UNIVERZITA PALACKÉHO V OLOMOUCI
PEDAGOGICKÁ FAKULTA
Katedra informační a technické výchovy

Bakalářská práce

Petr Mališ

Informační výchova se zaměřením na vzdělávání
Geografie

Možnosti transakčního chování databázových systémů
a jejich využití v modulech školních informačních systémů

Olomouc 2016

Vedoucí práce: Mgr. Jan Kubrický, Ph.D.

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně pod odborným dohledem vedoucího práce a uvedl jsem všechny použité podklady a literaturu.

V Olomouci dne 18. 4. 2016 Podpis:……..

Poděkování

Děkuji Mgr. Janu Kubrickému, Ph.D., za odborné vedení bakalářské práce a poskytnutí cenných rad k práci.

Obsah

Úvod

Informační systémy dnes pronikly snad do každé profese a ulehčují práci mnoha firmám a institucím. Výjimkou není ani školství, ve kterém informační systémy zaštiťují různé funkce, mezi ty například patří komunikace s rodiči, klasifikace žáků, plánování školního roku, elektronická třídní kniha nebo organizace přijímacího řízení. Školní informační systém pracuje s informacemi, které jsou ukládány a načítány z databáze. Zpracovaná data jsou mnohdy důležitá, ale i citlivá. Je proto nutné, aby při zápisu do databáze nebyla nijak poměněna, poškozena nebo aby nedošlo jen k zápisu části údajů. Stejně i při čtení dat musí databázový systém zajistit, aby data zobrazená uživatelům v prostředí aplikace odpovídala vloženým datům.

Při práci s databázovými systémy by každá zadaná akce měla být provedena formou transakce. Ta je spravována řízením souběžného přístupu, který zajišťuje, ať se při zpracovávání transakcí neobjeví konkurenční konflikty. Ty by mohly zapříčinit nekonzistentní a nekorektní stav databáze, kterému se chceme vyvarovat. Existuje mnoho metod k řešení souběžného přístupu. Ale použití stejné metody může mít různorodé řešení v jednotlivých distribuovaných databázových systémech.

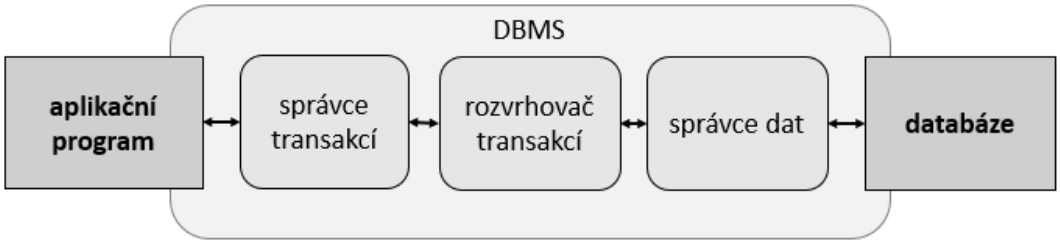
Hlavním cíle práce je uvést čtenáře do problematiky transakčního zpracování. Jsou zde vysvětleny základní termíny spojené s transakčním přístupem a anomálie, jenž mohou při zpracování nastat. Také jsou zde popsány metody řešící souběžný přístup a možnosti navrácení do výchozího konzistentního stavu. Dále se práce zabývá implementací řízení souběžného přístupu v RDBMS PostgreSQL. Ten byl vybrán, protože se jedná o jeden z nejrozšířenějších databázových systémů, který je navíc volně dostupný. Pro porovnání je obsažen i náhled do metod, které využívají konkurenční RDBMS Oracle a MySQL. V další části jsou zanalyzovány vybrané moduly školních informační systémů, na které jsou poznatky o transakčním přístupu aplikovány.

1 Transakce

S databázovými transakcemi se nepřímo setkáváme dennodenně. Každý vyspělý DBMS, který je využíván např. v informačních systémech nebo ve webových aplikacích, vykonává pomocí transakcí zadané úkony. Díky užití transakcím se zajišťuje konzistentní stav databáze a správné zapsání nebo navrácení informací. Proto je podpora a zvolené řešení transakčního přístupu důležitou součástí DBMS.

Transakce je označení pro logickou jednotku práce, ale i pro jednotku návratu a konzistence. Je to jedna ze základních funkcí, kterou DBMS poskytuje. Chápeme ji jako rozsáhlejší atomickou akci, která vede k převodu databáze z jednoho konzistentního stavu do druhého. Při zadání transakce jsou provedeny akce vedoucí ke čtení nebo zápisu dat databáze.

Příkaz zadaný přes aplikaci je zpracováván správcem transakcí. Podle požadavků se vytvoří rozvrh, podle kterého budou transakce provedeny. Jednotlivé akce zápisu a čtení vykonává podle rozvrhu správce dat, který přistupuje k databázi. Získaná data jsou navrácena správcem transakcí zpět do aplikace.



Obrázek 1. Architektura zpracování transakcí. (zdroj: vlastní zpracování)

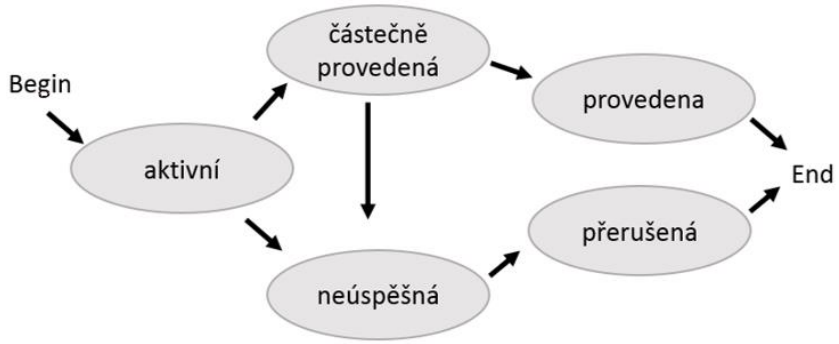
Dle standartu ANSI/ISO je definován model transakce SQL, ten využívá většina DBMS na trhu. Začátkem transakce je chápán první příkaz SQL, který je proveden. Ale může být také uvozena klauzulí BEGIN TRANSACTION (zahájit transakci). Dále následuje jeden nebo více SQL příkazů. Ty mohou buď data číst, nebo je zapisovat. Podle toho, které akce bude transakce vykonávat, je rozdělujeme na čtecí (READ ONLY) a aktualizující (READ WRITE). Konec transakce nastává buď stavem COMMIT (potvrzením), nebo ROLLBACK (navrácením). Při úspěšném provedení je transakce potvrzena, databáze je aktualizována a nachází se v novém konzistentním stavu. Při výskytu chyby, nebo zrušení transakce v jejím průběhu musí dojít k navrácení databáze do původního konzistentního stavu před počátkem operace.

Jako první evokuje pojem transakce v člověku převod peněz. A ten sám o sobě je i dobrým příkladem, proč se transakce využívají a jak vlastně vypadají. Na příkladu (Obrázek 2.) vidíme kód bankovního převodu. Zobrazená transakce obsahuje dva příkazy, které zajišťují převod peněz mezi dvěma účty. Kdyby byl proveden pouze jeden z nich, došlo by k narušení konzistence databáze, Jeden z účtu by se nenacházel v korektním stav. Díky použití transakce se nemůže stát, že by se peníze odečetly pouze z jednoho účtu nebo se pouze na jeden účet přičetly. Při zdařilé transakci bude ukončena s výsledkem COMMIT, budou provedeny obě akce, odečtení určité hodnoty z účtu konto1 a přičtení stejné částky na konto2. Jestli dojde během procesu k nějaké chybě, transakce se ukončí s výsledkem ROLLBACK. A databáze se navrátí do původního stavu přes spuštěním samotné transakce.

```
BEGIN TRANSACTION

UPDATE konto1 {balance = balance – 100};
if any error ocurred THEN GO TO UNDO; END IF

UPDATE konto2 {balance = balance + 100};
if any error ocurred THEN GO TO UNDO; END IF

COMMIT: GO TO FINISH;

UNDO: ROLLBACK;

FINISH: RETURN;
```

Obrázek 2. Příklad transakce (zdroj: zpracováno podle 4)

Transakce od svého zadání může nabýt několika stavů. Transakce je *aktivní* od okamžiku jejího spuštění až do jejího ukončení. Po provedení konečného příkazu se stane *částečně provedenou*. Pokud transakce nakonec je potvrzena, nazýváme ji *provedenou* transakcí. Jestli ale systém zjistí nějakou chybu a rozhodne, že transakce nemůže být úspěšně dokončena, zruší ji a transakce se stane *neúspěšnou*. Pak nabývá stavu *přerušené* transakce v okamžiku, kdy DB bude navrácena zpět do stavu před zahájením transakce. Z tohoto bodu může být znovu restartována nebo zrušena.



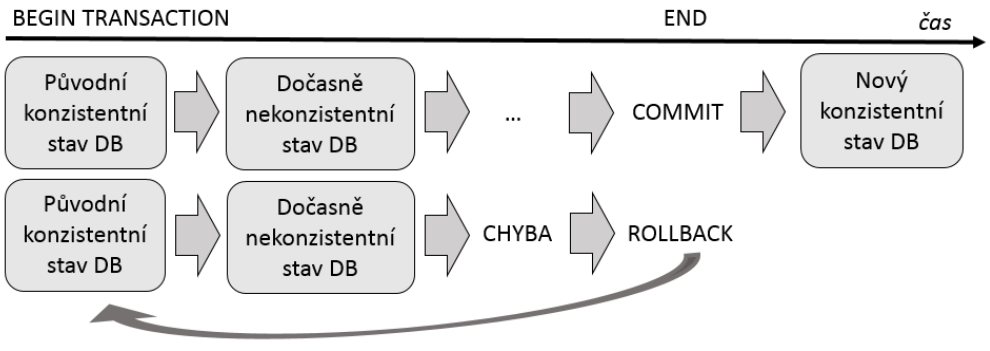
Obrázek 3. Stavy transakcí (zdroj: vlastní zpracování)

1.1 ACID vlastnosti

K spolehlivému chodu transakcí musí DBMS obsahovat mechanismy, které zajišťují integritu dat. Ty jsou shrnuty v ACID vlastnostech. ACID je akronymem pro *Atomicity, Consistency, Isolation, Durability* neboli atomičnost, konzistence, izolace a trvanlivost.

Atomičnost – každá transakce je chápaná jako jeden celek, jehož příkazy musí být všechny vykonány, nebo nesmí být provedeny vůbec a změny musí být navráceny. Zjednodušeně řečeno buď proběhne celá transakce, nebo nic.

Konzistence – při použití transakce je databáze převedena z jednoho konzistentního stavu do druhého. Transakce neumí rozpoznat pravdivost nových dat, kontroluje pouze, jestli byly splněny všechny její akce, tedy jestli se DB nachází v korektním stavu.



Obrázek 4. Stavy DB při zpracování transakce (zdroj: vlastní zpracování)

Izolovanost – jednotlivé transakce jsou mezi sebou odděleny, jsou pro sebe skryty. Nemůže se stát, aby najednou chtěly dvě transakce modifikovat stejné pole v databázi. Jejich akce musí být provedeny postupně. Splnění této vlastnosti hlídá řízení souběžného přístupu.

Trvanlivost – jakmile je transakce provedena, tak její výsledné změny v DB budou vždy zachovány, i když nastane například pád systému. Funkce navrácení, zaručující tuto vlastnost, je popsána v kapitole 1.3.

1.2 Řízení souběžného přístupu

Jednou z hlavních funkcí databází je dnes zajištění sdíleného přístupu k datům pro několik uživatelů najednou. Jestli by transakce chtěly pouze načítat stejná data, nemůže nastat konkurenční konflikt. Pokud by je ale více transakcí chtělo najednou i modifikovat, nastal by konflikt a výsledná hodnota pole by byla nekorektní a došlo by k narušení konzistence DB.

Nejjednodušším řešením, jak se vyhnout konfliktům souběžnosti a zajistit souběžný chod transakcí, je neumožnit chod více transakcí najednou. Transakce by mohla být spuštěna jen v té chvíli, až by byla ta předchozí dokončena. To by ale vedlo k malé efektivitě a velké časové náročnosti. Proto DBMS umožňují paralelní chod transakcí. Kontrolu konfliktů mezi souběžně jdoucími transakcemi zajišťuje concurrency control (řízení souběžného přístupu). Conolly (2009) jej definuje jako: „Proces správy více současných operací s databází tak, aby si tyto operace navzájem nepřekážely.“

1.2.1 Rozvrhy

Rozvrhem označujeme soubor transakcí, kterým je určena postupnost jednotlivých akcí u jedné či více transakcí v čase. Jsou vytvářeny manažerem konkurenčního přístupu a zajišťují souběžný chod. Díky rozvrhům bude databáze po úpravách vždy v novém konzistentním stavu. Výsledky různých použitých rozvrhů ale mají vliv na finální data a hodnoty se mohou lišit. Rozvrhy dělíme na sériové (serial), uspořádatelné (serializable) a zotavitelné (recoverable).

Sériovým rozvrhem rozumíme, že dvě a více transakcí bude provedeno postupně, jejich akce se nebudou střídat a nebudou provedeny najednou. Jak je ukázáno na příkladu, konečné hodnoty různého seřazení transakcí se mohou měnit podle toho, která z transakcí

byla umístěna v rozvrhu jako první. Proto by při užití sériového rozvrhu mělo být uvedeno řazení transakcí.

Tabulka 1. Sériový rozvrh T_A, T_B (zdroj: vlastní zpracování)

T_A	T_B	X	Y
		20	20
READ(X); X = X + 10		30	
WRITE(X)			
READ(Y); Y = Y / 10			2
WRITE(Y)			
COMMIT			
	READ(X); X = X * 3	90	
	WRITE(X)		
	READ(Y); Y = Y + 20		22
	WRITE(Y)		
	COMMIT		
Konečná hodnota:		90	22

Tabulka 2. Sériový rozvrh T_B, T_A (zdroj: vlastní zpracování)

T_A	T_B	X	Y
		20	20
	READ(X); X = X * 3	60	
	WRITE(X)		
	READ(Y); Y = Y + 20		40
	WRITE(Y)		
	COMMIT		
READ(X); X = X + 10		70	
WRITE(X)			
READ(Y); Y = Y / 10			4
WRITE(Y)			
COMMIT			
Konečná hodnota:		70	4

V *uspořádatelném rozvrhu* jsou akce transakcí proloženy. Výsledky různě uspořádaných rozvrhů mohou být jiné. Sériový rozvrh je vždy uspořádatelný, ale ne každý uspořádatelný rozvrh je zároveň sériový. Jestliže transakce byly provedeny uspořádaně, budou výsledně zapsaná korektní data, ale pouze v případě, jestliže použitému rozvrhu odpovídá nějaké sériové zpracování transakcí. Uspořádatelnost lze dokázat z precedenčního grafu.

Tabulka 3. Uspořádatelný rozvrh (zdroj: vlastní zpracování)

T_A	T_B	X	Y
READ(X); X = X + 10		20	20
WRITE(X)		30	
	READ(Y); Y = Y + 20		
	WRITE(Y)		40
READ(Y); Y = Y / 10			
WRITE(Y)			4
	READ(X); X = X * 3		
	WRITE(X)	90	
COMMIT	COMMIT		
Konečná hodnota:		90	4

Sériový rozvrh je vždy *zotavitelný*, protože transakce jsou zpracovávány postupně. Dokud jedna z transakcí neskončí, není tedy potvrzena nebo navrácena, další transakce nemůže začít. V případě uspořádatelného rozvrhu tomu tak nemusí být vždy. Rozvrh nazýváme zotavitelným pouze tehdy, jestli transakce T_B bude potvrzena až v bodě, ve kterém je předchozí transakce T_A, která upravovala hodnoty načtené T_B, již potvrzena. V 4. tabulce je popsán příklad nezotavitlného rozvrhu, ve kterém dojde k potvrzení T_B dřív než k vyhodnocení transakce T_A, a jeho upravené zotavitelné verze, kdy transakce T_A je potvrzena před transakcí T_B.

Tabulka 4. Zotavitelný a nezotavitelný rozvrh (zdroj: vlastní zpracování)

Zotavitelný rozvrh		Nezotavitelný rozvrh	
T_A	T_B	T_A	T_B
WRITE(Q)		WRITE(Q)	
	READ(Q)		READ(Q)
	WRITE(Q)		WRITE(Q)
ROLLBACK			COMMIT
	COMMIT	ROLLBACK	

Nezotavitelnému rozvrhu se můžeme vyhnout podmínkou, ta určí, že jedna transakce může číst pouze data, která byla již potvrzena, nebo můžeme využít pro navrácení *kaskádového rušení (cascading aborts)*. Kaskádové rušení spolu

se zamítnutou transakcí zruší i všechny ostatní transakce, pracující s hodnotami, které byly navrácenou transakcí upravovány.

1.2.2 Anomálie souběžného přístupu

Řízení souběžného přístupu zamezuje výskytům anomálii, které se při práci v databázi mohou vyskytnout. Jedná se o problémy *ztracené aktualizace* (špinavý zápis), *nezávazné závislosti* (špinavé čtení), problém *nekonzistentní analýzy* (neopakovatelné čtení) a *fantómové čtení*.

Ztracená aktualizace (lost update)

Tento problém nastává při přepsání hodnoty jedné transakce druhou. Na příkladu v tabulce č. X vidíme, že jsou spuštěny těsně za sebou dvě transakce, které budou přičítat k Q určitou hodnotu. Ve finále, i když obě transakce proběhly úspěšně, bude nová hodnota Q větší jen o hodnotu přičtenou druhou transakcí T_B, která zapisovala své změny jako poslední. Aktualizace provedena transakcí T_A bude ztracená, protože byla přepsána transakcí T_B. Proto je důležité, aby řízení souběžného přístupu zamezilo aktualizaci jednoho pole najednou více transakcemi.

Tabulka 5. Příklad ztracené aktualizace (zdroj: zpracováno podle 3)

T_A	T_B
Read(Q); Q = Q + 10	
	Read(Q); Q = Q + 20
Write(Q); COMMIT	
	Write(Q); COMMIT

Nezávazná závislost (dirty read)

Tato anomálie nastane při čtení nepotvrzených dat, kdy jedné transakci je dovoleno čtení nebo aktualizace pole dřív, než jiná probíhající transakce dokončí aktualizaci těchto dat. Když není probíhající transakce ještě potvrzena, vyskytuje se zde stále možnost, že bude navrácena. To by zapříčinilo ztrátu změněných dat, které už načetla druhá transakce a ve výsledku je chybně ponechá.

Na příkladu (tabulka 6) je znázorněno, že transakce T_B načte hodnotu Q, která byla aktualizovaná transakcí T_A. Ta ale na závěr selže a její změny budou navráceny. Transakce T_B stále bude pracovat s chybně načtenou hodnotou, protože předpokládá, že T_A doběhla úspěšně. Výsledná aktualizace Q bude chybná. Tomuto problému se dá

zabránit tak, že se nedovolí transakci načítat hodnotu polí, dokud běžící transakce, která modifikuje tuto pole, nebude ukončena.

Tabulka 6. Příklad nezávazné závislosti (zdroj: zpracováno podle 3)

Transakce A	Transakce B
READ(Q); Q = Q – 1; WRITE(Q) ROLLBACK	 READ(Q); Q = Q + 3; WRITE(Q); COMMIT

Nekonzistentní analýza (non-repeatable read)

Chyby mohou nastat nejen při zápisu ale i při čtení dat. Tento problém je podobný nezávazné závislosti s tím rozdílem, že u nezávazné závislosti je pracováno s daty, které neexistují. U problému nekonzistentní analýzy je pracováno s daty, které existují, ale jsou jiné než na začátku transakce. Transakce T_A přečte řádek, ten je vzápětí aktualizován transakcí T_B, která je potvrzena. Jestli T_A přečte znovu tento řádek, navrátí již jinou hodnotu, než poprvé.

Tabulka 7. Příklad nekonzistentní analýzy (zdroj: zpracováno podle 3)

Transakce A	Transakce B
READ(Q); Q = 10 READ(Q); Q = 20; COMMIT	 READ(Q); Q = Q + 10; WRITE(Q); COMMIT

Fantómové čtení (phantom)

Jedná se o případ, kdy je změněn počet vrácených řádků. Transakce T_A navrátí podle zadané podmínky určitý počet řádků, poté transakce T_B vloží nový řádek, který splňuje podmínky T_A. Jestli by transakce T_B zopakovala stejný dotaz, bude ji navrácen jiný počet řádků, než v prvním případě. Tento nový řádek je nazýván fantómem.

1.2.3 Izolační úrovně

Souběžné transakce musí být od sebe izolovány, jednotlivé akce nemohou narušit chod druhých transakcích. Podle ANSI/ISO SQL-92 standardu rozlišujeme 4 úrovně izolace transakcí, které zajišťují souběžný chod transakcí. Každá z úrovních zabraňuje určitým anomáliím. Problému ztracené aktualizace zabrání přímo databáze, nemůže se vyskytnout na žádné z úrovních. DBMS nemusí využívat všechny úrovně, stačí pouze ta nejvyšší, která ty nižší implementuje.

Nejvyšší úrovni je SERIALIZABLE, na které transakce proběhnou, jako kdyby byly prováděny za sebou, a nemohou se zde vyskytnout anomálie souběžnosti. Na úrovni REPEATABLE READ transakce neuvidí změny dat provedené souběžnými transakcemi, pracuje s potvrzenými daty, které databáze obsahovala na začátku spuštění. Přidané řádky souběžnými transakcemi jsou na této úrovni viditelné ostatním transakcím, proto se zde můžou objevit fantómy. Při užití úrovně READ COMMITTED transakce nevidí nepotvrzené změny provedené z jiných transakcí. Ale aktualizace, které jsou prováděny souběžně a jsou potvrzeny, mohou být viditelné a při použití dvou příkazů čtení může transakce navrátit dvě různé hodnoty. Proto bychom se měli vyhnout užití této úrovně pro transakce, které vytvářejí souhrny nebo provádí kalkulace. Nejnižší úrovní je READ COMMITTED. Není odolná proti potvrzeným i nepotvrzeným změnám souběžných transakcí. Výsledky navrácené transakcí na této úrovni nemusí být korektní a proto se zmiňovaná úroveň ve vyspělých DBMS nevyužívá.

Tabulka 8. Izolační úrovně podle standartu ANSI/ISO SQL (zdroj: zpracováno podle 7)

Izolační úroveň	Anomálie		
	Špinavé čtení	Neopakovatelné čtení	Fantóm
READ UNCOMMITTED	možné	možné	možné
READ COMMITTED	nemožné	možné	možné
REPEATABLE READ	nemožné	nemožné	možné
SERIALIZABLE	nemožné	nemožné	nemožné

1.3 Funkce navrácení

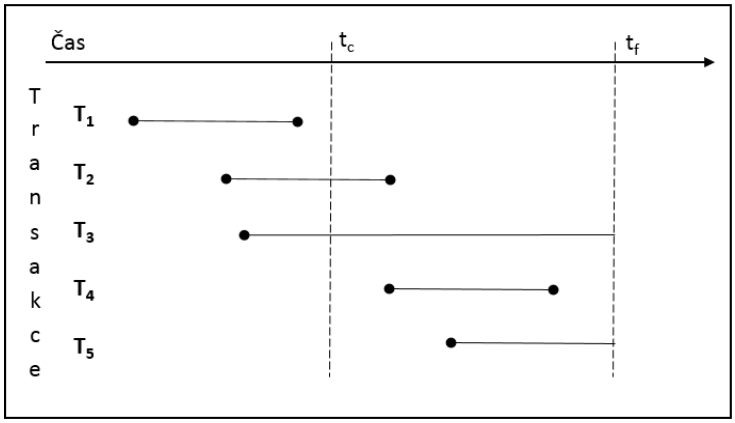
Funkce navrácení neboli zotavení databáze je základní služba, kterou by měl DBMS zajišťovat. Úzce souvisí právě s transakcemi, ty figurují jako jednotka návratu. Když je transakce potvrzena příkazem COMMIT, je ustanoven *commit point*. Ten označuje konec logické operace a nový konzistentní stav databáze. Při vyhodnocení transakce akcí ROLLBACK se databáze navrátí do posledního známého konzistentního

stavu. Jedná se o stav při spuštění dané transakce, poslední určený commit point. Dosáhne-li transakce commit pointu, změny v DB jsou trvalými. Před dosažením commit pointu se považují změny za dočasné a mohou být kdykoliv navráceny. Úspěšně provedenou transakci nelze vrátit zpět.

Jestliže některá z akcí transakce nebude úspěšně vykonána, je navrácena celá transakce. Důvodem může být systémová chyba, selhání hardwaru, chyba aplikace, nedbalost nebo konflikt zámků. Jestliže nastane například pád systému, probíhá zotavení pomocí správce transakcí, který ji provádí formou *WAL (write ahead log)*. Průběh všech transakcí je zaznamenán v časovém sledu do *logu* (souboru událostí), ve kterém jsou obsaženy všechny úkony transakce. Při restartu systému se záznam zpracovává pozpátku, navrací efekty nepovedených transakcí. Jedině tímto způsobem se dostane k původní hodnotě každého pole, protože mohlo být měněno několika po sobě jdoucími transakcemi. Po dosažení začátku logu postupuje vpřed a znovu spouští zadané transakce, které nebyly provedeny.

V logu se zaznamenávají informace o spuštění transakce, o zrušení a o provedení. A potom také veškeré změny, které transakce vykonala, ve formátu čas, identifikátor transakce, druh operace, změněný objekt, stará hodnota, nová hodnota. Dále se v pravidelných intervalech zaznamenávají *checkpointy*. Tento kontrolní bod označuje synchronizaci mezi logem a databází. Při checkpointu jsou nuceně všechny data z operační paměti zapsány do sekundární paměti. Díky těmto bodům nemusí systém při zotavení projíždět celý záznam logu, ale pouze k prvnímu, respektive poslednímu checkpointu.(3)

Veškeré změny databáze jsou udržovány ve vyrovnávací paměti a nezapíšou se do okamžiku, než transakce dosáhne commit pointu. Do DB a zároveň do souboru událostí jsou změny zapsány až při COMMIT akci.



Obrázek 5. Časový průběh transakcí (zdroj: zpracováno podle 3)

Na obrázku vidíme časový průběh transakcí. V čase t_f došlo k chybě systému. Poslední checkpoint před pádem systému byl vytvořen v čase t_c. Transakce T_1 byla úspěšně provedena před t_c. Transakce T_2 a T_4 doběhly po čase t_c, musí být znovu přepracovány. Transakce T_3 a T_5 nebyly zhotoveny před časem t_f a proto musí být navráceny.

Tabulka 9. Segment log souboru k obrázku č. 5 (zdroj: vlastní zpracování)

Čas	T ID	Operace	Objekt	Stará hodnota	Nová hodnota
	T_1	START			
	T_1	UPDATE	Q_1	(původní)	(nová)
	T_2	START			
t_c	T_3	START			
	T_2	UPDATE	Q_2	(původní)	(nová)
	T_1	COMMIT			
	T_2	COMMIT			
	T_4	START			
	T_3	DELETE	Q_1	(původní)	(nová)
	T_5	START			
	T_4	DELETE	Q_4	(původní)	
t_f	T_4	COMMIT			

2 Metody souběžného přístupu

V této kapitole jsou popsány nejčastěji využívané metody zajišťující paralelní chod transakcí. Souběžný přístup transakcí k datům představuje v DBMS důležitou funkci. Zpracování souběžných transakčních požadavků proběhne rychleji, než kdyby byly provedeny postupně. Tím se práce v DB zefektivní a je umožněna podpora přístupu více uživatelů najednou.

Metody užité pro řízení souběžného přístupu se dají rozdělit do dvou kategorií, podle toho jak nahlíží na problematiku souběžnosti. Více rozšířené jsou pesimistické metody, které předpokládají, že ke konfliktům mezi transakcemi dochází často, a proto je potřeba jim předcházet například zamykáním. Tím konfliktu zabrání, ale mohou tím vznikat jiné problémy, např. uváznutí transakcí.

Optimistický přístup předpokládá, že konflikty jsou jevem vyskytujícím se řídce a za efektivnější přístup tedy považují transakce nezdržovat. Je využíván v případech, kdy většina transakcí bude požadovat pouze čtení polí. Tyto příkazy samy o sobě nemohou narušit konzistentní stav databáze.

2.1 Metoda zámků

Nejvyužívanější metodou k zajištění souběžného přístupu je použití zámků. Ty zajišťují, aby při úpravě dat jednou transakcí byl odepřen přístup všem ostatním. Tímto zabraňují konfliktům souběžného přístupu. Transakce mohou požádat o dva základní zámky. Sdílený zámek (shared lock) dovoluje pouze čtení datové položky, a exkluzivní zámek (exclusive lock), při kterém transakce může danou položku nejen číst, ale i aktualizovat.

Tabulka 10. Kompatibilita zámků (zdroj: vlastní zpracování)

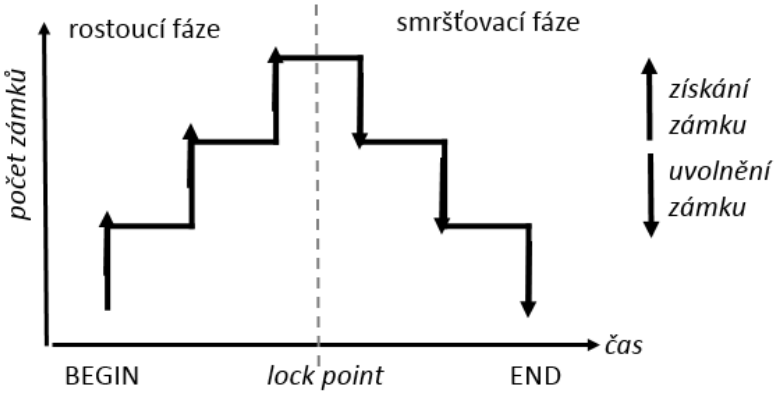
	sdílený	exkluzivní
sdílený	kompatibilní	nekompatibilní
exkluzivní	nekompatibilní	nekompatibilní

Sdílené zámky, které nedovolují změnu dat, se mohou na jedné položce DB vyskytnout od několika transakcí najednou. Ale jestli je už položka uzamknutá exkluzivním zámkem, nemůžou jiné transakce na daných datech provádět změny a ani je číst, tedy vytvářen nové zámky.

Transakce žádají správce souběžného přístupu o vytvoření zámku na určité data v DB. Jestli nenastane konflikt s jiným již existujícím zámkem na těchto datech, řízení souběžného přístupu pak transakci zámek udělí. V případě, kdy jsou požadovaná data již uzamčena, musí být určeno, jestli vytvoření nového zámku nenaruší ten stávající. Jestli je požadavku vyhověno, další zámek je vytvořen. V opačném případě je žádost zamítnuta a transakce musí čekat do uvolnění stávajícího zámku. Po dokončení transakce, ať už jejím zrušením nebo potvrzením, jsou data odemčena a položky uvolněny pro ostatní transakce.

2.1.1 Dvojfázové zamykání (2PL)

Tento uzamykací protokol je nejvyužívanější v komerčních databázových systémech. Udává pravidla, jak budou transakce uzamykat a odemykat objekty DB. Skládá se ze dvou fází. V první tzv. rostoucí (zamykací) fázi může transakce pouze vytvářet zámky. Při druhé fázi tzv. smršťovací (odemykající) je dovoleno transakci zámky jen rušit, ale nesmí žádné nové vytvářet. Po celou dobu rostoucí fáze může transakce získat tolik zámků, kolik potřebuje. Jakmile ale jeden zámek zruší, vstoupí do smršťovací fáze a může zámky jen rušit. Konec první fáze, moment kdy transakce obdrží finální zámek, je nazýván lock point (zamykací bod transakce). Podle těchto bodů jsou transakce seřazeny a díky tomu dvoufázové zamykání zajišťuje uspořadatelnost.



Obrázek 6. Graf dvojfázového zamykání (zdroj: vlastní zpracování)

Dvojfázové zamykání dělíme na základní, to je popsáno výše a dále striktní, rigorózní a konzervativní. U striktního protokolu transakce drží množinu exkluzivních zámku do bodu, než bude potvrzena nebo zamítnuta. Rigorózní 2PL si ponechává nejen exkluzivní, ale i sdílené zámky do bodu potvrzení nebo zamítnutí transakce. Konzervativní neboli statický 2PL zajistí přidělení všech potřebných zámku jedné

transakci najednou tzv. atomických způsobem. Na rozdíl od již zmíněných 2PL protokolů zde nemůže docházet ke vzniku uváznutí.

2.1.2 Parametry zamykání

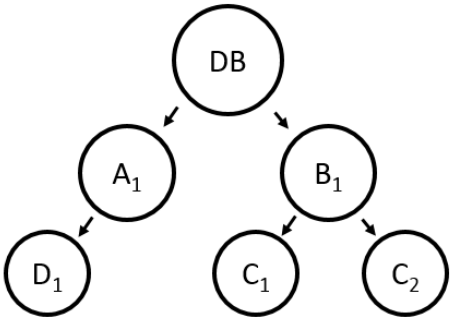
Ve vyspělejších systémech lze pro transakci nastavit ručně různé parametry užitých zámků. Ty mohou zlepšit výkonnost systému. Jedná se o *velikost zámku* neboli zrnitost. Dále je to *počet zámků*, které dostane transakce přidělena. *Eskalace zámku*, tedy jestli bude mnoho zámků na nižší úrovni nahrazeno jedním zámkem vyšší úrovně (například zámky na úrovni řádků, zámkem na celou tabulku). Posledním parametrem je *doba platnosti zámku*, která určuje maximální časovou lhůtu, po kterou může transakce držet zámek. Tato vlastnost může pomoci k vyřešení problému uváznutí.(7)

2.1.3 Zrnitost zámků

Databáze je celek tvořený daty, které jsou rozděleny do tabulek a ty do jednotlivých řádků a sloupců. Zámky mohou být užity pro uzamknutí právě jednoho pole, ale i pro větší celky, celou tabulku nebo databázi. Zrnitost zámků nám říká, jakou velikost má uzamknutý datový objekt. Zrnitost zámku je tím hrubší, čím větší položka byla uzamknuta.

V některých případech je výhodnější seskupit požadovaná data do jednoho většího celku a ten uzamknout. Snížíme tím počet celkově použitých zámků a usnadníme jejich správu. Tato metoda se nazývá *strukturní zamykání* (Multiple granularity locking). Zmiňovaný systém nám rozdělí databázi do pomyslného hierarchického stromu s několika uzly, které označují jednotlivé úrovně. Každý uzel může být v případě 2PL uzamknut, jak sdíleným tak i exkluzivním zámkem. Spolu s explicitně uzamčeným uzlem budou implicitně uzamknuti i jeho potomci, položky v úrovni, která spadá pod uzamčený uzel.

Mějme transakci T_A, která uzamkla uzel B_1. Druhá transakce T_B chce uzamknout záznam C_2, který spadá pod uzel B_1. Systém musí projet celý strom od kořene až k záznamu C_2 a zjistit, jestli požadovaný zámek nekoliduje s již existujícím zámkem. Jestli zjistí, že nějaký z vyšších uzlů je již uzamčen, transakce TB musí být opožděna.



Obrázek 7. Uzly strukturované zamykání (zdroj: vlastní zpracování)

Strukturní zamykání rozlišuje navíc další tři druhy zámků: *intenční sdílený, intenční exkluzivní, sdílený a intenčně exkluzivní zámek*. Ty upravují uzamčení potomků zamčeného uzlu.

U *intenčně sdíleného* (Intention Shared) jsou potomci zamčeného uzlu uzamčeny pouze sdíleným zámkem. Naopak při *tntenčně exkluzivním* (Intention Exclusive) zámku budou potomci zamčeného uzlu zamknuty exkluzivním zámkem. Poslední možný zámek, *sdílený a intenčně exkluzivní* (Shared and Intention Exclusive), tyto dva kombinuje. Všechny záznamy uzlu jsou uzamčeny explicitně sdíleným zámkem, ale u některých objektů je použit i exkluzivní zámek.

Je žádoucí, aby byl transakcím přidělen zámek s optimální zrnitostí. Dlouhotrvající transakce, které budou přistupovat k mnoha položkám, bude umožněno použití hrubšího zámku, který uzamkne například celou tabulku, ve které se dané položky nacházejí. Je to efektivnější, než kdyby transakce musela zažádat o zámek na každé pole samotné. Kratší transakce naopak budou více využívat jemnější zámky, protože nepracují s tolika položkami a nemusí zbytečně blokovat velké celky databáze.

2.1.4 Problém uváznutí (deadlocks)

Při využití zámků zamezíme výskytu již zmíněných problémů souběžnosti. Ale stále se může objevovat problém uváznutí neboli deadlocks. Zmíněný stav nastává, když si transakce vzájemně uzamknou záznamy, se kterými potřebují pracovat a následně jedna na druhou čekají, až jim bude záznam odemknut.

Příklad uváznutí je popsán v tabulce č. 10. Transakce T_A uzamkla exkluzivním zámkem záznam Q a transakce T_B uzamkla záznam B, aby transakce T_A mohla pokračovat, čeká na odemčení záznamu Z, který drží pod zámkem transakce T_B, ta naopak čeká na uvolnění záznamu Q.

Tabulka 11. Příklad uváznutí transakcí (zdroj: zpracováno podle 3)

T_A	T_B
LOCK (Q) EXCLUSIVE	
	LOCK (Z) EXCLUSIVE
LOCK (Z) EXCLUSIVE	LOCK (Q) EXCLUSIVE
wait	wait
wait	wait

DBMS dokážou tyto situace identifikovat a zamezit jim. Využívají k tomu určité časové lhůty pro udělení zámku jedné transakci. Po vypršení času musí transakce zámek pustit. Také může probíhat analýza transakcí před jejich spuštěním, jestli nemůže k uváznutí dojít. Tato možnost je ale systémově náročná a nepoužívá se tak často. Finálním řešením, kdyby bod uváznutí nastal, je zrušení jedné z problémových transakcí a jejím opětovným spuštěním, takže uživatel problém nepocítí.

2.2 Nezámkové protokoly

Kromě zámkového protokolu rozeznáváme i metody, které zámky nevyužívají. Ty se v distribuovaných DBMS ale tolik nevyskytují. Oba popisované způsoby se řadí do kategorie optimistických metod, jež nepředpokládají velký výskyt konfliktů souběžnosti.

2.2.1 Metoda časových značek (Timestamp ordering)

K zajištění uspořádatelnosti je zde využíváno hodnoty časové značky, kterou je každá transakce označena. Ty jsou generovány v okamžiku iniciace transakce podle aktuálního systémového času, nebo inkrementací logického čítače. Dojde-li ke konfliktu souběžného přístupu, bude starší transakce, ta s menší hodnotou časové značky, mít větší prioritu a bude vykonána dříve. Naopak druhá konfliktní transakce bude navrácena a bude ji přidělená nová časová značka při restartování.

U tohoto protokolu může docházet k *stárnutí* transakcí (starvation). To nastane, když bude docházet k častým konfliktům a časově náročnější transakce bude opětovně restartovaná. Metoda časových značek je ale odolná vůči uváznutí.

2.2.2 Validační metoda

Tato metoda využívá pro řízení souběžného přístupu dvě nebo tři fáze podle toho, zda transakce požaduje pouze čtení nebo i aktualizaci dat. První z nich je *čtecí fáze*, nastává v bodě, když se spustí transakce. V této fázi ji je dovoleno načítat z DB hodnoty

polí, uchovávat a upravovat je v lokálním úložišti, ke kterému má přístup pouze tato transakce. Dále nastává *validační fáze*. V této fázi systém vyhodnotí, zda prováděné aktualizace nenaruší uspořádatelnost. A jestli může být vyhodnocena jako úspěšná. Pro transakce pouze s akcemi čtení se překontroluje, zda načtené hodnoty jsou stále aktuální. Pro transakce, které požadují i aktualizaci polí, je ověřeno, zda změny ponechají databázi v konzistentním stavu. V opačném případě bude transakce zrušena a znovu spuštěna. Poslední fází je *zápis*, který nabývají pouze transakce s akcemi modifikace. Nastává v bodě, kdy je transakce schválena a její změny budou aplikovány z lokálního úložiště do databáze.

2.3 Multiversion Concurrency Control

Již zmíněné metody řízení souběžnosti zajišťují souběžnost pomocí zpožďování nebo rušení transakcí. Užitím MVCC (více verzové řízení souběžnosti) schématu se při zápisu WRITE (Q) vytvoří nová kopie (snapshot) Q. Když transakce bude požadovat čtení READ (Q), systém rozhodne, která z verzí Q bude čtena.

2.3.1 MVCC časových značek

Tak jako v již popsaném protokolu založeném na časových značkách, i tady každá transakce obdrží při iniciaci unikátní značku. Při každém zápisu transakcí WRITE (Q) je vytvořena nová kopie Q, která kromě hodnoty Q obsahuje i W-Timestamp - časovou značku transakce, která tuto kopii vytvořila. A dále i R-Timestamp – nejvyšší značku transakce, která hodnotu Q četla. Nevýhoda tohoto protokolu spočívá v tom, že i při čtení se musí aktualizovat pole R-Timestamp. A při konfliktech transakce nečekají, ale jsou navráceny a restartovány. Multiverzová metoda časových značek nepodporuje v základní verzi kaskádovost a návratnost transakcí.

Jestliže transakce T_A požaduje čtení nebo zápis Q, systém vybere verzi Q_k s nejvyšší časovou značkou, která je zároveň ale menší nebo rovna hodnotě Timestamp T_A. Neboli při čtení bude vybrána posledně změněná verze. Když T_A vyžádá READ (Q), je ji navrácena hodnota z verze Q_k. Jestli T_A vyžádá WRITE (Q) a Timestamp(T_A) < R-Timestamp(Q_k), tak systém navrátí transakci T_A. Pokud bude Timestamp(T_A) = W-Timestamp(Q_k), hodnota Q_k se přepíše. V případě, že by Timestamp(T_A) > W-Timestamp(Q_k), vytvoří se nová verze Q, která obdrží Timestamp transakce T_A.

Staré, nepotřebné verze jsou smazány podle daných pravidel. Máme-li dvě verze, Q_k a Q_i, obě mají W-timestamp menší než je časová značka nejstarší transakce. Poté starší verze Q není potřeba a může být vymazána.

2.3.2 MVCC dvojfázové zamykání

Tento protokol kombinuje 2PL a MVCC dohromady. Kromě dvojfázového uzamykání se používají číselné značky, které jsou udělovány jednotlivým verzím dat. Při uzamknutí objektu Q exkluzivním zámkem, je zabráněno vytvořením jakéhokoliv dalšího zámku na Q. Transakce, které v tomto případě budou požadovat pouze sdílený zámek na Q, by musely v běžném 2PL vyčkat. Díky užití MVCC, bude při žádosti o exkluzivní zámek na Q vytvořena nová kopie Q a na té bude zámek udělen. Ostatní transakce mohou stále číst starou kopii Q. Po dokončení aktualizací na kopii Q se zvýší hodnota značky o 1. Novým transakcím bude k dispozici vždy verze Q s nejvyšší hodnotou značky. Staré verze jsou mazány podle stejných pravidel jako u multiverzového protokolu časových značek.

3 PostgreSQL

PostgreSQL je objektově-relační databázový systém, který je k volně k získání pod licencí open-source. Jedná se o jeden z nejrozšířenějších, pokročilých, multiplatformních RDBMS. Základy systému vznikly na Kalifornské univerzitě v Berkeley v roce 1988. O šest let později byl do systému implementován interpret jazyka SQL. Dnes se na vývoji podílejí lidé z celého světa a díky tomu patří PostgreSQL k nejvyspělejším open-source databázovým systémem. K popisování byla zvolená verze 9.5.

3.1 Implementace řízení souběžnosti v PostgreSQL

Každý zadaný příkaz jazyka SQL je proveden v samostatné transakci. Konzistenci dat zde zajišťuje MVCC a 2PL. Při každém příkazu je vytvořen snímek dat a tím umožňuje souběžný běh transakcí.

Každý řádek v DB obsahuje políčko s ID transakce, která jej vytvořila, a políčko s ID transakce, která ukončila jeho platnost. Nové transakci je pak uděleno ID podle čítače. Řádek je viditelným, jestliže jeho ID tvořící transakce patří potvrzené transakci, nebo té, která právě neprobíhá, nebo je nižší než čítač. Zároveň musí platit, že ID expirující transakce je prázdné, bude vyšší než stav čítače na začátku dotazu, nebo ID odpovídá právě probíhající transakci. Jestli je prováděna aktualizace, systém vytvoří novou verzi řádku, ale zároveň ještě bude držet v paměti starou verzi. Tyto verze a již dávno expirované řádky se musí pravidelně vymazávat a k tomu slouží příkaz VACUUM, který může být použit na úrovni tabulky i celé DB.

3.2 Zámky v PostgreSQL

PostgreSQL využívá zámky, jak na úrovni řádku, tak i na úrovní celé tabulky a stránkové zámky. Systém automaticky přiděluje a spravuje zámky, ale mohou být vyvolány i příkazem LOCK. Uzamykání funguje v jednotlivých fází tak, jak je popsáno u 2PL. Každá transakce zažádá o zámek, který po přidělení bude držet až do svého ukončení. Existuje několik různých typů zámků, které se liší hlavně tím, se kterými zámky nastane konflikt, když budou použity na jednom objektu. Správce zámků ale neobsahuje prevenci uváznutí. Dojde-li k uváznutí, systém jej detekuje a jednu z transakcí zruší a její již proběhlé změny navrátí do původního stavu.

Zámky na úrovni tabulky

ACCESS SHARE – Tento zámek je vytvářen při akci, když je žádáno pouze čtení na tabulce. Díky tomu je slučitelný s většinou ostatních zámku vyjma ACCESS EXCLUSIVE, se kterým je v konfliktu.

ROW SHARE – Zámek je použit při příkazu SELECT FOR UPDATE a SELECT FOR SHARE. Ty uzamykají vybrané řádky a zabraňují modifikaci nebo vymazání daných řádku ostatními transakcemi, dokud ta probíhající neskončí.

ROW EXCLUSIVE – Zámek je vyvolán na cílovou tabulku příkazy UPDATE, DELETE a INSERT nebo jinými operacemi, které mají za cíl modifikovat data v tabulce. Spolu se zámkem ROW EXCLUSIVE na hlavní tabulce jsou získány ACCESS SHARE zámky na všech referenčních tabulkách.

SHARE UPDATE EXCLUSIVE – Při užití tohoto zámku je tabulka chráněna před změnami souběžného schématu a před příkazem VACUUM, který čistí a optimalizuje databázi. Vzniká užitím příkazů ANALYZE, CREATE INDEX CONCURRENTLY, ALTER TABLE VALIDATE (a jiné varianty) a VACUUM (kromě FULL).

SHARE – Zámek, který chrání tabulku před souběžnými změnami dat, je vytvořen při zadání příkazu CREATE INDEX (bez CONCURRENTLY).

SHARE ROW EXCLUSIVE – Jedná se o podobný zámek SHARE zámku, zabraňuje před souběžnými změnami dat, ale jedná se o jedinečný zámek. To znamená, že v jednom okamžiku může být držen pouze jednou operací. Vzniká užitím CREATE TRIGGER a různými variantami ALTER TABLE.

EXCLUSIVE – Tento zámek umožňuje souběžné užití pouze s ACESS SHARE zámky, uzamčená data mohou být pouze čteny jinými transakcemi. Je vytvářen příkazem REFRESH MATERIALIZED VIEW CONCURRENTLY.

ACCESS EXCLUSIVE – Tento zámek je nekompatibilní se všemi zámky. Zaručuje totiž transakci, která drží tento řádek, jedinečný přístup k zvolené tabulce. Je vyvolán příkazy, při kterých se například odstraňují (DROP TABLE) a vyprazdňují (TRUNCATE) tabulky, ale i dalšími CLUSTER, REINDEX, REFRESH MATERIALIZED VIEW (bez CONCURRENTLY), VACUUM FULL a také některými

DDL příkazy ALTER TABLE. Také je použit jako výchozí zámek při zadání LOCK TABLE, když není specifikován daný zámek.

Tabulka 12. Kompatibilita zámků tabulek v PostgreSQL (zdroj: 14)

Zažádaný zámek	**Stávající zámek**							
	AS	RS	RE	SUE	S	SRE	E	AE
ACCESS SHARE								X
ROW SHARE							X	X
ROW EXCLUSIVE					X	X	X	X
SHARE UPDATE EXCLUSIVE				X	X	X	X	X
SHARE			X	X		X	X	X
SHARE ROW EXCLUSIVE			X	X	X	X	X	X
EXCLUSIVE		X	X	X	X	X	X	X
ACCESS EXCLUSIVE	X	X	X	X	X	X	X	X

Zámky na úrovni řádku

Kromě zámků celé tabulky, může transakce zažádat pouze zámek na vybrané řádky. Jedna transakce může na stejném řádku držet více konfliktních zámků. U dvou souběžně jdoucích transakcí k tomu nemůže dojít.

FOR UPDATE – Tento zámek je vyžádán příkazy DELETE a UPDATE. Vybrané řádky příkazem SELECT jsou uzamknuty s cílem je aktualizovat. Je zabráněno ostatním transakcím, aby tento řádek modifikovaly a vytvářely zde jiné zámky, dokud současná transakce, která zámek FOR UPDATE drží, neskončí.

FOR NO KEY UPDATE – Tato možnost uzamknutí je podobná FOR UPDATE zámku, jen jeho ochrana je nižší. Je získáván příkazy UPDATE.

FOR SHARE – Zámek zajišťuje sdílení řádku pro čtení, ale zabraňuje před akcemi UPDATE a DELETE.

FOR KEY SHARE – Ochranou nejnižší zámek, který je kompatibilní se všemi řádkovými zámky, kromě FOR UPDATE zámku. Zabraňuje tedy aktualizacím nebo odstranění řádku, které by zapříčinily změnu klíče.

Stránkové zámky

Řádky v databázi jsou ukládány do jednotlivých stránek po 8kb. Jednotlivé stránky můžeme také uzamknout. Rozlišujeme dva typy zámku SHARE a EXCLUSIVE, které jsou využity pro kontrolu přístupu do sdílené paměti. Tyto zámky jsou přidělovány automaticky a jsou uvolněny hned po aktualizaci nebo načtení řádku z dané stránky.

Poradenské zámky

PostgreSQL nabízí aplikacím vytvářet i vlastní zámky. V tomto případě je ale na samotné aplikaci, zda tyto zámky použije správně. Mohou být uděleny na úrovni transakce, která až skončí, zámek uvolní a na úrovni sezení (session), kdy je zámek držen, dokud sezení nebude ukončeno. Jestliže při tomto sezení bude transakce navrácena zpět, bude zámek stále zachován. Když se vyskytnou žádosti o zámek na úrovni transakce a sezení se stejným identifikátorem budou navzájem vyrušeny. Pokud už je daný zámek již držen a zažádá o něj transakce nebo sezení, musí počkat, dokud zámek nebude uvolněn.

3.3 Izolační úrovně

PostgreSQL podporuje všechny čtyři izolační úrovně dle standartu SQL. Nejnižší úroveň READ UNCOMMITTED se zde ale chová stejně jako READ COMMITED a proto není v systému implementovaná. Další odlišností od standartu je zamezení fantomového čtení při použití REPEATABLE READ. Výchozí izolační úrovní, která je pro transakce použita, je READ COMMITTED. Úroveň můžeme ale změnit pomocí příkazu SET TRANSACTION.

READ COMMITTED – Při spuštění transakce je vytvořen snapshot databáze, obsahující změny potvrzené před jejím startem. Kromě toho dotaz v transakci vidí i změny, které byly provedené v rámci této transakce, ale nejsou ještě potvrzené. To může zapříčinit stav, kdy dva dotazy jedné transakce mohou navrátit jiné data.

REPEATABLE READ – Transakce probíhající na této úrovni vidí jen data, která byla potvrzená před spuštěním transakce. Na rozdíl od READ COMMITTED vidí celá transakce stále stejné hodnoty. Úroveň zabraňuje výskytu anomálií souběžnosti, ale nezaručuje uspořádatelnost, proto aplikace musí být připraven k navrácení transakce, například když dojde k paralelní aktualizaci řádku dvěma transakcemi.

SERIALIZABLE – Nejstriktnější úroveň izolace, která zajistí sériové zpracování transakcí místo paralelního. Pro aktuální transakci jsou všechny ostatní běžící transakce

skryté. Tato úroveň zamezuje třem anomáliím souběžnosti ANSI standartu. Od verze 9.1 používá PostgreSQL k implementaci této úrovně techniku Serializable Snapshot Isolation. Využívá snapshotů, aby zajistila všem akcím čtení jedné transakce stejnou konzistentní kopii databáze. Na počátku transakce bude provedena privátní kopie databáze, se kterou transakce bude pracovat.

Tabulka 13. Izolační úrovně v PostgreSQL (zdroj: 14)

Izolační úroveň	Anomálie		
	Špinavé čtení	Neopakovatelné čtení	Fantóm
READ COMMITTED	nemožné	možné	možné
REPEATABLE READ	nemožné	nemožné	nemožné
SERIALIZABLE	nemožné	nemožné	nemožné

4 Řízení souběžného přístupu v jiných RDBMS

K porovnání řešení řízení souběžného přístupu PostgreSQL jsem zvolil další dva nejrozšířenější relační databázové systémy, které jsou také multiplatformní a nabízí pokročilé funkce k zpracování dat. Je to komerční řešení Oracle, jehož první verze vznikla v roce 1978 a open-source systém MySQL. Ten byl poprvé vydán v roce 1995 a dnes je spravován společností Oracle. Oba systémy pracují s dotazovacím jazykem SQL, ORACLE navíc ještě nabízí vlastní rozšíření dotazů PL/SQL. U obou systému je popisována poslední dostupná verze, Oracle 12 a MySQL 5.7.

4.1 Oracle

V tomto systému je zajištěna konzistenci dat a správné řízení transakcí pomocí MVCC a 2PL. Multiverzní řízení souběžnosti využívá k označení transakcí System Change Number (SCN), které je přidělováno databázi. Toto číslo může nabýt maximální hodnoty 2^{48}, pak dojde k vynulování. Tento systém podporuje konzistentní čtení jak na úrovni transakce, tak i na úrovni příkazu. To znamená, že každému příkazu v transakci jsou k dispozici stejná data ze stejného okamžiku, ve kterém transakce započala.

4.1.1 Zámky

Systém Oracle používá dvojfázové zamykání, vyskytují se zde tedy exkluzivní a sdílené zámky. Uzamykání a odemykání je zajištěno automaticky systémem, ale i samotný uživatel nebo aplikace může manuálně řídit zámky. Zámky mohou uzamknout jak jeden řádek, tak i celou tabulku. Rozlišujeme zde tři druhy zámků, jsou to DML, DDL a interní zámky.

Zámky DML (Data Manipulation Language) jsou získávány příkazy SELECT, INSERT, UPDATE, MERGE a DELETE. Zajišťují souběžný běh akcí, které modifikují databázi. Dalším druhem jsou *zámky DDL* (Data Definition Language), které budou přiděleny například příkazům CREATE, ALTER, DROP a TRUNCATE. Ty chrání strukturu tabulek, či pohledů před změnou nebo smazáním dat, na které je odkazováno. Exkluzivní DDL zámek znemožňuje získání jiného zámku. *Interní zámky* chrání vnitřní a paměťové struktury databáze. Jsou pouze pod kontrolou systému a uživatel do těchto zámků nemůže zasahovat.

4.1.2 Izolační úrovně

Ze čtyř základní izolačních úrovní, které jsou definovány SQL standardem, jsou přímo podporovány v systému Oracle jen dvě - READ COMMITTED, SERIALIZABLE. Dále pak nabízí vlastní izolační úroveň READ ONLY, která se vymyká tomuto standartu. V tomto systému není na žádné z úrovních možný výskyt špinavého čtení tak jako u PostgreSQL. Oba systémy také podporují snapshot isolation, oproti Oraclu verze užitá PostgreSQL garantuje plnou uspořádatelnost.

Výchozí nastavenou úrovní je READ COMMITTED, ta je určena pouze pro čtení dat. Transakce má k dispozici pouze data, která byla potvrzena na začátku prvního dotazu. Může zde dojít ke konfliktu, když transakce chce číst ještě nepotvrzená data. V tomto případě musí počkat, dokud nebude transakce modifikující tento řádek ukončena. Na této úrovni je možný výskyt neopakovatelného čtení a fantomů.

Nejstriktnější úrovní je SERIALIZABLE, která poskytuje nejvyšší úroveň izolace. Nevyskytují se zde anomálie souběžnosti. Transakce pracuje v prostředí, které se ji jeví, jakoby neprobíhaly žádné další modifikace v databázi. Je zajištěno, aby všechna data, která byla už jednou přečtena, měly při druhém čtením v rámci téže transakce stejnou hodnotu jako na počátku, nebo hodnotu již změněnou touto transakcí. Pokud se jiná transakce bude snažit modifikovat data, která jsou používána transakcí na této úrovni, bude při pokusu zrušena s chybou ORA-08177: nelze získat souběžný přístup pro tuto transakci. Této úrovni odpovídá v PostgreSQL úroveň REPEATABLE READ.

Úroveň READ ONLY je podobná úrovni SERIALIZABLE. Jak z názvu vypovídá, úroveň je určena pouze pro čtení dat a transakcím není povoleno provádět modifikace. Načtená data jsou vždy konzistentní k bodu, kdy transakce byla spuštěna. Je využíván místo standartní úrovně REPEATABLE READ.

4.2 MySQL

V tomto systému můžou být databáze nebo i jednotlivé tabulky uloženy v jednom z mnoha formátu na ukládání dat. Každý formát má své využití a vlastnosti. Nejpoužívanější je MyISAM, který je nastaven defaultně, a InnoDB. Podrobný popis všech dostupných formátů v aktuální verzi je popsán v MySQL referenční příručce v kapitole 15. Alternative Storage Engines.

MyISAM představuje jednoduché úložiště, které je rychlé, nabízí možnost fulltextových indexů. Nezaručuje ale integritu dat, nepodporuje transakční přístup a nenabízí funkci navrácení. Proto jeho využití nalezneme jako úložiště pro efektivní čtení dat. Naopak *InnoDB* zajišťuje pokročilejší způsob ukládání, který transakce spolu s navrácením při pádu podporuje. Řízení souběžného přístupu může v InnoDB využívat i zámky řádků, zatímco u MyISAM můžeme narazit pouze na zámky na úrovni tabulky.

Jedna transakce může obsahovat akce, které budou přistupovat k tabulkám, které jsou uloženy i v jiných formátech nepodporující transakce. Pokud ale transakce neskončí potvrzením, tak budou provedené změny navráceny pouze v tabulkách, které jsou uloženy ve formátu podporující transakce. V tabulkách transakce nepodporující budou změny trvalé. (8)

4.2.1 Souběžný přístup v InnoDB

Jestli chceme využívat transakce v MySQL, musíme zrušit možnost automatického potvrzení všech příkazů pomocí SET AUTOCOMIT = 0. Po tomto nastavení bude každý příkaz vztahující se na tabulku, která transakční přístup podporuje, proveden jako transakce. Nebude tedy defaultně bez ohledu na chyby potvrzen, ale bude muset skončit buď možností COMMIT nebo ROLLBACK.

V InnoDB nalezneme řízení transakcí pomocí 2PL v kombinaci s MVCC. Stejně jako PostgreSQL je zde podporována serializovatelná izolovanost transakcí. Všechny příkazy v jedné transakci mají k dispozici stejný snapshot. Pro případ navrácení MVCC ukládá staré hodnoty jednotlivých řádků k zajištění souběžného chodu a pro možnou funkci navrácení. Hodnoty jsou uloženy v *rollback segmentu*, ze kterého jsou poté načteny, když je potřeba transakci navrátit. Ke každému řádku jsou přidávány tři hodnoty. Pole DB_TRX_ID obsahuje ID transakce, která naposledy tento řádek měnila. DB_ROLL_PTR původní hodnoty řádku, než byl poměněn. Třetím polem je DB_ROW_ID, kde je uloženo ID řádku pro případ, kdyby byly některé řádky smazány nebo přidány.

Při odstranění nebo aktualizaci dat nebude změna fyzicky na disk ihned zapsána. Dojde k tomu v okamžiku až MySQL zahodí záznam v logu, který byl vytvořen při provádění této akce. Tato operace se nazývá purge. Jestliže budeme postupně vymazávat velké množství řádků, může se stát, že log bude nabývat na velikosti, a dojde k zpomalení databáze.

4.2.2 Zámky v InnoDB

Správce souběžného přístupu přiděluje v InnoDB zámky automaticky. Zámky mohou být na úrovni řádků nebo tabulek. Nepodporuje ale eskalaci zámku. Nalezneme zde i několik speciálních zámků, které se v již popisovaných DBMS nevyskytují. První z nich je *gap lock*. Ten například při výběru dat SELECT * FROM table WHERE Q>10 uzamkne nejen všechny řádky s hodnotou Q větší než 10. Zabrání rovněž, aby jiná transakce nepřidala do této tabulky nové řádky, které by měly hodnotu větší než 10. A také tomu, aby hodnota stávajících polí nemohla být změněna na číslo větší než 10.

Dalším zámkem je *record lock*, který například při příkazu SELECT * FROM table WHERE Q=10 brání všem ostatním transakcím modifikovat nebo mazat pole, u kterých je hodnota Q rovna 10. Posledním zámkem, pouze se vyskytující v InnoDB je *next-key lock*. Ten kombinuje gap lock a record lock. Uzamyká řádky podle jejich ID. Zámek bude přidělen nejen vybraným řádkům a mezerám, které se ve výběru vyskytují, ale i hodnotám, které jsou nižší nebo vyšší než oblast výběru a v praxi v tabulce ještě neexistují. Next-key lock tedy zabrání vložení nových řádků a blokuje výskyt fantómů.

Jestliže nastane uváznutí transakcí, je tento stav systémem odhalen a jedna z transakcí je navrácena a restartovaná. K zrušení je vybírána méně náročná, kratší transakce. InnoDB nepředpokládá výskyt zámků a tedy i uváznutí, jestli máme nastaveno AUTOCOMMIT = 1 a využijeme ruční příkaz LOCK TABLES.

4.2.3 Izolační úrovně

InnoDB podporuje všechny čtyři izolační úrovně dle standartu ANSI. Jejich vlastnosti jsou stejné, jak je popsáno v kapitole 1.2.3. Automaticky nastavenou úrovní je REPEATABLE READ. Díky užití MVCC je v systému podporováno konzistentní čtení dat bez nutnosti zámků. To znamená, že ostatní transakce mohou k těmto polím přistupovat. Při užití výběru SELECT v defaultní izolační úrovni, budou všechny akce čtení dat v transakci přistupovat k snapshotu, který byl použit první akcí SELECT. Tento snapshot obsahuje pouze potvrzené změny před počátkem této transakce. V úrovni READ COMMITTED je každému čtení v rámci transakce zpřístupněn nový snímek databáze.

5 Školní informační systémy

Informační systém představuje uspořádaný celek prvků, činností s jejich vlastnostmi a vztahy, který zpracovává informace. Jedná se o soubor lidí, prostředků a metod, které zabezpečují sběr, přenos, uchování i zpracování dat a následné navrácení informací pro potřebu uživatelů v tomto systému. (9) Podle této definice můžeme popsat školní informační systém jako „soubor lidí, metod a technických prostředků zajišťujících sběr, uchování, analýzu a prezentaci dat určených pro poskytování informací v oblasti vzdělávání. Umožňují výrazně zefektivnit fungování celé vzdělávací instituce.“ (Dostál, 2011)

Školní informační systémy jsou velice blízké podnikovým informačním systémům, o kterých se více můžete dozvědět v knize od Zdeňka Molnára: Podnikové informační systémy. Oba typy systému jsou určeny pro jednodušší, efektivnější a bezpečnější práci s informacemi. Ty jsou zadávány do systému uživateli. Informace jsou dále archivovány, zpracovávány a podle potřeby navráceny uživatelům. ŠIS také zajišťují komunikaci mezi školou a jinými orgány, nebo rodiči. Dnes je ŠIS nepostradatelnou součástí nejen každé školy, ale i mateřských školek nebo jiných výchovně-vzdělávacích institucí. (5)

5.1 Řešení na trhu

Na českém trhu je k dispozici několik ŠIS. Možná řešení by se dala rozdělit na ty, které požadují pro běh určitý databázový systém na straně školy, a na ty, které jsou provozovány na serverech vývojářských firem.

Jestli ŠIS z první skupiny budou podporovat pouze zpoplatněný RDBMS, může se celková pořizovací cena navýšit. Nutno také počítat se správou těchto systémů na straně školy. Do této skupiny se například řadí IS Bakaláři, který je jedním z nejrozšířenějších řešení. Podporuje pouze databázový systémem MS SQL server 2008 a výše. S podporou MySQL serveru skončil u verze 10. Dalším ze zástupců je *IS SAS*, ten je kompatibilní s open-source RDBMS Firebird. Oba systémy jsou určeny pro základní, střední a vyšší odborné školy. Jednou z možností pro vysoké školy je *IS STAG*, vyžadující pouze RDBMS Oracle, minimálně Standart One Edition.

U druhé skupiny odpadají pro školu vyšší hardwarové a softwarové nároky pro provoz. Pro přístup k těmto systémům stačí pouze internetové připojení a webový prohlížeč. Jmenovitě do této skupiny patří například *iŠkola.cz, Etřídnice, Škola OnLine*.

5.2 ŠIS z pohledu databáze

Navenek se ŠIS uživatelům tváří jako program nebo webová aplikace, ve které jsou jejich možnosti akcí omezené uživatelským rozhraním na určité moduly, ve kterých mohou provádět jen dané akce. Z pohledu databáze se ŠIS skládá z několika skupin uživatelů a každý modul je vázán na skupinu vzájemně propojených tabulek. Každá skupina uživatelů má přístup pouze k určitým modulům neboli tabulkám. I když by se uživatel neměl přes rozhraní ŠIS dostat do modulu a k informacím, ke kterým nemá přístup, je vhodné, aby u každé tabulky bylo nastaveno, zda mohou uživatelé v této tabulce číst nebo i provádět změny. Jednotlivé akce uživatelů, které jsou zasílány DBMS přes aplikaci, by měly být zpracovávané jako transakce, aby nedošlo k porušení integrity dat. A proto je potřeba, aby zvolený DBMS, na kterém ŠIS funguje, podporoval transakční přístup.

5.2.1 Skupiny uživatelů

Každý z uživatelů je v databázi evidován a rozřazen do jedné ze skupin. Ke každé skupině se vztahují pravomoci, které upravují možnosti uživatelů dané skupiny, které z modulů mohou využívat a jaké akce v nich mohou vykonávat. Jmenujme základní uživatelské skupiny: žáci, pedagogičtí pracovníci, nepedagogičtí pracovníci a správce systému.

Hierarchicky nejnižší skupinou budou žáci. Ti budou moci přistupovat v modulech jenom k informacím, které se týkají jejich osoby, jejich třídy nebo se jedná o celoškolní informace. Ve většině modulech bude mít právy omezené akce pouze na čtení. Přes tento typ účtu mají mnohdy přístup do ŠIS i rodiče, pokud pro ně není vytvořeno samostatné přihlášení. Ti disponují velmi podobnými oprávněními jako žáci, proto je můžeme do této skupiny zahrnout.

Uživatelé patřící do skupiny pedagogičtí pracovníci jsou zaměstnanci školy. Nalezneme zde například učitele a vychovatele. Této skupině bude povolen přístup do většiny modulů, které se týkají nejen výuky ale interních záležitostí školy. Každý učitel je charakterizován třídami a předměty, které učí. Podle nich by měly být ve výsledku upraveny i jeho práva. V každé třídě je navíc stanoven třídní učitel a jeho zástupce. Tito uživatelé budou mít v rámci třídy větší pravomoci než ostatní učitelé. Může se jednat například o změnu údajů na kartě žáka.

Další skupinou jsou nepedagogičtí pracovníci. Jejich pravomoci se budou hlavně vztahovat na moduly interní správy školy, například evidence majetku, evidence žáků, správa přijímacích zkoušek. Pro každého člena mohou být práva přístupu jiná, záleží podle jejich náplně práce ve škole.

Správce systému nemusíme úplně chápat jako skupinu uživatelů, ale spíše jako funkci, která je přidělena určitým osobám (například ředitel, jeho zástupce, samotný správce systému). Uživatel s rolí správce systému disponuje všemi pravomocemi, tady je hierarchicky nejvýš. Samotná osoba, která je zodpovědná za správu školní informační sítě, má přístup i přímo do DB ŠIS. Dle potřeby ji může i ručně upravovat, mazat nebo zálohovat.

Správce systému má právo využívat přímo v DB jako jediný nejvyšší zámky: SHARE UPDATE EXCLUSIVE, SHARE, SHARE ROW EXCLUSIVE, EXCLUISIVE a ACCESS EXCLUSIVE, který blokuje přístup všem ostatním zámků a může provádět nutnou správu databáze. Jedná se například o tvorbu nebo správu již existujících tabulek. Důležité je také vymazávání již nepotřebných kopií dat, které byly vytvořeny pomocí MVCC příkazem VACUUM. Je nutné jej pravidelně zadávat pro vyčistění paměti a optimalizaci DB.

5.2.2 Jednotlivé moduly

Jak bylo uvedeno výše, každá skupina má přístup do některých modulů a v nich má ještě omezené akce, které může vykonat. Podle typu těchto akcí, bude RDBMS automaticky rozhodovat, jaké vlastnosti transakce, vykonávací tyto příkazy, bude mít. Tedy na jaké izolační úrovni bude akce vykonávána a jaké zámky využije. Tyto vlastnosti můžeme i v PostgreSQL definovat manuálně.

Jelikož PostgreSQL podporuje funkci konzistentního čtení, transakce obsahující pouze akce čtení nebudou způsobovat konflikty souběžnosti a jejich správa bude jednodušší. Transakce od uživatelů, kteří mohou v rámci modulu pouze zobrazovat data, by měly mít standardně nastaveny jeden ze sdílených zámků (například ACESS SHARE, ROW SHARE) a využívat izolační úrovně READ COMMITTED nebo REPEATABLE READ.

Tabulka 14. Nejběžnější nabídka současných ŠIS (zdroj: 16)

Organizační jednotky	Přijímací řízení	Výuka	Adminis-trativa	Účet-nictví	Technická správa	Státní správa	Rodiče a veřejnost
Přijímací řízení	C		U			U	U
Evidence žáků		C	U			U	U
Klasifikace žáků		C					U
Informace o vyučování		C					U
Maturitní zkoušky		C					U
Informace o absolventech		C					U
Rozvrh hodin		C					U
Suplování		C					U
Školní akce		C					U
Zápisy z porad	U	C	U		U	U	U
Pedagogická dokumentace		C	U			U	
Školní knihovna		U	C				
Evidence pracovníků			C	U		U	
Evidence majetku			C	U		U	
Správa řízení			C			U	
Směrnice a provozní řády	U	U	C	U	U		
Ekonomická agenda			U	C		U	
Informace o škole			C			U	U
Operativní informace pro žáky			U				
Operativní informace pro zaměstnance			U				
Informace pro rodiče							U
Informace pro veřejnost							U
Informace pro státní správu						U	

Vysvětlivky: U – využívání dat, C – vytváření dat

Pokud je skupině uživatelů dovoleno v modulu i data vkládat, měnit a mazat, transakce budou obdařeny exkluzivním zámkem ROW EXCLUSIVE, který je automaticky vyvolán právě s příkazy UPDATE, DELET a INSERT. V případě modifikací bude transakce probíhat izolační úroveň SERIALIZABLE.

Klasifikace žáků

Modul klasifikace žáků je určen pro udělování známek žákům. Žáci budou mít v tomto modulu pouze práva k zobrazení svých zámek. Jelikož se jedná o pouze akci READ, bude jim přidělen zámek sdílený zámek ACCESS SHARE. Transakce bude probíhat na nejnižší úrovni READ COMMITTED. Pedagogičtí pracovníci budou moci zobrazovat hodnocení všech zámků. Známky z předmětů žáků, které učí, budou moci i zapisovat, editovat nebo mazat. Záznamy všech žáků budou moci uzamknout ACCESS SHARE zámkem. Tabulky se známkami žáků, které učí, budou moci uzamknout ROW SHARE nebo ROW EXCLUSIVE zámkem. V případě editace pouze jedné ze známek, kterou zadali, může být využito zámku onen řádek FOR NO KEY UPDATE. K dispozici jim budou všechny tři izolační úrovně.

Správa absence

Žáci mohou pouze číst záznamy o své absenci. Tak jako v předešlých modulech budou jejich zadané akce provedeny na úrovni READ COMMITTED pod zámkem ACCESS SHARE nebo FOR SHARE a FOR KEY SHARE. Učitelé budou moci zapisovat absenci ve všech hodinách například z důvodu suplování. Editovat a mazat absence mohou pouze v hodinách, které sami zapsali. Třídní učitelé budou moci editovat a mazat (neboli omlouvat) absence u všech žáků jeho třídy.

Evidence žáků

Tento modul obsahuje seznam všech žáků. Každý žák má svoji vlastní kartu, na ní jsou k dispozici osobní informace, ale i jejich průběžná klasifikace. Každý uživatel ze skupiny žáků má práva zobrazit pouze svoji vlastní kartu. Jestli je v ŠIS rozlišena uživatelská skupina rodičů, tak jim bude přiděleno právo měnit osobní informace žáků nebo kontaktní informace. Transakce zadané touto skupinou budou moci obdržet nejen zámky ACCESS SHARE, FOR SHARE, FOR KEY SHARE, které v tomto modulu pouze dostanou žáci, ale i ROW SHARE a ROW EXCLUSIVE.

Skupina pedagogických pracovníků bude mít k dispozici karty všech žáků, ale pouze třídní učitelé a jejich zástupci budou moci upravovat údaje evidovaných žáků ze své třídy. Transakce vycházející z jejich příkazu mohou získat stejných zámků jako u skupiny rodičů. K seznamu žáků budou mít také přístup administrativní pracovníci školy, kteří budou mít právo čtení i zápisu pro všechny žáky.

Karta žáka je propojena nejen s osobními informacemi uvedenými v evidenci, ale i s moduly klasifikace a absence žáka. V tomto modulu může být také zadán tisk seznamu žáků, nebo vysvědčení. Uživatelům, kteří tuto funkci mohou použít (administrativní a pedagogičtí pracovníci, vedení), bude v transakci přidělena nejvyšší izolační úroveň SERIALIZABLE. Tím se zabezpečí, aby při tisku bylo pracováno se stálou kopii databáze, a jiným akcím bude zamezeno provádět při výkonu této akce změny.

Přijímací řízení

Do tohoto modulu bude mít přístup veřejnost, vedení školy a administrativní pracovnici. Vedení a administrativa má v rámci modulu stejná práva. Může zapisovat, měnit a mazat data. Budou využívat zámků ROW SHARE a ROW EXCLUSIVE. Pouze tyto dvě skupiny mají přístup k osobním informacím uchazečů o studiu. Pro ostatní jsou informace o výsledcích přijímacího řízení evidovány pouze pod ID číslem každého uchazeče. Proto mohou být tyto seznamy volně přístupné veřejnosti na webu školy. Ty budou automaticky generovány, nebo mohou být filtrovány například dle zadaných kritérií (termín konání zkoušky). Tyto akce budou probíhat na úrovni REPEATABLE READ, aby se ke koncovému uživateli nedostaly nepotvrzené data z jiných transakcí.

Závěr

V databázových systémech je využíváno několik různých metod k zajištění souběžného přístupu transakcí. Užití stejné metody se může ale v jednotlivých systémech ve finále projevit odlišnými vlastnostmi. PostgreSQL a Oracle podporují transakční přístup přímo a každý zadaný příkaz je proveden jako transakce. U MySQL je potřeba zvolit formát ukládání dat podporující transakce.

Ve všech třech systémech je implementováno MVCC spolu s izolací snapshotů. Také používají metodu dvoufázového zamykání řádků a tabulek. Všechny systémy rozlišují základní rozdělení na exkluzivní a sdílené zámky, ale celkový výčet zámku každého systému se liší. PostgreSQL nabízí několik zámků pro úroveň tabulky, řádku, stránkové zámky na pozadí a poradenské zámky, které si uživatel může sám nadefinovat. Oracle rozlišuje zámky podle typu příkazu na DML, DDL a na interní zámky, které kontroluje systém. Typ úložiště InnoDB v MySQL nabízí i tři speciální zámky, ze kterých nejpřísnější ochranu zajišťuje next-key lock, který blokuje vkládání nových řádků a zamezuje výskytu fantómů. PostgreSQL spolu s MySQL podporují na rozdíl od Oraclu skutečnou uspořádatelnost pomoci SSI. Všechny tři systémy splňují standarty ANSI/ISO SQL pro souběžný přístup.

Při zavedení ŠIS musí škola počítat i s pořízením daného databázového serveru, který tento systém podporuje. Existují i řešení, běžící na straně poskytovatele. Škola se pak nemusí zabývat funkční stránkou ŠIS. Ani jedno z uvedených řešení nepodporuje RDBMS PostgreSQL a pouze IS SAS využívá bezplatný databázový systém FireBird.

Školní informační systém je soubor modulů a uživatelů. Každý uživatel má omezené akce READ a WRITE v tabulkách, spadajících pod modul, do kterého smí tento uživatel přistupovat. Akce uživatelů, které pouze čtou data, by měly být omezeny na izolační úrovně READ COMMITTED, nebo REPEATABLE READ a využívat zámků ACCESS SHARE, aby neblokovali ostatní transakce v systému. Jestliže uživatelé chtějí zapisovat data, budou jejich akce provedeny v transakcích na úrovni SERIALIZABLE pod zámkem ROW EXCLUSIVE. Úrovně a zámky jsou automaticky přidělovány řízením souběžného přístupu, ale můžeme je také definovat pro jednotlivý typ akcí ručně.

Tato práce měla za cíl přiblížit problematiku řízení transakcí v databázových systémech. Popisuje význam transakcí a jednotlivé metody a přístupy, kterým jsou zpracovávány. Dále zde bylo podrobněji popsáno řešení souběžného přístupu

v PostgreSQL a dalších dvou RDMBS. Získané poznatky byly využity u popisu fungování ŠIS z pohledu databáze. Ty by mohly například dále být využity pro tvorbu vlastního ŠIS.

Použitá literatura a zdroje

1) BERNSTEIN, Philip A a Eric NEWCOMER. Principles of transaction processing. 2nd ed. Burlington, MA: Morgan Kaufmann Publishers, c2009. Morgan Kaufmann series in data management systems. ISBN 1558606238.
2) BERNSTEIN, Philip A, Vassos HADZILACOS a Nathan GOODMAN.Concurrency control and recovery in database systems. Reading, Mass.: Addison-Wesley Pub. Co., c1987. ISBN 0201107155.
3) CONOLLY, Thomas, Carolyn E BEGG a Richard HOLOWCZAK.Mistrovství - databáze: profesionální průvodce tvorbou efektivních databází. Vyd. 1. Brno: Computer Press, 2009. ISBN 978-80-251-2328-7.
4) DATE, C. An introduction to database systems. 8th ed. Boston: Pearson/Addison Wesley, 2004. ISBN 0321197844.
5) DOSTÁL, Jiří. Školní informační systémy. Vyd. 1. Olomouc: Univerzita Palackého v Olomouci, 2011. ISBN 978-80-244-2784-3.
6) GARCIA-MOLINA, Hector, JEFFREY D. ULLMAN a Jennifer WIDOM.Database systems: the complete book. International ed. Upper Saddle River, NJ [u.a.]: Prentice Hall [u.a.], 2002. ISBN 0130980439.
7) GROFF, James R a Paul N WEINBERG. SQL: kompletní průvodce. Vyd. 1. Brno: CP Books, 2005. Programování (CP Books). ISBN 80-251-0369-2.
8) KOFLER, Michael a David KRAMER. The definitive guide to MySQL. 2nd ed. New York: Distributed to the book trade in the United States by Springer-Verlag, c2004. ISBN 1590591445.
9) MOLNÁR, Zdeněk. Podnikové informační systémy. Vyd. 2., přeprac. V Praze: České vysoké učení technické, 2009. ISBN 978-80-01-04380-6.
10) MOMJIAN, Bruce. PostgreSQL: praktický průvodce. Vyd. 1. Brno: Computer Press, 2003. ISBN 80-7226-954-2.
11) O'NEIL, Patrick a Elizabeth O'NEIL. Database--principles, programming, and performance. 2nd ed. San Francisco: Morgan Kaufmann Publishers, c2001. ISBN 1558604383.
12) SILBERSCHATZ, Avi, HANK F. KORTH a S. SUDARSHAN. Database system concepts. 5. ed., international ed. Boston, Mass. [u.a.]: McGraw-Hill, 2006. ISBN 007124476X.

13) WEIKUM, Gerhard. Transactional information systems: theory, algorithms, and the practice of concurrency control and recovery. San Francisco: Morgan Kaufmann Publishers, c2002. Morgan Kaufmann series in data management systems. ISBN 1-55860-508-8.

Internetové zdroje

14) Dokumentace PostgreSQL a wiki [online]. The PostgreSQL Global Development Group, 2016 [cit. 2016-04-07]. Dostupné z: http://www.postgresql.org/

15) MySQL :: MySQL 5.7 Reference Manual [online]. Oracle Corporation, 2016 [cit. 2016-04-07]. Dostupné z: http://dev.mysql.com/doc/refman/5.7/en/

16) NEUMAJER, Ondřej. Školní informační systémy. Metodický portál RVP.CZ - unikátní PROSTOR PRO UČITELE, sdílení zkušeností a spolupráci [online]. Praha: NÚV, 2016 [cit. 2016-04-07]. Dostupné z: http://clanky.rvp.cz/clanek/c/Z/8019/skolni-informacni-systemy.html/

17) Oracle Database Online Documentation 12c Release 1 (12.1) [online]. Oracle Corporation, 2016 [cit. 2016-04-07]. Dostupné z: https://docs.oracle.com/database/121/index.html

Seznam zkratek a termínů v anglickém jazyce

2PL	Two-Phase Locking
DB	Databáze
DBMS	Database Management Systém
DDL	Data Definition Language
DML	Data Manipulation Language
MVCC	Multiversion Concurrency Control
RDBMS	Relation Database Management System
SQL	Structured Query Language
SSI	Serializable Snapshot Isolation
ŠIS	Školní informační systém
WAL	Write Ahead Log

Commit	potvrzení transakce
Concurrency Control	řízení souběžného přístupu
Data Definition Language	příkazy pracující se strukturou DB objektů
Data Manipulation Language	příkazy pracující s daty
Database Management System	systém řízení báze dat
Deadlock	uváznutí
Checkpoint	kontrolní bod
Multiple Granularity Locking	strukturní zamykání
Multiversion Concurrency Control	správce souběžného přístupu pracující s více kopiemi databáze
Relation Database Management System	relační systém řízení báze dat
Rollback	navrácení změn transakce
Serializable Snapshot Isolation	snapshot isolation podporující uspořádatelnost
Snapshot	snímek neboli kopie databáze
Snapshot Isolation	technika MVCC, transakce vidí izolovaný snímek od svého počátku
Starvation	stárnutí transakce (opakované restartování)
Structured Query Language	standardizovány dotazovací jazyk
Timestamp	časová značka
Two-Phase Locking	dvojfázové zamykání
Write Ahead Log	metoda navrácení změn pomocí souboru událostí

Seznam tabulek

Seznam obrázků

ANOTACE

Jméno a příjmení:	Petr Mališ
Katedra:	Katedra informační a technické výchovy
Vedoucí práce:	Mgr. Jan Kubrický, Ph.D.
Rok obhajoby:	2016

Název práce:	Možnosti transakčního chování databázových systémů a jejích využití v modulech školních informačních systémů
Název v angličtině:	Options of transactional behavior of a database systems and their usage in modules of school information systems
Anotace práce:	Tato bakalářská práce se zabývá transakcemi v databázových systémech a jejich implementací do školních informačních systémů. První část popisuje metody používané k řízení souběžného přístupu a možnosti navrácení. Ve druhé části jsou popsány metody souběžného přístupu databázového systému PostgreSQL a porovnány se systémy MySQL a Oracle. V závěru jsou získané poznatky implementovány do školních informačních systémů.
Klíčová slova:	Transakce, řízení souběžného přístupu, RDBMS, školní informační systém
Anotace v angličtině:	This bachelor's thesis is dealing with transactions in database management systems and their implementation into school information systems. A first part of this work describes methods used for concurrency control and rollback functions. A following part describes concurrency control used in DBMS PostgreSQL and it is compared to other systems – MySQL and Oracle. In a conclusion gained observations are implemented into school information systems.
Klíčová slova v angličtině:	Transaction, concurrency control, RDBMS, school information system
Přílohy vázané v práci:	-
Rozsah práce:	41 s. (61 000 znaků)
Jazyk práce:	čeština