného na primárním serveru. Ve výchozím stavu začíná s prázdným pomocným serializačním grafem (TSG[5]), který je generován dle následujících pravidel:

Pro každou operaci $p_i(x)$ z transakce T_i se zjišťuje, existuje-li operace $q_j(x)$ z transakce T_j, která je s touto konfliktní. Pokud je nalezena je do grafu mezi vrcholy T_i a T_j, přidána hrana (hrany) podle postupu zachyceného v Obr. 6.

[5] TSG – Temporal Serialization Graph – pomocný serializační graf. Při jeho postupném vytváření jsou jednotlivé operace serializovány do plánovače (řazeny za sebe dle určitých pravidel). Samotný graf pak má kontrolní funkci, jenž určuje může-li se vůbec daná operace do plánovače přidat či nikoli.



Obr. 6 – Určení orientace hrany grafu (TSG) mezi transakcí T_i a T_j

Po přidání této hrany je zkontrolováno, nevznikla-li díky ní v grafu kružnice. Pokud ano, je zamítnuta operace $p_i(x)$, což má za následek zrušení celé transakce T_i. O tomto výsledku je následně informován klient čekající na rozhodnutí o žádosti na commit této transakce. Pokud však hrana v TSG kružnici netvoří, je operace $p_i(x)$ schválena a zařazena do plánovače.

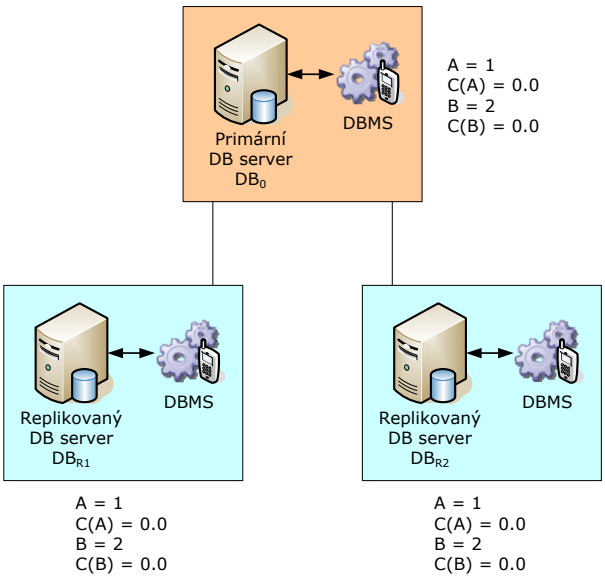
Když plánovač obdrží požadavek na commit transakce T_i, zkontroluje jestli všechny operace této transakce byly již zařazeny do plánovače a jsou-li stále součástí aktivní (nezamítnuté) transakce. Je-li tomu tak, pak je commit schválen a následně i proveden. V opačném případě žádost musí počkat, až plánovač obdrží všechny operace této transakce. Pokud byla transakce již dříve zamítnuta, je tato informace znovu odeslána klientovi.

Po úspěšném commitu transakce plánovač navýší verze u všech datových jednotek změněných touto transakcí o jedničku. Poté primární server nové hodnoty datových jednotek spolu se změněným číslem verze rozešle na všechny aktivní replikované servery. Pokud je některý z nich zrovna neaktivní, je tato informace uložena v logu a bude mu odeslána ihned po jeho připojení.

Pokud klient sám zažádá o abort transakce, je tato odebrána z TSG včetně všech souvisejících hran, operace jsou vyjmuty z plánovače a klientovi je abort schválen. [6]

Příklad

Mějme primární server s databází DB_0 (s datovými jednotkami *A* a *B*), která je kompletně replikovaná na serverech DB_{R1} a DB_{R2} (viz Obr. 7).



Obr. 7 – Schéma replikovaných databází a jejich dat k ukázkovému příkladu

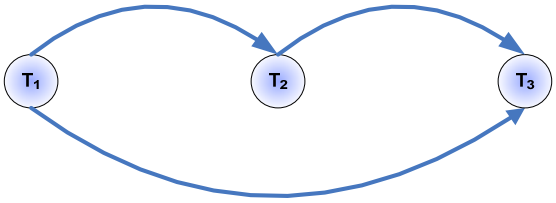
Nyní dojde k následujícímu sledu transakcí a operací v rámci nich na uvedených serverech:

- T_1: $r_1(A)_{R1}$ $r_1(B)_{R1}$ $w_1(A, A+5)_{R1}$
- T_2: $r_2(A)_{R1}$ $w_2(A, A+7)_{R2}$
- T_3: $r_3(B)_{R1}$ $r_3(A)_{R1}$ $w_3(A, A+10)_{R2}$

Tyto operace by poté klasicky byly zapsány v tomto očekávaném pořadí:

$$S_1 = r_1(A)_{R1}\ r_1(B)_{R1}\ w_1(A, A+5)_{R1}\ r_2(A)_{R1}\ w_2(A, A+7)_{R2}\ r_3(B)_{R1}\ r_3(A)_{R1}\ w_3(A, A+10)_{R2}$$

Při zápisu tohoto sledu operací by pak vznikl pomocný serializační graf, vyobrazený na Obr. 8.



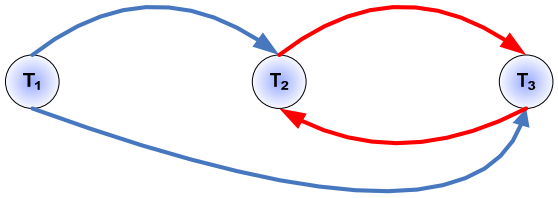
Obr. 8 – Pomocný serializační graf (TSG) závislosti transakcí k plánovači S_1

Jak je z grafu patrné, neobsahuje žádnou kružnici a vše se tedy jeví býti bezproblémovým. Nicméně tak tomu je pouze na první pohled. Zaměříme-li se totiž na konečnou hodnotu datové jednotky *A*, zjistíme, že na serveru DB_{R1} bude rovna 6, zatímco na DB_{R2} a na primárním serveru DB_0 bude její hodnota 16. Jednotlivé operace byly totiž obyčejně poskládány za sebe bez aplikace výše popsaného mechanismu, díky čemuž jsou pak obě tyto hodnoty nekonzistentní.

Hodnota načtená operací $r_3(A)_{R1}$ ze serveru DB_{R1} není shodná s hodnotu zapsanou operací $w_2(A, A+7)_{R2}$ na server DB_{R2}. To proto, že tato načtená hodnota předchází operaci zápisu a je tedy nutné operacím v plánovači pozměnit pořadí. Výsledkem tedy bude tento plán:

$$S`_1 = r_1(A)_{R1}\ r_1(B)_{R1}\ w_1(A, A+5)_{R1}\ r_2(A)_{R1}\ r_3(B)_{R1}\ r_3(A)_{R1}\ w_2(A, A+7)_{R2}\ w_3(A, A+10)_{R2}$$

Tato nekonfliktní serializace je již korektní. Podle již dříve popsaného postupu bude vytvořen nový pomocný serializační graf, jenž je vyobrazen na Obr. 9.



Obr. 9 – Pomocný serializační graf (TSG) závislosti transakcí k plánovači $S`_1$

Tentokrát již ovšem graf kružnici obsahuje. Transakce T_2 totiž čeká na transakci T_3, která zase naopak čeká na transakci T_2. Tyto dvě transakce by tedy na sebe čekaly donekonečna a díky tomu by nemohla skončit ani transakce T_1, čekající na zakončení těchto dvou. Transakce T_3 musí být tedy zrušena, aby ostatní transakce mohly být korektně ukončeny.

Po těchto úpravách bude tedy konečná hodnota datové jednotky *A* na serveru DB_{R1} rovna 6, zatímco na DB_{R2} a na primárním serveru DB_0 bude její hodnota 13, což je zároveň konzistentní hodnota. Ta bude tedy primárním serverem rozšířena na ostatní repliky, v tomto případě tedy na server DB_{R1}. [6]

Správnost protokolu

Představený postup zároveň splňuje dvě základní kritéria správnosti:

- Provádění serializovaných operací konkurenčních transakcí na replikované databázi je ekvivalentní postupnému provádění na stejné nereplikované databázi.
- Všechny kopie postupně konvergují do stejného konzistentního stavu.

Matematické důkazy viz [6].

Závěry

Navržený postup řešení zaručuje udržení replikovaných kopií databáze v konzistentním stavu při různém přístupu mobilních zařízení k těmto serverům. Po teoretické stránce se jedná o velmi zajímavé řešení nicméně z hlediska praktického zde je několik dalších obtíží.

Hlavním problémem je vysoký stupeň centralizace celého řešení a jeho absolutní závislost na primárním serveru. Ten musí být neustále dostupný všem replikovaným kopiím, aby mohly plnit svou funkci a zároveň jim velmi pohotově odpovídat na veškeré požadavky. Dále pak existence správy časových značek (čísel verzí) pro každou datovou jednotku bude mít za následek značné kapacitní nároky u každé kopie databáze.

Model pro dočasná odpojení mobilních databází

Další prezentovanou teorií bude návrh řešení modelu pro odpojování mobilních částí distribuovaných databází. Jejími autory tentokrát jsou Joanne Holliday, Divyakant Agrawal a Amr El Abbadi a prezentovali ji v článku „*Disconnection modes for mobile databases*" [2].

Databázový model

Databázový model, jehož se týká dané řešení je založen na distribuovaném databázovém systému. V něm se nachází konečný počet stanic S_i, přičemž každá z nich obsahuje kompletní replikaci celé databáze. Jedná se o klasické relační databáze, s transakčním přístupem, standardním dvoufázovým uzamykacím protokolem a synchronizačním mechanizmem typu ROWA[6] či quorum[7].

Počet stanic v uvažovaném modelu je pak pevně daný, ovšem může docházet k tomu, že se některé ze stanic na určitou dobu ze společné sítě odpojují a poté zase připojují. Po tuto dobu mohou vyžadovat unikátní přístup k určité časti databáze, kterou mezitím v terénu upravují a následně synchronizují. [2]

[6] ROWA – Read One Write All – synchronizační metoda replikovaných databází. Díky ní lze data přečíst z libovolné repliky databáze, ovšem při jejich změně bude commit schválen až když budou tyto změny zapsány do všech replik databáze.

[7] Quorum – synchronizační metoda replikovaných databází, při níž jsou data zapisována na repliku s nejnižším zatížením a čtena z repliky obsahující jejich nejnovější verzi.

Typy možných odpojení

Neplánovaná odpojení stanice od sítě bez předchozího upozornění nejsou ničím výjimečným a mnohdy bohužel nevyhnutelným (např. při výpadku proudu, ztráty spojení atd.). Prevence může pomoci pouze do určité míry, tudíž nezbývá, nežli s nimi počítat a minimalizovat jejich škody, například transakčním zpracováním.

Tato metoda ovšem vychází z řádného odpojení, při kterém mají stanice určitý čas se mezi sebou informovat o tomto odpojení a zařídit se podle toho s přihlédnutím na další požadavky a okolnosti. V tomto modelu rozlišujeme tyto druhy plánovaného odpojení:

- Při *základním odpojení* stanice S_i, si tato stanice ponechává svou repliku databáze u sebe, ovšem pouze pro čtení. Během svého odpojení ji tedy nemůže nijak měnit. Díky tomu nejsou ani žádné nároky na ostatní stanice.
- V případě odpojení v *kontrolním módu* stanice S_i, požaduje tato stanice kontrolu nad určitou částí databáze, tedy možnost tato data měnit i v offline režimu bez kolize s ostatními stanicemi, které zůstávají připojeny k síti.

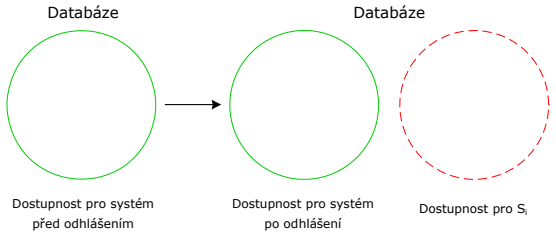
Druhů kontroly nad částí databáze je celá řada. Jednou z nich je například *rozdělení databáze*, kdy oddělenou část zcela spravuje výhradně odpojená stanice S_i a ostatní do ní po tuto dobu nemají vůbec přístup ani pro čtení, stejně tak, jako S_i ke zbytku databáze (viz Obr. 11). Další možností je *kontrola s možností čtení zbytku databáze* odpojenou stanicí S_i, přičemž nad určenou částí mu zůstává výhradní kontrola (viz Obr. 12). Třetí možností je pouze kontrola nad částí databáze, přičemž zbytku *systému* zůstane tato část v původním stavu (před odpojením S_i) alespoň *pro čtení* (viz Obr. 14).

Model pak počítá ještě s možností *volného* a *optimistického* kontrolního módu. První z nich rozděluje kontrolu pro zápis dle přání odpojované stanice a ponechává jí i zbytku systému druhou část databáze pro

čtení (viz Obr. 16). Optimistický mód pak vychází z toho, že by nemělo docházet ke kolizím a ponechává odpojené stanici i zbytku systému plnou kontrolu nad celou databází (viz Obr. 18). [2]

Základní odpojení

Jedná se o jednoduché plánované odpojení, při kterém zůstává odpojené stanici celá databáze k dispozici pouze pro čtení.



Obr. 10 – Základní odpojení stanice S_i. Plná čára ohraničuje část databáze, nad níž má příslušná strana plnou kontrolu (pro čtení i zápis), tečkovaná tu část, která je pouze pro čtení, chybějící část pak tu, k níž vůbec nemá daná strana přístup. [2]

Před samotným odpojením stanice S_i, provede nejprve příslušná opatření. Jako první odpojí veškeré klienty k ní přistupující. Dále se ujistí, že všechny změny dat, které byly provedeny přímo na ni byly replikovány dále na ostatní stanice. Na druhou stranu provede operace uskutečněné na jiných stanicích, tak aby databáze byla v co nejaktuálnější verzi.

Dalším krokem je nalezení zástupce pro dobu své nepřítomnosti. Za něho je zvolena nejlépe nejvýkonnější volná stanice v síti. Veškeré zprávy v síti totiž obcházejí všechny stanice, dokud nenajdou tu správnou, díky čemuž může tento zástupce snadno odpovídat na požadavky kladené na odpojenou stanici, jako by byla k síti stále připojena. Stanice S_i tedy přesměruje svou adresu na tuto.

Funkcí zástupce je tedy jednak odpovídat na požadavky ostatních na odpojenou stanici a za druhé zároveň tyto požadavky uchovávat, aby mohly být předány stanici po jejím návratu do sítě a ta dle nich mohla aktualizovat svůj stav.

Celý proces základního odpojení tedy probíhá takto:

- Stanice S_i zvolí svého zástupce. Tím by měla být co nejvýkonnější aktuálně připojená stanice, nejlépe pro tuto funkci přímo předurčená.
- Stanice předá zástupci oprávnění ji zastupovat z pohledu ukládání požadavků na změny dat (logů).
- Ověří si, že zástupce svou funkci přijal a odpojí se.

Po opětovném připojení stanice S_i pak dojde k těmto krokům:

- Nejprve stanice kontaktuje svého zástupce. Pokud se ten mezitím také odpojil, zjistí, kdo byl „jmenován" na jeho místo.
- Převezme od zástupce seznam updatů, které za dobu jeho nepřítomnosti na jeho konto nashromáždil a vykoná je nad svou replikou databáze.
- Převezme nazpět od zástupce oprávnění přijímat požadavky adresované této stanici a zahájí normální provoz.

Teoreticky může nastat i případ, že se postupně odpojí všechny stanice. Poslední z nich pak bude vlastně zástupcem všech ostatních a bude mít jejich aktualizační informace, které ovšem nebude mít komu předat. Po přihlášení stanic k síti pak tyto nebudou moci zahájit běžný provoz, dokud se znovu nepřipojí ona poslední odpojená stanice a neposkytne jim příslušné aktualizační požadavky. [2]

Kontrolní mód

Pokud se chce stanice S_i odpojit a mít přitom možnost měnit určitá data v databázi, musí před samotným odpojením svůj úmysl dát patřičně na vědomí ostatním stanicím. Nejlépe lze výhradního přístupu k určité části databáze dosáhnout jejím dvoufázovým uzamčením pro zápis v rámci transakce. U této transakce, zvané pseudotransakce, je pak třeba deaktivovat *deadlock*, který by mohl nastat s ostatními klasickými transakcemi a patřičně upravit *timeout*, aby transakce nebyla po určité době zrušena. Tato pseudotransakce pak zamezí ostatním stanicím přístup k této části databáze a stanice S_i

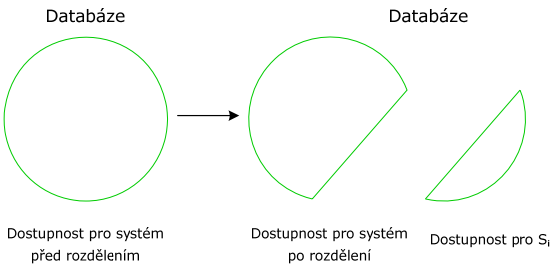
na ní může mezitím v offline režimu pracovat bez obav, že po jejím opětovném připojení dojde k synchronizačním konfliktům.

Celý postup odpojení stanice S_i v kontrolním módu pak probíhá takto:

- Stanice S_i zvolí svého zástupce, jako je tomu v případě základního odpojení a provede všechny jeho kroky předávání oprávnění.
- Zároveň S_i zahájí pseudotransakci, která umístí zámek pro zápis na příslušná data v distribuované databázi.
- Proběhne-li toto uzamčení dat úspěšně, stanice se odpojí s kontrolním oprávněním na tato uzamčená data. Pokud uzamykání selže, což by znamenalo, že nějaká jiná transakce právě pracuje s požadovanými daty, může to S_i zkusit znovu, případně vynechat ze zamykané oblasti tato již zablokovaná data.

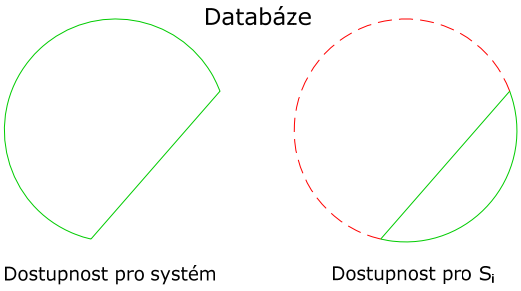
Po znovu připojení stanice S_i do sítě se nejprve provedou kroky totožné s těmi u základního připojení. V dalším kroku stanice v rámci stále aktivní pseudotransakce rozešle nové hodnoty pro položky, které změnila za dobu svého odpojení, a pesudotransakci potvrdí, čímž zároveň uvolní zámky na těchto datech.

Tato pravidla jsou shodná pro všechny druhy odpojení se v kontrolním módu: rozdělení databáze, kontrolní mód se čtením zbytku DB pro stanici, kontrolní mód se čtením zbytku DB pro systém.



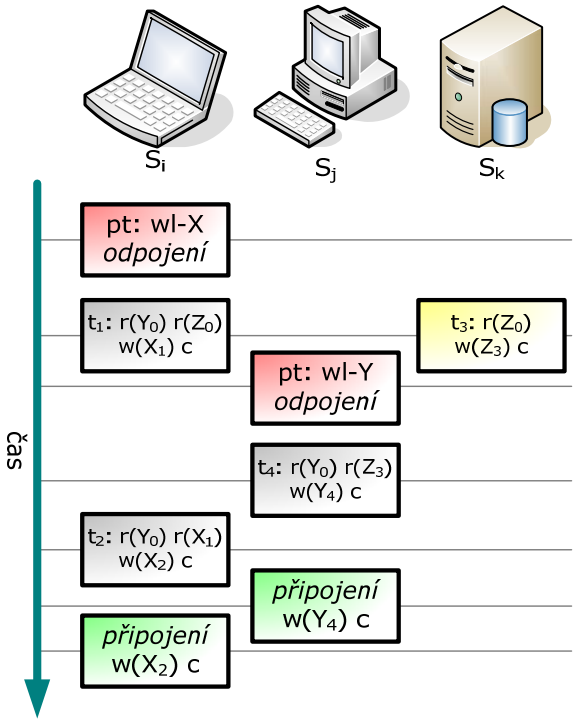
Obr. 11 – Část databáze je pod výhradní kontrolou odpojené stanice S_i [2]

Rozdělení databáze je pak nejjednodušší metodou z kontrolních módů. Stanice, jenž se chce odpojit, zablokuje příslušná data pseudotransakcí a odpojí se. Díky tomu má přístup k těmto datům pouze ona a nikdo jiný. Systém pak zase může pracovat se zbytkem databáze, do kteréhož mu pro změnu nemůže zasahovat tato odpojená stanice. Jelikož je databáze striktně rozdělena, změněná data obou částí lze poté zase snadno spojit (viz Obr. 11).



Obr. 12 – Výhradní kontrolní mód nad částí databáze s možností čtení jejího zbytku odpojenou stanicí S_i [2]

O něco složitější je *kontrolní mód*, ve kterém má odpojená *stanice k dispozici zbytek databáze pro četní*.



Obr. 13 – Příklad průběhu operací s daty u výhradního kontrolního módu nad částí databáze s možností čtení jejího zbytku odpojenými stanicemi

Na Obr. 13 je uveden příklad možného průběhu operací s daty při tomto přístupu. V síti existují tři stanice S_i, S_j a S_k. Průběh jednotlivých operací v čase následují od shora dolů. Proměnná *pt* zastupuje pseudotransakci, t_i klasickou transakci, *wl-X* zámek pro zápis na datovou jednotku *X*, *r(X)* operaci čtení, *w(X)* operaci zápisu, *c* commit transakce, X_i, Y_i, Z_i tři různé datové jednotky verze *i*.

V tomto příkladu nejprve stanice S_i uzamkne datovou jednotku *X* pomocí pseudotranskacke a odpojí se. Transakce t_1 a t_2 tak na této stanici proběhnou v offline režimu. Jako další se odpojí stanice S_j, před čímž ještě uzamkne datovou jednotku *Y*. V offline režimu spustí transakci t_4. Stanice S_k zůstala připojená po celou dobu, přičemž spustila transakci t_3.

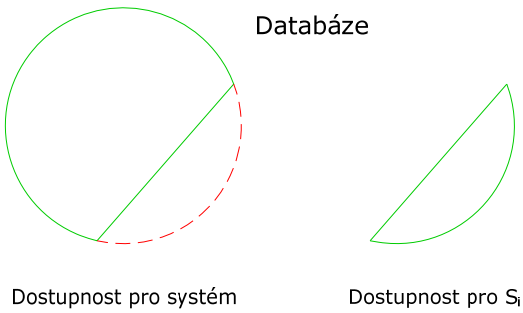
Za povšimnutí stojí, že stanice S_i může provádět transakce během svého odpojení aniž by musela provést uzamčení ostatních dat pro čtení před svým odpojením. To proto, neboť je vždy přečtena hodnota verze těchto dat, kterou měly těsně před odpojením stanice. Pro zachování konzistentnosti je pak nezbytné, aby bylo možné serializovat všechny transakce provedené S_i během jejího odpojení. Toho lze docílit pokud budou splněny tyto podmínky:

- Modifikována mohou být pouze data, která byla uzamčena pro zápis pseoudtransakcí před odpojením stanice S_i.
- Tato data nemohou být po uzamčení pseudotransakcí přečtena ani změněna nikým jiným a pseudotransakce nemůže být zrušena (s výjimkou S_i).
- Data neuzamčená pseudotransakcí může odpojená stanice používat pouze pro čtení, nebyla-li ovšem uzamčena pro zápis jinou stanicí.

Tato tři pravidla garantují, že bude možné jednotlivé transakce všech zúčastněných stanic serializovat, jelikož dodržují dvoufázový uzamykací protokol. Na Obr. 13 je vidět, že transakce t_4 je spuštěna za podmínek, kdy je datová jednotka *X* uzamčena pro zápis pseudotransakcí a ostatní data mohou být uzamčena transakcí která pokračuje i po odpojení stanice S_j. Ekvivalentní a správné sériové pořadí těchto transakcí tedy je t_1, t_2, t_3, t_4. Veškeré transakce prováděné odpojenou stanicí jsou při serializaci po jejím opětovném připojení brány za sebou tak jak byly vykonány, jako jedna velká transakce zaštítěná pseudotransakcí, s počátkem odpojení této stanice.

K porušení uvedených pravidel však může celkem snadno dojít. Nechť stanice S_i uzamkne datovou jednotku *X* a odpojí se. Poté stanice S_k uzamkne datovou jednotku *Y* a odpojí se. Ač je tedy *Y* uzamknuto pseudotransakcí stanice S_k, a nikdo by tedy neměl mít k němu přístup ani pro čtení ani pro zápis, stanice S_i může datovou jednotku *Y* bez problémů číst ze své lokální kopie pro čtení, protože o zámku vytvořeném stanicí S_k po jejím odpojení neví.

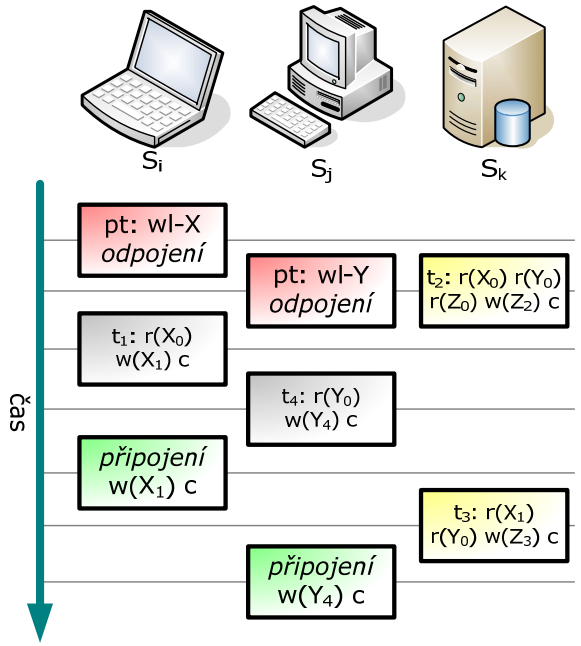
Pokud se v tomto případě první znovu připojí stanice S_k, upraví hodnotu datové jednotky *Y* a provede změny, které mu nastřádal jeho zástupce. Když se připojí S_i neměl by vůbec zjistit, že S_k byl kdy odpojen. Dojde-li ke znovu připojení stanic v opačném pořadí, bude průběh následující: První se připojí S_i, nejprve upraví hodnotu datové jednotky *X* a uvolní zámek na tuto jednotku potvrzením (commit) pseudotransakce. Že byla datová jednotka *Y* uzamčena poté zjistí až při vykonávání změn podle seznamu od zástupce. Poté se připojí S_k, upraví hodnotu datové jednotky *Y* a provede změny podle seznamu od zástupce.



Obr. 14 – Kontrola pro zápis nad částí databáze odpojenou stanicí S_i, přičemž zbytek systému má k dispozici pro čtení verzi před jejím odpojením [2]

Další možností přístupu je *kontrolní mód*, nad částí databáze, která zároveň zůstane ostatním neodpojeným stanicím *k dispozici pro četní*.

Obr. 15 na příkladu demonstruje postup tohoto přístupu při zpracování operací na třech stanicích.

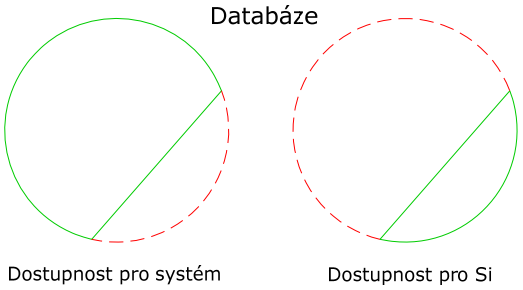


Obr. 15 – Příklad průběhu operací s daty u kontrolního módu nad částí databáze s možností jejího současného čtení zbytkem systému

V tomto příkladu mají při odpojení stanic S_i a S_j ostatní stanice stále přístup pro čtení k původním hodnotám datových jednotek, jež uzamkly. Tudíž transakce t_2 přečte staré hodnoty X a Y, zatímco t_3 přečte hodnotu X již aktualizovanou stanicí S_i a u Y stále její starou hodnotu. Správné serializované pořadí transakcí je tedy t_2, t_1, t_3, t_4. I v tomto případě jsou veškeré transakce prováděné odpojenou stanicí při serializaci po jejím opětovném připojení spuštěny za sebou tak jak byly vykonány, jako jedna velká transakce zaštítěná pseudotransakcí, ovšem její zařazení se tentokrát odvíjí od chvíle znovu připojení této stanice. Podmínky pro funkčnost tohoto přístupu jsou tedy následující:

- Odpojená stanice S_i smí měnit pouze data uzamčená její pseudotransakcí.
- Původní hodnoty těchto datových jednotek mohou být přečteny ostatními připojenými stanicemi. Poté co se S_i znovu připojí a aplikuje změny těchto dat, změní tím zámek pro čtení na tato data na zámek pro zápis, aby transakce ostatních stanic, které měly načtené staré hodnoty a pracovali s nimi byly zrušeny.
- Datové jednotky neuzamčené pseudotransakcí nejsou odpojené stanici dostupné ani pro zápis ani pro čtení. [2]

Volný kontrolní mód



Obr. 16 – Volný kontrolní mód [2]

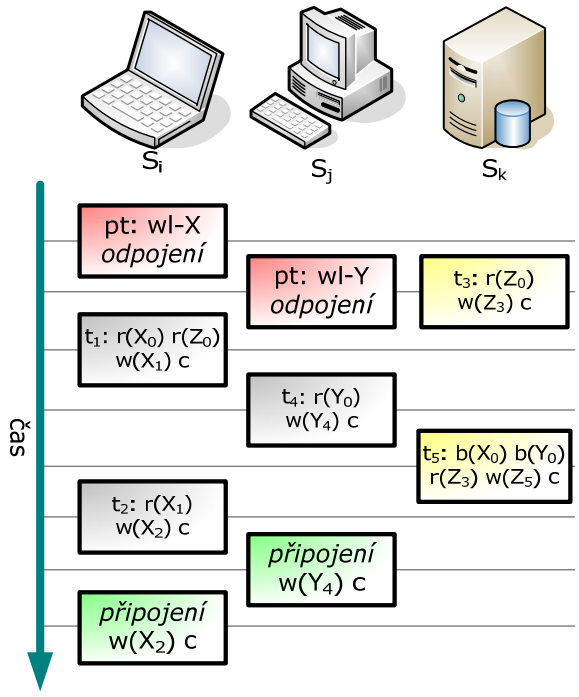
V případě volného kontrolního módu mohou všechny stanice, bez ohledu na to, jsou-li právě připojeny či nikoli, číst kteroukoli část databáze. Tento přístup je v podstatě kombinací kontrolních módů, při nichž má odpojená stanice k dispozici zbytek databáze pro četní a zároveň mohou ostatní připojené stanice číst původní hodnoty části databáze uzamčené odpojenými stanicemi.

Pro řešení těchto přístupů zavedeme krom standardních operací čtení (read - *r*) a zápis (write - *w*) ještě operaci *prohlížení* (browse - *b*). Prohlížení znamená, že je hodnota datové jednotky přečtena pouze pro „prohlížení" a nebude součástí žádného výpočtu, který by mohl ovlivnit jiná data v databázi operací zápis. Přečtená hodnota je totiž aktuální hodnota datové jednotky, která je součástí dosud nepotvrzené (či nezamítnuté) transakce. Rozšířená tabulka kompatibilních operací pak bude vypadat následovně:

	r	w	b
r	✓	✗	✓
w	✗	✗	✓
b	✓	✓	✓

Jak je z této tabulky patrné, prohlížení je kompatibilní vždy, ať je již datová jednotka uzamčena pro čtení či zápis. Tudíž, uzamkne-li odpojující se stanice datovou jednotku pro zápis, aby k ní měla výhradní přístup po dobu své nepřítomnosti, mohou i přesto ostatní stanice tuto původní hodnotu kdykoli prohlížet. Stejně tak odpojená stanice smí prohlížet libovolnou původní hodnotu databáze, aniž by ji musela mít uzamčenu. Tento přístup povoluje některá neserializované spouštění a umožňuje větší souběžnost operací.

Obr. 17 na příkladu demonstruje postup volného kontrolního módu při zpracování operací na třech stanicích s využitím prohlížecích zámků.



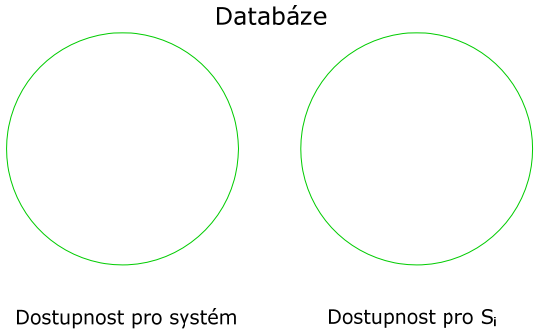
Obr. 17 – Příklad průběhu operací s daty u volného kontrolního módu

V tomto příkladu transakce t_5 stanice S_k nemůže získat zámek pro čtení datových jednotek X ani Y, proto se musí spokojit se zámky pro prohlížení.

Celý tento přístup funguje následovně: Stanici S_i je povoleno uzamknout určitou část databáze odpojit se a zároveň číst data ze zbytku databáze, která uzamčena nebyla. Ostatní stanice pak mohou data uzamčená S_i pouze prohlížet. Pravidla fungování tohoto postupu jsou tato:

- Odpojená stanice S_i může měnit pouze data, která předtím uzamkla pomocí pseudotransakce.
- Tyto datové jednotky uzamčené stanicí S_i mohou být po dobu jejího odpojení ostatními stanicemi pouze prohlíženy. Pokud by některá stanice požadovala zápis do těchto jednotek, bude její transakce zamítnuta a zrušena, neboť čekání na uvolnění zámku by mohlo trvat nepřiměřeně dlouho.
- Datové jednotky neuzamčené pseudotransakcí může stanice S_i během svého odpojení pouze číst. [2]

Optimistický kontrolní mód



Obr. 18 – Optimistický kontrolní mód [2]

Optimistický kontrolní mód umožňuje všem stanicím, ať již jsou připojeny k síti nebo ne, libovolný přístup ke kterékoli části databáze, není-li tato uzamčena lokálně. Tento přístup však může způsobovat konflikty zapříčiňující zamítání transakcí stanic, provedených během odpojení, při pokusu prosadit jejich změny do společné databáze při znovu připojení stanice.

Existuje sice řada přístupů, jak tyto konflikty řešit a minimalizovat jejich počet, ovšem i tak je toto řešení použitelné pouze ve specifických případech. Následují určité metody řešení konfliktů:

- Transakce na stanici S_i během jejího odpojení přistupující k datovým jednotkám, které byly při odpojení přiřazeny ke správě této stanici, smí být potvrzeny. Při znovupřipojení jsou pak změny v těchto datech provedené stanicí S_i upřednostněny před změnami provedenými jinými stanicemi.
- Transakce na stanici S_i, která přistupuje k jiným datům, než těm, která jí byla přiřazena může být v době odpojení potvrzena. Při prosazování těchto změn do společné databáze po znovu připojení jsou však upřednostněny případné změny provedené jinými stanicemi.
- Transakce na stanici S_i měnící jak data jí přiřazená tak i ta ostatní může být při znovu připojení v případě konfliktu zamítnuta celá.
- Transakce ostatních připojených stanic jsou omezeny shodně. [2]

Závěry

Navrhované přístupy k řešení odpojování, práci v terénu a znovu připojování stanic do sítě distribuovné replikované databáze detailně popisuje možnosti a postupy tohoto řešení. Za určitých okolností by mohlo být jistě převedeno do praxe a s úspěchem provozováno, avšak je zde i řada nedostatků.

Předně jde o předpoklad, že každá stanice musí mít u sebe replikovánu kompletní databázi. Tento předpoklad však zrovna u mobilních zařízeních s limitovanou paměťovou kapacitou nemusí být vždy splněn. Ze hry tedy rovnou vypadávají malá mobilní zařízení typu PDA, MDA a další. Naopak uplatnit by se zde mohly například notebooky.

Další problém může nastat v případě, že se některá ze stanic po svém korektním odpojení již do sítě nevrátí (například může být zničena při nehodě nebo odcizena). V tom případě by data touto stanicí uzamčená již nikdy nebyla přístupná bez konkrétního zásahu administrátora.

Tyto přístupy odpojovaných stanic mohou nalézt uplatnění například ve firmě se stabilní podnikovou sítí, zaměstnávající terénní pracovníky, kteří se často vrací na některou z poboček a mají dostatečně výkonná mobilními zařízení, přičemž by každý z nich měl mít s dostatečně vysokou pravděpodobností vymezenu část databáze za kterou by byl zodpovědný.

Závěr

V tomto článku byla názorně prezentována jak teoretická základna mobilních databázových systémů, tak na konkrétních příkladech ze současnosti představen jejich aktuální stav v komerční sféře, a v neposlední řadě i nastíněny a zhodnoceny dvě teoretická řešení různých problémů z této oblasti.

Mobilní databáze se stávají stále více využívanou a požadovanou nezbytností. Jakékoli další objevy v tomto oboru tak jsou vítány, ba přímo vyžadovány. Současné mobilní databázové systémy jsou již na celkem vysoké úrovni, avšak vývoj stále kvalitnějšího hardwaru pro mobilní zařízení dává prostor pro nasazování neustále kvalitnějších a komfortnějších databázových řešení a aplikací. Papírové formuláře se tak stávají minulostí, jelikož přímý zápis dat do databáze z místa a okamžiku jejich pořízení je nejenom pohodlnější, ale i daleko přesnější a pohotovější.

Mobilní databázové systémy jsou tedy oblastí s velkým potenciálem do budoucna, přičemž v teoretických řešeních mají stále rezervy, avšak s klesající tendencí.

Literatura

[1] Connolly, T.M.; Begg, C.E.: *Database systems: a practical approach to design, implementation, and management*, 4th edition, Addison-Wesley, UK, 2005, ISBN 0-321-21025-5.

[2] Holliday, J.; Agrawal, D., Abbadi, A.E.: Disconnection Modes for Mobile Databases, Kluwer Academic Publishers, Hingham, MA, USA, 2002.

[3] Holubec, J.: *Ano mobilnímu přístupu*, dbsvet.cz, ČR, 2005.

[4] Chan, D.; Roddick, J.F.: *Context-Sensitive Mobile Database Summarisation*, Australian Computer Society, Darlinghurst, Australia, Australia, 2003.

[5] Kinský, F.: *Z off-line zařízení na podnikové databáze*, dbsvet.cz, ČR, 2006.

[6] Monteiro, J.M.; Brayner, Â.; Lifschitz, S.: *A Mechanism for Replicated Data Consistency in Mobile Computing Environments*, ACM Press, Seoul, Korea, 2007.

[7] Nouali, N.; Doucet, A.; Drias, H.: *A Two-Phase Commit Protocol for Mobile Wireless Environment*, Australian Computer Society, Darlinghurst, Australia, Australia, 2005.

[8] Palazzo, S.; Puliafito, A.; Scarpa, M.: *Design and evaluation of a replicated database for mobile systems*, Kluwer Academic Publishers, Hingham, MA, USA, 2000.

[9] Rydval, S.: *Databáze do dlaně*, Softwarové noviny, ČR, říjen 2002.

[10] Wikipedie, cs.wikipedia.org.