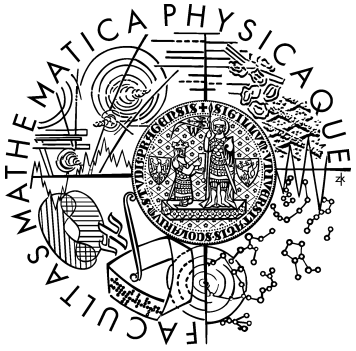
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Luboš MATÁSEK

Automatické generování logovacích triggerů v DB

Katedra softwarového inženýrství

Vedoucí práce: Mgr. Václav Matouš

Studijní program: Informatika

Studijní obor: Programování

2009

Poděkování

Děkuji svému školiteli Mgr. Václavu Matoušovi za vedení bakalářské práce a zároveň i za cenné rady a připomínky, které mi k ní přinášel.

Dále děkuji rodičům a babičce za morální i finanční podporu, kterou mi poskytovali během celého vysokoškolského studia. V neposlední řadě děkuji i přítelkyni Terezce za oporu, kterou mi byla v průběhu psaní této práce.

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 27. května 2009.

.....

Luboš Matásek

Obsah

Seznam obrázků

Seznam tabulek

Název práce: Automatické generování logovacích triggerů v DB
Autor: Luboš Matásek
Katedra: Katedra softwarového inženýrství
Vedoucí práce: Mgr. Václav Matouš
e-mail vedoucího: vaclav.matous@mff.cuni.cz

Abstrakt: Předložená práce se zabývá automatizovaným generováním logovacích triggerů. Praktickým výsledkem je aplikace Database Logger, která umožňuje jejich interaktivní generování a správu. Aplikace podporuje logování DML databázových operací (insert, update, delete) včetně update triggerů pro sloupce. Současná verze podporuje dvě databázové platformy: Oracle a PostgreSQL. Formát logu byl zvolen s ohledem na zachování maximálního objemu informace a jednoduché dotazování. Kromě univerzálního logování projekt umožňuje i používání historických tabulek, které mohou sloužit jako pohled na historická data v jedné konkrétní tabulce. Aplikace dále podporuje jednoduché dotazování nad logy i historickými tabulkami a export výsledků do XML.

Klíčová slova: databáze, trigger, logování, DML operace, XML

Title: An automatic generation of logging database triggers
Author: Luboš Matásek
Department: Department of Software Engineering
Supervisor: Mgr. Václav Matouš
Supervisor's e-mail address: vaclav.matous@mff.cuni.cz

Abstract: In the thesis we study automatic generation of logging database triggers. As the result an application called Database Logger for logging triggers management was developed. The application allows DML events logging (insert, update, delete) including column update triggers. Current version supports two database platforms: Oracle and PostgreSQL. The log format preserves the maximum of information value and brings the possibility of easy querying. Users can also create historical tables to produce historical data views. Application supports log and historical table querying and their export into XML.

Keywords: database, trigger, logging, DML events, XML

Kapitola 1

Úvod

Dnešní moderní aplikace využívají pro uložení dat ve velké míře databáze. A protože právě data jsou často nejcennější součástí celého systému, je potřeba přikládat stále větší význam jejich bezpečnosti. Bezpečností dat přitom nerozumíme pouze zabezpečení proti vnějšímu útoku, ale i zajištění vnitřní bezpečnosti systému. Uživatelé systému totiž mohou (často neúmyslně) na datech způsobit nenapravitelné škody.

Jednou z minimálních bezpečnostních politik může být například logování kritických databázových operací. Záznamy logu by měly obsahovat aspoň základní informace o provedených změnách, jako například o čase, autorovi a druhu provedené změny. V optimálním případě jsou zaznamenány přímo konkrétní změny na úrovni databázových sloupců.

Pro automatizované zaznamenávání databázových operací přímo na úrovni databáze se hodí tzv. *triggery*[1]. Při větším počtu tabulek v databázi je ruční vytváření logovacích triggerů bohužel poměrně neefektivní a zdlouhavá práce. Hlavní kostra triggeru je totiž pro všechny tabulky stejná a kód může být v ideálním případě generován automatizovaně.

1.1 Cíl práce

Cílem bakalářské práce je vytvoření interaktivní grafické aplikace pro automatizované generování logovacích triggerů. Aplikace bude podporovat logování aspoň ve dvou databázových platformách (pro pilotní implementaci byly zvoleny platformy *Oracle* a *PostgreSQL*). Logování půjde omezit na vybrané tabulky, resp. prováděné operace.

[1]Databázový trigger je procedurální kód, který je proveden jako odpověď na určitou událost v databázi (např. přidání, editaci nebo smazání řádku tabulky, přihlášení uživatele).

Formát logu není předem stanoven, ale bude zvolen s ohledem na možnost vyhledávání a zaznamenávání změn v datech. Aplikace bude zároveň poskytovat aspoň jednoduché rozhraní pro dotazování.

Architektura celé aplikace bude vytvořena s ohledem na možnost snadného rozšíření o podporu dalších databázových platforem.

Cílovou skupinou projektu jsou aspoň středně pokročilí správci databází, kteří budou schopni porozumět všem prováděným operacím a v případě potřeby na ně dokáží adekvátně reagovat. Zároveň mohou logování přizpůsobit svým specifickým potřebám a rozšířit ho např. o indexy pro rychlejší vyhledávání nebo integritní omezení pro zajištění konzistence dat.

Kapitola 2

Teoretická východiska

2.1 Ukládání logovacích záznamů v databázi

Jednou z nejdůležitějších otázek celého projektu byla volba vhodného formátu logu. Proto byla hned v úvodu práce provedena analýza možných přístupů k ukládání logovacích dat. Hlavním požadavkem na řešení bylo, aby mohly být zaznamenávány nejen popisné informace, ale přímo i hodnoty jednotlivých databázových sloupců. Na základě těchto požadavků byly dále rozpracovány následující tři přístupy:

- komplexní logovací tabulka,
- jednoduchá logovací tabulka a změnová tabulka,
- sada historických tabulek.

2.1.1 Komplexní logovací tabulka

Tato možnost zahrnuje vytvoření jediné tabulky, která bude obsahovat v každém záznamu základní popisné informace (druh operace, čas, autor) a vedle toho i věrnou reprezentaci obsahu databázových sloupců ve zvoleném textovém formátu.

Výhodou tohoto přístupu je jednoduchá správa a dotazování na popisné informace. Naproti tomu nevýhodou je nutnost konverze obsahu sloupců na text, která nemusí být ani možná (např. v případě uživatelských datových typů) a u některých databází složité dotazování na hodnoty sloupců.

Podstatnou otázkou je i textový formát použitý pro uložení dat. Jako nejperspektivnější byly uvažovány následující dva způsoby:

seznam hodnot s oddělovači bohužel neumožňuje intuitivní rozlišování mezi NULL hodnotami a prázdnými řetězci. Dotazování na hodnoty sloupců bude navíc poměrně složité (nutnost uvádět i názvy sloupců nebo znát předem jejich pořadí) a pomalé (možnost zrychlení pomocí fulltextového

indexu – klasický B–strom by nestačil, dotazy totiž nebudou typicky na shodu).

XML umožňuje věrné zachycení podoby původního záznamu včetně NULL hodnot, jmen sloupců atd. Platformy, které nepodporují nativní ukládání XML dokumentů, by ukládaly záznamy jako text. Snadnost dotazování by závisela na existenci nástroje pro dotazování nad XML daty (např. Oracle podporuje jazyky *XQuery*[1] i *XPath*[2], PostgreSQL aspoň *XPath*). Rychlost by obdobně závisela na podpoře pro tvorbu XML indexů. Každopádně nelze očekávat řádově horší výsledky než v případě předchozí metody (na XML dokumenty bude v nejhorším případě pohlíženo jako na jistý druh seznamu s oddělovači).

2.1.2 Jednoduchá logovací tabulka a změnová tabulka

Logovací tabulka obsahuje pouze základní popisné informace. Hodnoty sloupců jsou uloženy ve změnové tabulce, která obsahuje pro každý sloupec původní tabulky jeden záznam s názvem sloupce a s hodnotou před a po změně.

Tento způsob uložení má bohužel zásadní problém a tím jsou datové typy sloupců změnové tabulky. Aby do ní bylo možné ukládat záznamy z různých sloupců, museli bychom je nejdříve transformovat na stejný datový typ. Jediným realistickým řešením je konverze na textový řetězec, což je řešení téměř identické s předchozí variantou. Výhodou je přímočará reprezentace NULL hodnot a jednoduché dotazování na hodnoty sloupců (bohužel bychom nemohli využít typované dotazy).

2.1.3 Historické tabulky

Historická tabulka přesně kopíruje strukturu vybrané tabulky v databázi. Díky tomu do ní lze ukládat kopie řádků v nezměněné podobě. V praxi by se do historické tabulky uložil změněný nebo odstraněný záznam spolu se základními popisnými údaji.

Pokud bychom navíc udělali drobnou změnu a do originální tabulky jsme přidali sloupec s časem vytvoření záznamu, tak by nebyl problém vytvořit pohled zobrazující stav k libovolnému časovému okamžiku v minulosti (odtud název historická tabulka).

[1] XQuery je funkcionální dotazovací jazyk pro dotazování nad XML daty. V současnosti se používá verze XQuery 1.0, podrobné informace lze nalézt v doporučení W3C konsorcia [9].

[2] XPath je dotazovací jazyk pro výběr uzlů z XML dokumentů. V současnosti existují verze 1.0 a 2.0, podrobnosti viz doporučení W3C [8] konsorcia.

Výhodou tohoto přístupu je především jednoduché a rychlé dotazování na hodnoty sloupců. Nevýhodou je velký počet historických tabulek a s tím spojená režie.

2.1.4 Zvolené řešení

Na základě zhodnocení výše uvedených metod byla pro logování zvolena kombinace komplexní logovací tabulky a historických tabulek. Obě metody budou implementovány nezávisle a jejich nasazení bude záviset pouze na uvážení uživatele.

Komplexní logovací tabulka bude sloužit především k základnímu logování s důrazem na vyhledávání na základě popisných informací (datum a čas, tabulka, autor). Hodnoty sloupců budou ukládány jako XML dokument odpovídající *XML Schema*[3] schématu uvedenému v příloze A.1.

Historické tabulky budou nasazeny v minimální podobě a budou sloužit výhradně k realizaci pohledů na historická data. Každý záznam ve zdrojové i historické tabulce proto bude obsahovat jednoznačnou identifikaci doby platnosti.

2.2 Logovací koncept

Logování databázových operací může být u drtivé většiny databázových platforem realizováno přímo na úrovni databáze pomocí triggerů. Tento postup bude využívat i navržená aplikace. Praktickou realizaci proto můžeme rozdělit do dvou samostatných částí:

databázová část zajišťuje funkcionalitu na straně databáze, tj. vytváří potřebné datové struktury a uložené procedury. Vzhledem k drobným odlišnostem napříč databázovými platformami je nezbytné vytvořit pro každou platformu vlastní implementaci.

grafické uživatelské rozhraní představuje samostatnou aplikaci, která uživateli poskytne přehledné rozhraní pro správu logování. Tato část projektu je již odstíněna od databázové části a může proto využívat libovolné univerzální rozhraní pro připojení k databázi. Aplikace bude dále poskytovat prostředky pro export logovacích dat z databáze.

[3]XML Schema je jazyk pro popis struktury XML dokumentu. Podrobné informace lze nalézt na stránkách konsorcia W3C [6].

2.2.1 Pracovní prostředí

Pojmem *pracovní prostředí* budeme označovat skupinu databázových objektů, které jsou nezbytné pro praktickou realizaci logování. Konkrétně se jedná o následující:

prefixová funkce vrací prefix, který bude použit v názvech všech objektů vytvořených logovací aplikací. Název prefixové funkce je standardně `db_logger_prefix` (změnit ho lze pouze globálně na úrovni celé aplikace viz popis v části 3.5.7). Prefix slouží primárně k identifikaci logovacích objektů z důvodů jednoduché správy.

logovací sekvence slouží ke generování posloupnosti unikátních číselných identifikátorů.

Oba prvky pracovního prostředí jsou vytvářeny společně a bez jejich přítomnosti nelze realizovat logování.

2.2.2 Logovací tabulka

Logovací tabulka není povinnou součástí aplikace. Pro její vytvoření je nutná existence pracovního prostředí. Sloupce logovací tabulky jsou zachyceny v tab. 2.1 (konkrétní rozsahy typů nejsou uváděny a záleží na specifických vlastnostech každé platformy).

Standardně nejsou na logovací tabulce vytvářeny žádné indexy. Uživatel je musí v případě potřeby vytvořit ručně. Volba druhu indexu a indexovaných sloupců závisí výhradně na dotazech, jejichž vyhodnocování bude uživatel primárně požadovat.

Název	**Typ**	**Popis**
id	integer	jednoznačný číselný identifikátor (primární klíč)
table_name	varchar	název tabulky
user_name	varchar	jméno uživatele
operation	char(1)	typ operace (I = insert, U = update, D = delete)
changed	timestamp	datum a čas provedení
old_values	xml	hodnoty sloupců před změnou (vyplněno pouze pro update a delete)
new_values	xml	hodnoty sloupců po změně (vyplněno pouze pro insert a update)

Tabulka 2.1: Sloupce logovací tabulky

Data z logovací tabulky bude možné exportovat do XML dokumentu, který se hodí např. k archivaci nebo k pokročilému dotazování pomocí XPath nebo XQuery. Schéma exportního formátu je uvedeno v příloze A.2.

2.2.3 Logovací triggery

Logovací triggery generuje aplikace automaticky. Tělo triggeru je tvořeno procedurálním kódem (u databáze Oracle v PL/SQL, u PostgreSQL v pgPL/SQL), který zajišťuje vložení záznamu do logovací tabulky.

Operace, které budou logovány lze zvolit při vytváření triggeru. Pokud databáze podporuje update triggery pro sloupce[4] (příkladem může být databázová platforma Oracle), tak lze použít i ty .

Aplikace standardně nedoporučuje logování některých datových typů. Jedná se především o binární data (např. *BLOB*), typy velkého rozsahu (např. *CLOB*, *text* nebo *XML*) a objektové datové typy. V případě potřeby lze jejich logování povolit ručně při vytváření triggeru.

2.2.4 Historické tabulky

Historická tabulka přesně kopíruje schéma rodičovské tabulky, tj. obsahuje sloupce stejných názvů a datových typů. Kromě toho jsou do rodičovské i historické tabulky přidány tři pomocné *historické sloupce* viz tab. 2.2.

Název	**Typ**	**Popis**
PREFIX_id	integer	jednoznačný číselný identifikátor
PREFIX_created	timestamp	začátek platnosti (tj. vytvoření nebo změna) řádku v databázi
PREFIX_expired	timestamp	konec platnosti (tj. smazání) řádku v databázi; tento sloupec bude vytvořen pouze v historické tabulce

Tabulka 2.2: Sloupce historické tabulky

Záznamy z rodičovské tabulky jsou přesunuty do historické tabulky pokaždé, když dojde ke smazání řádku nebo ke změně hodnoty libovolného sloupce. Z historické tabulky se záznamy nikdy neodstraňují.

Integritní omezení na rodičovské tabulce se do historické tabulky automaticky nepromítají, protože by v mnoha případech nemohla být dodržena (např. u unikátních hodnot sloupců). V případě potřeby je možné doplnit je ručně, stejně jako indexy.

[4]Trigger se v tomto případě spustí pouze pokud dojde k fyzické změně konkrétního sloupce (sloupců), nikoliv při všech updatech.

Sloupec PREFIX_id slouží v historické tabulce jako umělý primární klíč. Klíč z původní tabulky převzít nemůžeme (kvůli unikátnosti) a vytvoření tabulky bez primárního klíče by nám znemožnilo jednoznačně identifikovat záznamy v tabulce.

Stejně jako v případě logovací tabulky lze data historického pohledu exportovat do XML dokumentu. Schéma popisující jeho strukturu je uvedeno v příloze A.3.

Kapitola 3

Implementace

3.1 Zadání

Vytvořte aplikaci, která uživateli umožní pro zvolenou databázi vytvořit logovací triggery nad zvolenými tabulkami a pro zvolené operace (insert, update, delete). Pokud příslušná databáze podporuje update triggery také pro sloupce, pak by měl mít uživatel možnost zvolit i jednotlivé sloupce.

Pro implementaci zvažte různé možnosti formátu logu, především s ohledem na možnosti vyhledávání různých událostí. Nedílnou součástí řešení je implementace alespoň pro dvě různé databázové platformy, např. MySQL, PostgreSQL nebo Oracle, přičemž aplikace by měla být v rámci možností co nejvíce rozšiřitelná na další platformy.

3.2 Vývojová platforma

Aplikace je napsána v programovacím jazyce *Java* (pro běh je nutná aspoň verze J2SE 5.0). Hlavní důvody pro tuto volbu jsou následující:

- možnost tvorby multiplatformních aplikací, které jdou spouštět na mnoha operačních systémech,
- přítomnost technologie *JDBC* [2], která poskytuje jednotné API pro přístup k celé řadě databázových platforem,
- existence prostředků pro tvorbu GUI.

Jako vývojové prostředí byla použita aplikace *NetBeans 6.5* [7], která obsahuje jak všechny důležité vývojářské nástroje, tak i prostředky pro interaktivní tvorbu uživatelského rozhraní.

3.3 Podporované databázové platformy

Aplikace nativně podporuje dvě databázové platformy:

Oracle jako zástupce proprietárních řešení a zároveň v současnosti nejrozšířenější podnikovou databázi. Aplikace byla testována s verzí databáze *Oracle 10g Release 2*.

PostgreSQL jako příklad pokročilé open-source databáze. Testování proběhlo s verzí *PostgreSQL 8.3.7*.

Podporu dalších platforem lze relativně snadno doplnit. Přesný postup je popsán v části 4.8.

3.4 Struktura tříd aplikace

Aplikace je napsána kompletně v programovacím jazyce Java. Všechny třídy jsou umístěny v balíku `cz.matasek.dbLogger` nebo v některém z jeho potomků. Struktura a stručný popis obsahu balíků je uveden v následujícím přehledu:

cz.matasek.dbLogger obsahuje především spustitelnou třídu `Main` a dále třídy s definicemi konstant.

cz.matasek.dbLogger.database obsahuje vše co souvisí s připojením k databázi. Jedná se především o interface `DatabaseConnection`, který realizuje všechna volání databáze. Dále je zde dostupná výchozí implementace tohoto rozhraní `AbstractDatabaseConnection` a také implementace pro konkrétní databázové platformy.

cz.matasek.dbLogger.exception obsahuje definice výjimek použitých v aplikaci.

cz.matasek.dbLogger.gui obsahuje komponenty grafického uživatelského rozhraní.

cz.matasek.dbLogger.model obsahuje modely databázových objektů – např. databázi, tabulku, sloupec.

cz.matasek.dbLogger.sql obsahuje datové struktury potřebné pro provádění SQL dotazů. Asi nejdůležitější je třída `SQLStatement`, která představuje model konkrétního dotazu.

3.5 Důležité třídy

3.5.1 Interface DatabaseConnection

Rozhraní `DatabaseConnection` funguje jako výhradní přístupový kanál k databázi. Obsahuje funkce nezbytné pro generování logovacích triggerů, které lze rozdělit do tří hlavních skupin:

- dotazy na existenci nebo stav prvků v databázi (např. existence logovací tabulky nebo logovacích triggerů),
- funkce pro tvorbu nebo rušení logovacích objektů (např. vytvoření pracovního prostředí, logovací tabulky),
- dotazy vracející data (např. historické pohledy, obsah logovací tabulky nebo seznam tabulek v databázi).

3.5.2 Třída AbstractDatabaseConnection

Tato abstraktní třída je implementací rozhraní `DatabaseConnection`. Z důvodu snadné rozšiřitelnosti využívá koncept, kdy všechny potřebné SQL dotazy jsou uloženy v textových souborech, v separátní složce pro každou databázovou platformu. Při provádění dotazů pak tato třída transparentně načte potřebný soubor s SQL příkazem, který následně provede.

Takto navržená třída má dvě hlavní výhody:

1. Uživatel může snadno provést drobné změny přímo v externích textových souborech s SQL příkazy. Lze přitom upravit např. logovací triggery, dotaz vracející seznam tabulek v databázi nebo automatickou tvorbu indexů.
2. Přidání podpory pro novou databázovou platformu se při existenci JDBC driveru omezí v podstatě jen na vytvoření potřebných SQL příkazů a jejich nahrání do příslušné složky.

3.5.3 Třída SQLStatement

Třída `SQLStatement` je modelem libovolného SQL dotazu. Konkrétní podobu dotazu určují dva faktory – typ dotazu (je určen výčtem `SQLStatementType`) a databázová platforma. Na základě kombinace těchto parametrů se ze souborového systému načte požadovaný SQL dotaz, následně jsou provedeny náhrady klíčových slov a nakonec je dotaz proveden.

Klíčová slova, která je v SQL dotazech možné používat, jsou uvedena v příloze B. Každé klíčové slovo musí být ohraničeno znaky %. Následující kód je ukázkou SQL příkazu pro smazání logovací tabulky:

```
DROP TABLE %LOGGING_TABLE_NAME%
```

Načítané dotazy se v aplikaci automaticky cachují a při opakovaném spouštění se hledají v operační paměti a nikoliv na disku. Kromě toho třída zajišťuje i logování všech prováděných dotazů – každý příkaz je po úspěšném provedení uložen do statické proměnné obsahující historií dotazů.

3.5.4 Třídy Column a Table

`Column` je model sloupce v databázi. Každý sloupec obsahuje informaci o názvu a datovém typu.

`Table` je model databázové tabulky. Tabulka obsahuje jednak seznam datových sloupců (jako objekty typu `Column`) a dále informaci o existenci logovacích triggerů (popř. i o logovaných operacích) a historické tabulky.

3.5.5 Třída GUI

`GUI` je třída představující hlavní okno aplikace. V aplikaci je použit tzv. *CardLayout*, což znamená, že okna aplikace jsou dostupná pod záložkami v horní části okna. Třída GUI funguje jako singleton a je staticky dostupná napříč celou aplikací.

3.5.6 Interface Refreshable

Interface `Refreshable` označuje třídy, jejichž vnitřní stav je občas potřeba obnovit. V aplikaci ho implementuje většina panelů GUI kvůli zobrazování aktuálních dat z databáze.

3.5.7 Třída Settings

Třída `Settings` zajišťuje načítání nastavení z externího XML souboru `settings.xml`. Soubor obsahuje následující konfigurační elementy:

prefix_function_name název prefixové funkce

default_logging_prefix standardní prefix logovacích objektů

logging_table_suffix přípona názvu logovací tabulky

history_table_suffix přípona názvu historické tabulky

sequence_suffix přípona názvu logovací sekvence

logging_trigger_suffix přípona názvu logovacího triggeru

history_trigger_suffix přípona názvu historického triggeru

sql_directory název složky s SQL příkazy

xsl_directory název složky s XSLT transformacemi

xsl_history název souboru s XSLT transformací historické tabulky do HTML

xsl_query název souboru s XSLT transformací logovací tabulky do HTML

Změny provedené v konfiguračním souboru se projeví ihned po restartu aplikace.

3.5.8 Třída SQLDataType

Třída `SQLDataType` obsahuje výčet známých datových typů. Konstruktor každého typu obsahuje příznak, zda je logování doporučeno (*true*) nebo ne (*false*). Sloupce doporučených datových typů mají při vytváření logovacího triggeru automaticky zaškrtnuté logování.

3.6 Externí knihovny

Aplikace používá tři externí knihovny, které se starají o připojení k databázi, resp. vzhled celé aplikace.

`ojdbc6.jar` je JDBC driver pro databázi Oracle,

`postgresql-8.3-604.jdbc4.jar` je JDBC driver pro databázi PostgreSQL,

`looks-2.2.1.jar` je skin pro Swing aplikace.

3.7 Dokumentace

Všechny zdrojové kódy obsahují komentáře, které mohou být použity pro vygenerování *JavaDoc* dokumentace. Ta je dostupná na přiloženém CD a obsahuje podrobný popis všech tříd, funkcí a proměnných.

3.8 Test rychlosti logování

Pro zjištění náročnosti logování byla provedena krátká série testů zjišťujících dobu potřebnou pro přidání (popř. editaci nebo smazání) 1000 řádků do databáze. Test byl proveden dvakrát – jednou s aktivním logováním a podruhé bez něj. Výsledky

Platforma	**Typ**	**Čas [s] (no log)**	**Čas [s] (log)**	**Poměr**
Oracle	insert	3,26	9,59	2,94
PostgreSQL	insert	0,15	0,67	4,47
Oracle	update	0,67	8,59	12,82
PostgreSQL	update	0,07	0,55	7,86
Oracle	delete	0,67	6,15	9,18
PostgreSQL	delete	0,02	0,61	30,5

Tabulka 3.1: Test rychlosti logování

jsou uvedeny v tab. 3.1 (ve sloupci čas je uvedena doba pro zpracování celého tisíce záznamů, poměr je podíl časů se zapnutým logováním a bez něj).

Zjištěné výsledky jsou pouze orientační a jejich absolutní hodnoty jsou nevýznamné, protože databáze nebyly konfigurovány s ohledem na rychlost, ale bylo použité jejich standardní nastavení. Ukázalo se však, že manipulace s daty je při zapnutém logování *řádově asi 10-krát pomalejší*. Toto zpomalení by mohlo mít dopad na použitelnost systému pouze u tabulek s obrovským počtem DML operací. U menších databázi, pro které je vyvíjený nástroj určen především, však nejsou dosažené hodnoty překážkou pro nasazení logování.

Kapitola 4

Uživatelská dokumentace

4.1 Instalace

Aplikace *Database Logger* je distribuována jako JAR archiv `DatabaseLogger.jar`, a proto není nutná žádná explicitní instalace. Adresář s archivem však musí obsahovat i složky `sql` (obsahuje používané SQL příkazy rozdělené do adresářů podle databázové platformy), `xsl` (obsahuje XSLT transformace pro export do HTML) a `xsd` (obsahuje schémata používaných XML dokumentů).

Aplikaci lze spustit na libovolném operačním systému, pro který je dostupná Java aspoň ve verzi 5. Z příkazové řádky lze provést spuštění následujícím příkazem:

```
java -jar DatabaseLogger.jar
```

4.2 Funkčnost aplikace

Aplikace umožňuje uživateli automatizované logování zvolených databázových operací. Získaná data mohou podle typu logování sloužit jako archiv provedených změn nebo z nich může být získán historický stav tabulky.

4.2.1 Datové struktury

Před začátkem logování je nutné v databázi vytvořit logovací prostředí, tj. následující dvě pomocné datové struktury (podrobný popis viz část 4.3.2):

prefixová funkce vrací prefix povinně použitý u všech vytvářených objektů.

logovací sekvence umožňuje generování posloupnosti jedinečných číselných identifikátorů.

4.2.2 Logovací tabulka

Logovací tabulka slouží k ukládání záznamů o provedených operacích v databázi. Tabulku je potřeba explicitně vytvořit (postup viz část 4.3.2) a nad každou tabulkou, jejíž změny chceme logovat, je potřeba vytvořit ještě logovací trigger (postup viz část 4.3.3). Při vytváření triggeru lze specifikovat operace i sloupce, které se budou logovat.

Do logovací tabulky se pro každý záznam ukládají následující údaje:

- název tabulky,
- jméno uživatele, který změnu provedl,
- databázová operace (insert, update nebo delete),
- datum a čas změny,
- hodnoty vybraných sloupců před změnou (pouze v případě operací update a delete),
- hodnoty vybraných sloupců po změně (pouze v případě operací insert a update).

4.2.3 Historické tabulky

Historická tabulka slouží k zaznamenávání všech změn v jedné konkrétní tabulce. Díky tomu, že přesně kopíruje schéma logované tabulky, lze do ní ukládat data ze všech sloupců bezztrátově (tj. data nemusí být konvertována např. na textové řetězce). Logování změn do historické, resp. logovací tabulky je zcela nezávislé.

Historická tabulka slouží výhradně ke generování historických pohledů – tj. obrazu tabulky k nějakému okamžiku v minulosti.

4.3 Ovládání aplikace

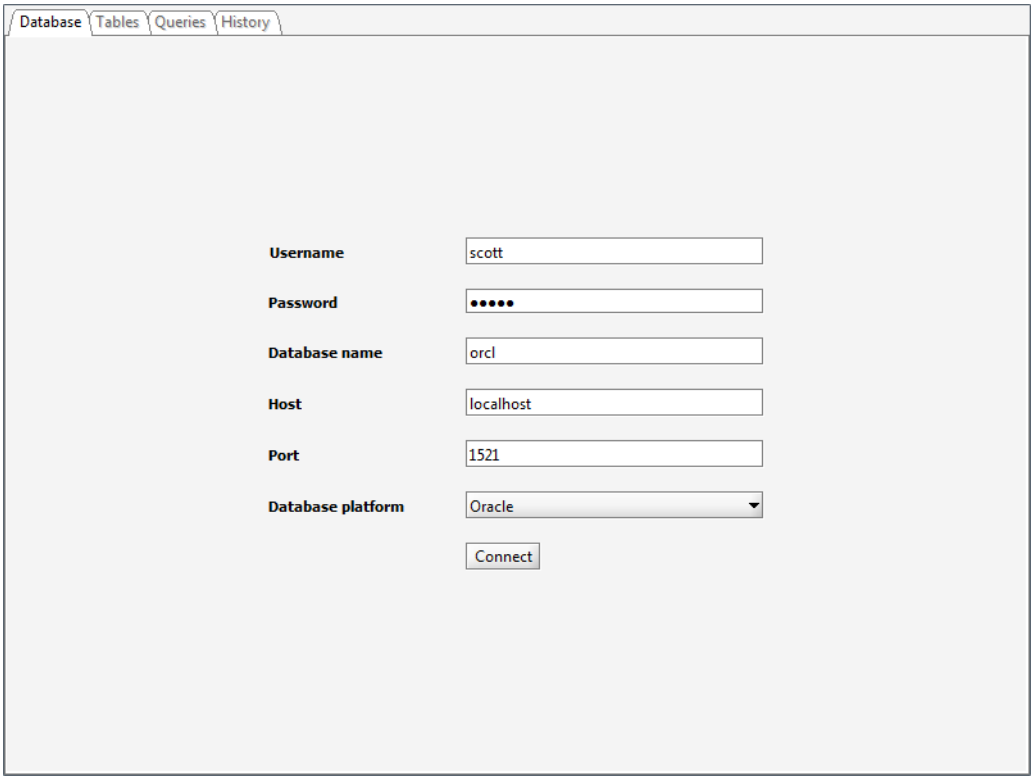
4.3.1 Přihlášení k databázi

Po spuštění aplikace se objeví formulář pro připojení k databázi viz obr. 4.1. Pro navázání připojení je nutné vyplnit následující povinné údaje:

- uživatelské jméno,
- heslo,
- název databáze,
- hostitelský server,

- port,
- databázovou platformu.

V případě úspěšného připojení se zobrazí základní informace o databázi, jinak se objeví dialog s popisem chyby připojení.



Obrázek 4.1: Formulář pro připojení k databázi

4.3.2 Informace o databázi

Po úspěšném připojení k databázi se v záložce *Database* zobrazí okno se základními informacemi o databázi a logovacích objektech viz obr. 4.2. V tomto okně lze zároveň vytvořit pracovní prostředí pro logování, popř. logovací tabulku. V následujícím seznamu je uveden stručný popis důležitých objektů:

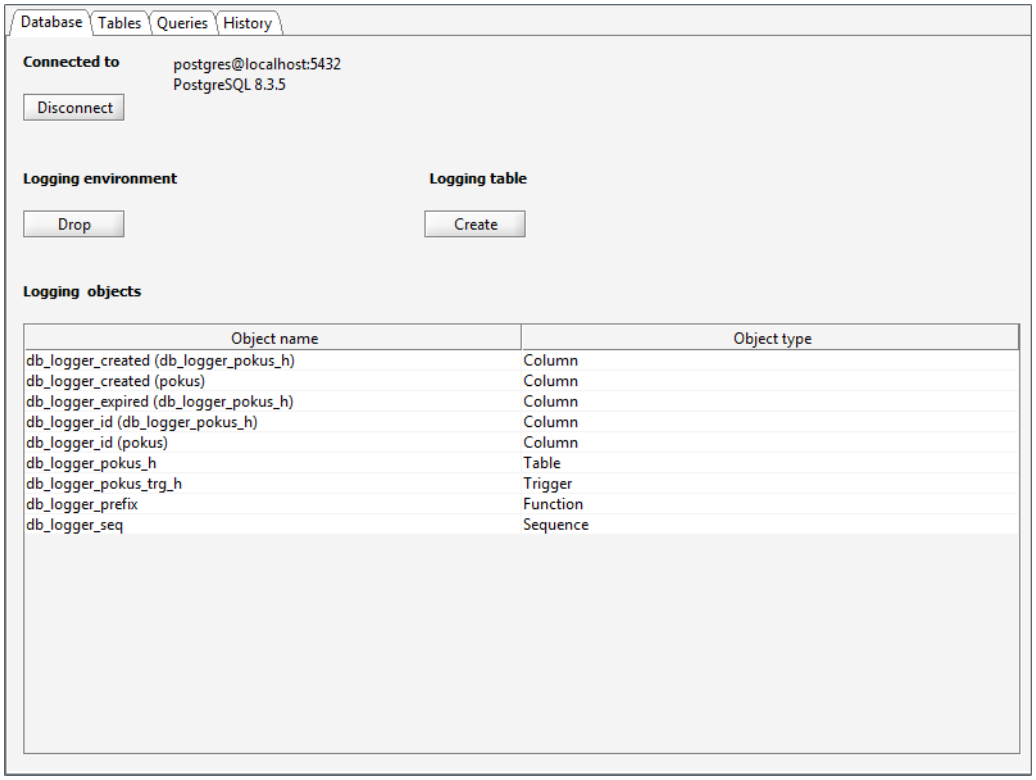
Connected to obsahuje údaje o právě připojené databázi – uživatelské jméno, databázovou platformu atd.

Disconnect je tlačítko pro odhlášení z databáze.

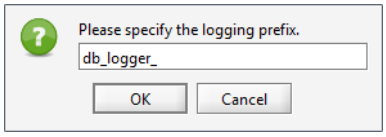
Logging environment umožňuje vytvoření (tlačítko *Create*), resp. zrušení (tlačítko *Drop*) pracovního prostředí. Při vytvoření logovacího prostředí musí uživatel povinně zadat logovací prefix viz obr. 4.3.

Logging table umožňuje vytvoření (tlačítko *Create*), resp. zrušení (tlačítko *Drop*) logovací tabulky.

Logging objects je tabulka obsahující seznam objektů, které byly automaticky vytvořeny logovací aplikací. Tabulka obsahuje pro každý objekt jeho název a typ (např. tabulka, trigger).



Obrázek 4.2: Okno s informacemi o databázi



Obrázek 4.3: Zadávání logovacího prefixu.

4.3.3 Informace o tabulce

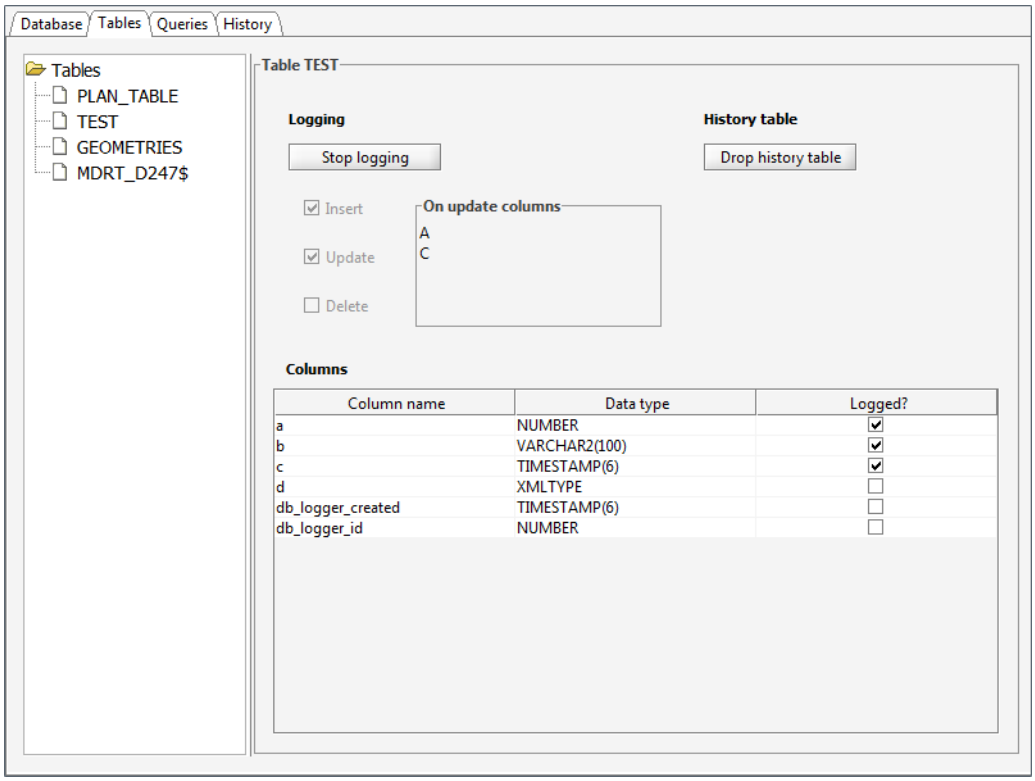
Pod záložkou *Tables* jsou dostupné informace o tabulkách dostupných v databázi viz obr. 4.4. V levé části okna se nachází seznam tabulek, po kliknutí na konkrétní tabulku se v pravé části objeví okno s podrobnými informacemi. Okno obsahuje následující ovládací prvky:

Logging obsahuje informace o stavu logování dané tabulky. Logování může být spuštěno (tlačítko *Start logging*) nebo pozastaveno (tlačítko *Stop logging*).

Pokud je logování aktivní jsou pomocí zaškrtávacích políček vyznačeny logované operace, popř. je uveden i seznam sloupců, které spouští on update triggery pro sloupce (v seznamu *On update columns*).

History table umožňuje vytvoření historické tabulky a začátek logování úplné historie (tlačítko *Create history table*), resp. její zrušení (tlačítko *Drop history table*).

Columns obsahuje tabulku se seznamem sloupců. Pro každý sloupec je uveden název (*Column name*), datový typ (*Data type*) a příznak logování (*Logged?*).



Obrázek 4.4: Okno s informacemi o tabulce

4.4 Vytvoření logovacího triggeru

Logovací trigger lze vytvořit na záložce *Tables* pomocí tlačítka *Start logging*. Po stisknutí se objeví okno obsahující podrobná nastavení logování viz obr. 4.5.

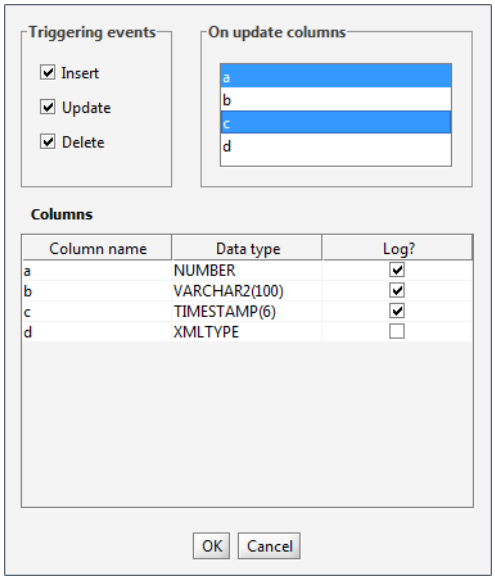
U každého triggeru lze nastavit následující parametry:

Triggering events obsahuje zaškrtávací políčka pro vyznačení logovaných operací (insert, update, delete).

On update columns obsahuje seznam sloupců, nad kterými bude vytvořen on update trigger pro sloupce (pokud není zaškrtnuto políčko update, nastavení

se ignoruje). V případě, že není vybrán žádný sloupec (popř. jsou vybrány všechny), tak se vytvoří pouze klasický update trigger.

Columns obsahuje tabulku se seznamem sloupců. U každého sloupce lze explicitně nastavit, zda se jeho hodnoty budou logovat nebo ne.



Obrázek 4.5: Dialog pro vytvoření logovacího triggeru

4.5 Dotazování nad logovací tabulkou

Okno pro dotazování nad logovací tabulkou je dostupné pod záložkou *Queries* viz obr. 4.6. Parametry dotazu lze nastavovat pomocí následujících ovládacích prvků:

Operations umožňuje výběr požadovaných operací (insert, update, delete).

Tables slouží pro výběr tabulek. Pomocí tlačítka *Add* lze do seznamu doplnit další tabulky.

Users slouží pro výběr uživatelů. Další uživatele lze opět přidat pomocí tlačítka *Add*.

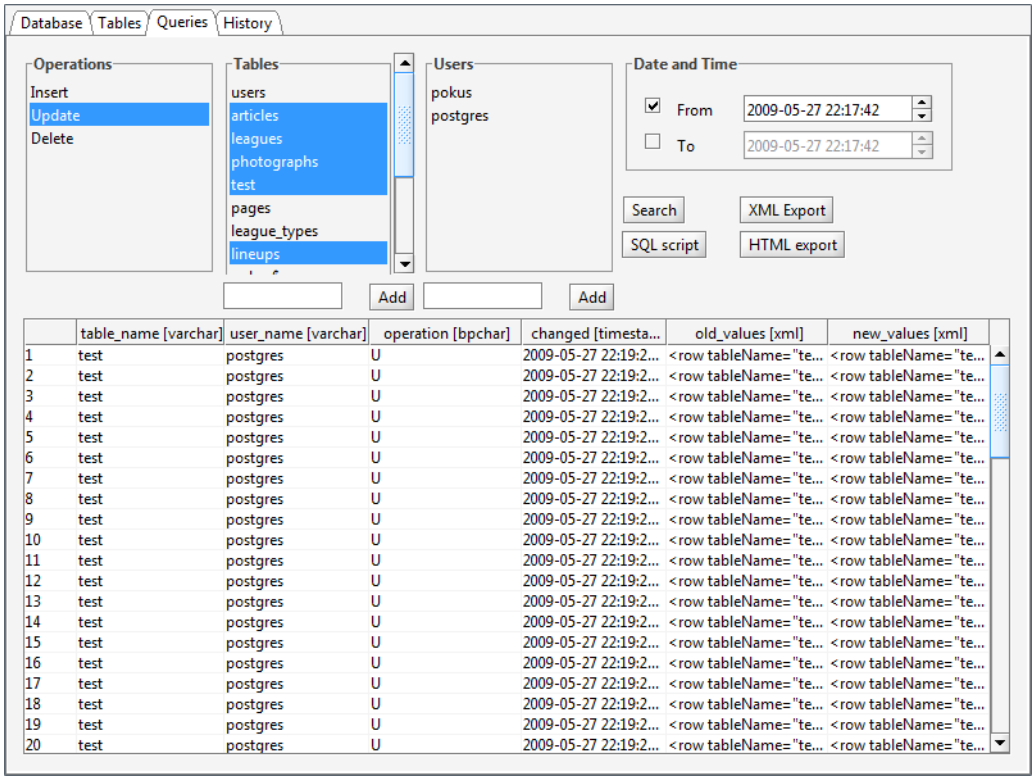
Date and Time slouží pro vymezení sledovaného časového období. Zvlášť lze zadat datum a čas pro začátek (přepínač *From*) i konec (přepínač *To*) sledovaného období.

Search vyhledá v logovací tabulce záznamy podle specifikovaných kritérií.

SQL script zobrazí SQL kód, který provedl výběr aktuálně zobrazených dat viz obr. 4.7.

XML export exportuje načtená data do XML souboru.

HTML export exportuje načtená data do HTML souboru.



	table_name [varchar]	user_name [varchar]	operation [bpchar]	changed [timesta...	old_values [xml]	new_values [xml]
1	test	postgres	U	2009-05-27 22:19:2...	<row tableName="te...	<row tableName="te...
2	test	postgres	U	2009-05-27 22:19:2...	<row tableName="te...	<row tableName="te...
3	test	postgres	U	2009-05-27 22:19:2...	<row tableName="te...	<row tableName="te...
4	test	postgres	U	2009-05-27 22:19:2...	<row tableName="te...	<row tableName="te...
5	test	postgres	U	2009-05-27 22:19:2...	<row tableName="te...	<row tableName="te...
6	test	postgres	U	2009-05-27 22:19:2...	<row tableName="te...	<row tableName="te...
7	test	postgres	U	2009-05-27 22:19:2...	<row tableName="te...	<row tableName="te...
8	test	postgres	U	2009-05-27 22:19:2...	<row tableName="te...	<row tableName="te...
9	test	postgres	U	2009-05-27 22:19:2...	<row tableName="te...	<row tableName="te...
10	test	postgres	U	2009-05-27 22:19:2...	<row tableName="te...	<row tableName="te...
11	test	postgres	U	2009-05-27 22:19:2...	<row tableName="te...	<row tableName="te...
12	test	postgres	U	2009-05-27 22:19:2...	<row tableName="te...	<row tableName="te...
13	test	postgres	U	2009-05-27 22:19:2...	<row tableName="te...	<row tableName="te...
14	test	postgres	U	2009-05-27 22:19:2...	<row tableName="te...	<row tableName="te...
15	test	postgres	U	2009-05-27 22:19:2...	<row tableName="te...	<row tableName="te...
16	test	postgres	U	2009-05-27 22:19:2...	<row tableName="te...	<row tableName="te...
17	test	postgres	U	2009-05-27 22:19:2...	<row tableName="te...	<row tableName="te...
18	test	postgres	U	2009-05-27 22:19:2...	<row tableName="te...	<row tableName="te...
19	test	postgres	U	2009-05-27 22:19:2...	<row tableName="te...	<row tableName="te...
20	test	postgres	U	2009-05-27 22:19:2...	<row tableName="te...	<row tableName="te...

Obrázek 4.6: Okno pro dotazování nad logovací tabulkou

```
SELECT
   table_name, user_name, operation, changed, old_values, new_values
FROM
   db_logger_logs
WHERE
   table_name IN ('photographs','test','league_types')
   AND
   user_name IN ('pokus')
   AND
   operation IN ('I','D')
   AND
   changed >= to_timestamp('2009-05-19 09:22:20', 'YYYY-MM-DD HH24:MI:SS')
```

Obrázek 4.7: Zobrazení aktuálně provedeného SQL příkazu

4.6 Načtení historických dat

Okno pro zobrazení historických dat je dostupné pod záložkou *History* viz obr. 4.8. V okně jsou dostupné následující ovládací prvky:

Tables slouží pro výběr požadované tabulky.

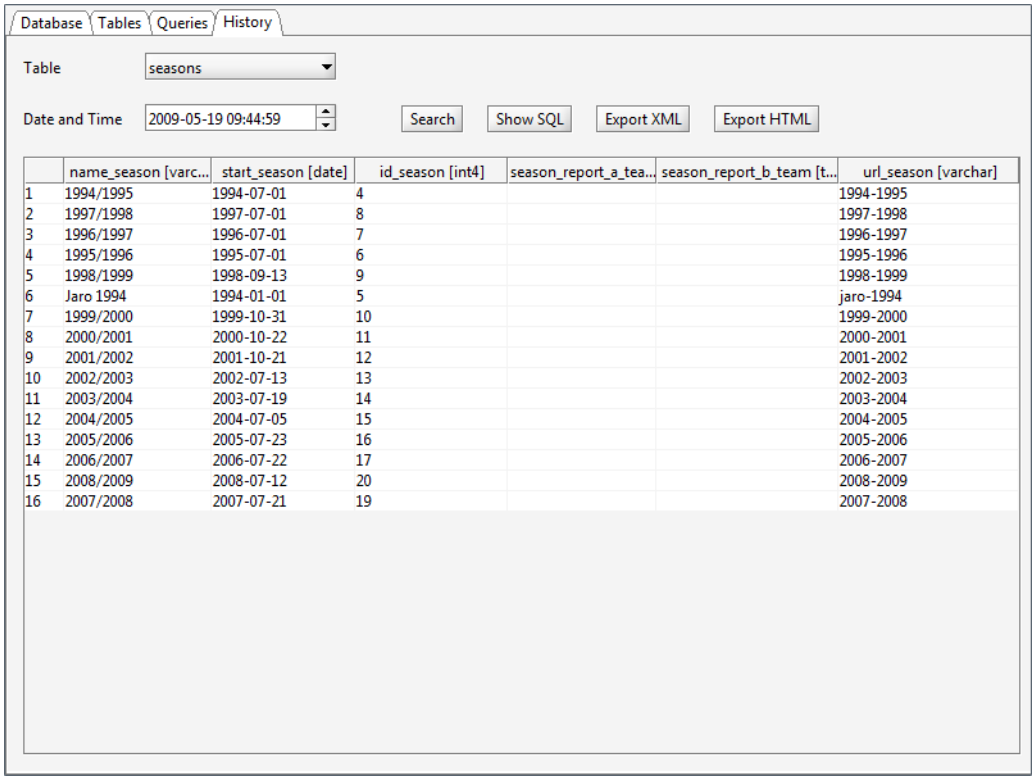
Date and Time slouží pro výběr časového okamžiku v historii.

Search vyhledá záznamy pro danou tabulku a historický okamžik.

SQL script zobrazí SQL kód, který provedl výběr aktuálně zobrazených dat.

XML export exportuje načtená data do XML souboru.

HTML export exportuje načtená data do HTML souboru.



	name_season [varc...	start_season [date]	id_season [int4]	season_report_a_tea...	season_report_b_team [t...	url_season [varchar]
1	1994/1995	1994-07-01	4			1994-1995
2	1997/1998	1997-07-01	8			1997-1998
3	1996/1997	1996-07-01	7			1996-1997
4	1995/1996	1995-07-01	6			1995-1996
5	1998/1999	1998-09-13	9			1998-1999
6	Jaro 1994	1994-01-01	5			jaro-1994
7	1999/2000	1999-10-31	10			1999-2000
8	2000/2001	2000-10-22	11			2000-2001
9	2001/2002	2001-10-21	12			2001-2002
10	2002/2003	2002-07-13	13			2002-2003
11	2003/2004	2003-07-19	14			2003-2004
12	2004/2005	2004-07-05	15			2004-2005
13	2005/2006	2005-07-23	16			2005-2006
14	2006/2007	2006-07-22	17			2006-2007
15	2008/2009	2008-07-12	20			2008-2009
16	2007/2008	2007-07-21	19			2007-2008

Obrázek 4.8: Okno pro zobrazení historických dat

4.7 Pokročilé možnosti

4.7.1 Tvorba indexů

Aplikace standardně nevytváří žádné indexy pro zrychlení dotazování. Důvodem je, že aplikace má být co nejjednodušší a nemá zatěžovat databázi zbytečnou režií.

Uživatel, který požaduje vytvoření indexů, může jejich podporu doplnit jedním z následujících způsobů:

- ručně vytvoří požadované indexy přímo v databázi,
- pro automatické vytváření indexů doplní jejich definici do zdrojového SQL dotazu pro vytvoření logovací (soubor `create_logging_table.sql`), resp. historické tabulky (soubor `create_history_table.sql`).

4.7.2 Dotazování nad XML daty

Protože jsou hodnoty sloupců v logovací tabulce uloženy v XML dokumentu, je potřeba používat pro vyhledávání podle hodnot sloupců některý z XML dotazovacích jazyků (např. XPath nebo XQuery). Prakticky lze dotazování provádět na dvou místech:

- přímo v databázi, pokud má podporu pro XML dotazování (Oracle i PostgreSQL tuto podporu nabízí),
- v XML dokumentu, který je vyexportován podle popisu v části 4.5.

Jako ukázkový příklad uveďme použití XPath dotazu v Oracle. Následující dotaz vybere všechny záznamy, u kterých byla hodnota sloupce *column* před změnou nastavena na *value*:

```
SELECT
  *
FROM
  db_logger_logs
WHERE
  existsNode(
    old_values,
    '/db:row/db:col[@name = "column"] = "value"',
    'xmlns:db="http://www.matasek.net/DatabaseLogger"'
  ) = 1;
```

4.8 Přidání nové databázové platformy

Aplikace byla navržena s ohledem na možnost rozšíření o další databázové platformy. Novou platformu lze do aplikace přidat dvěma odlišnými způsoby. V obou případech je ale nutné nejprve upravit definici výčtu `DatabasePlatform`, který obsahuje seznam podporovaných platforem.

První možnost zahrnuje novou a zcela vlastní implementaci rozhraní `DatabaseConnection`. Tato možnost je poměrně pracná, ale může být použita kdykoliv pro libovolnou platformu.

Druhá jednodušší možnost využívá pilotní implementaci pomocí třídy `AbstractDatabaseConnection`. Tato implementace předpokládá existenci SQL příkazů pro všechny prováděné operace. Soubory s SQL příkazy jsou umístěny ve složce `sql` a v její podsložce odvozené od názvu platformy. Celkem je potřeba vytvořit 30 dotazů, jejich přesný význam je popsán v příloze C.

4.9 Licence aplikace

Aplikace je šířena pod licencí *BSD license*. To znamená, že aplikace i její zdrojové kódy mohou být šířeny popř. modifikovány při splnění několika málo podmínek ohledně uvádění copyrightu a BSD licence.

Kapitola 5

Závěr

V teoretické části aplikace jsou stručně popsány některé možné přístupy k ukládání logovacích dat. Dále je popsán celý koncept logování v relační databázi.

V praktické části byla vytvořena aplikace *Database Logger*, která splňuje všechny požadavky specifikované v zadání. Nad rámec těchto požadavků se podařilo implementovat export logů z databáze do XML a HTML.

5.1 Budoucnost projektu

V budoucnu by aplikace mohla být rozšířena o některou z následujících možností:

- přidání podpory pro další databázové platformy,
- rozšíření možností dotazování nad logovací tabulkou,
- volitelná tvorba indexů nad některými sloupci,
- volitelná implementace logovací tabulky jako XML tabulky (nikoliv relační).

Literatura

[1] *Java Platform, Standard Edition 6 API Specification* [online]. 2008.
Dostupné z: <http://java.sun.com/javase/6/docs/api/>.

[2] *JDK 6 Java Database Connectivity (JDBC) – related APIs & Developer Guides* [online]. 2002.
Dostupné z: <http://java.sun.com/javase/6/docs/technotes/guides/jdbc>.

[3] *Oracle Technology Network* [online].
Dostupné z: <http://www.oracle.com/technology/index.html>.

[4] *PostgreSQL: Documentation* [online]. 1996–2009.
Dostupné z: <http://www.postgresql.org/docs/>.

[5] *The BSD License* [online]. 2006.
Dostupné z: <http://www.opensource.org/licenses/bsd-license.php>.

[6] *W3C XML Schema* [online]. 2000–2008.
Dostupné z: <http://www.w3.org/XML/Schema>.

[7] *Welcome to NetBeans* [online].
Dostupné z: <http://www.netbeans.org/>.

[8] *XML Path Language (XPath) 2.0* [online]. 2007.
Dostupné z: <http://www.w3.org/TR/xpath20/>.

[9] *XQuery 1.0: An XML Query Language* [online]. 2007.
Dostupné z: <http://www.w3.org/TR/xquery/>.

Příloha A

Schémata použitých XML dokumentů

Všechny používané XML dokumenty jsou generovány v defaultním jmenném prostoru *http://www.matasek.net/DatabaseLogger*.

A.1 XML schéma pro reprezentaci řádku tabulky

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns="http://www.matasek.net/DatabaseLogger"
    targetNamespace="http://www.matasek.net/DatabaseLogger"
    elementFormDefault="qualified">

    <xs:simpleType name="nameType">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>

    <xs:complexType name="columnType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="name" type="nameType"
                    use="required"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

    <xs:complexType name="rowType">
        <xs:sequence>
            <xs:element name="col" type="columnType" minOccurs="0"
                maxOccurs="unbounded" nillable="true"/>
```

```
        </xs:sequence>
        <xs:attribute name="tableName" type="nameType" use="required"/>
    </xs:complexType>

    <xs:element name="row" type="rowType"/>
</xs:schema
```

A.2 XML schéma pro export logovací tabulky

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns="http://www.matasek.net/DatabaseLogger"
    targetNamespace="http://www.matasek.net/DatabaseLogger"
    elementFormDefault="qualified">

    <xs:simpleType name="nameType">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>

    <xs:simpleType name="operationType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="I"/>
            <xs:enumeration value="U"/>
            <xs:enumeration value="D"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="changedType">
        <xs:restriction base="xs:dateTime"/>
    </xs:simpleType>

    <xs:complexType name="columnType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="name" type="nameType"
                    use="required"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

    <xs:complexType name="rowType">
        <xs:sequence>
```

```
            <xs:element name="col" type="columnType" minOccurs="0"
                maxOccurs="unbounded" nillable="true"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="loggerRowType">
        <xs:sequence>
            <xs:element name="table" type="nameType"/>
            <xs:element name="user" type="nameType"/>
            <xs:element name="operation" type="operationType"/>
            <xs:element name="changed" type="changedType"/>
            <xs:element name="old_values" type="rowType"
                nillable="true"/>
            <xs:element name="new_values" type="rowType"
                nillable="true"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="queryType">
        <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:element name="row" type="loggerRowType"/>
        </xs:sequence>
    </xs:complexType>

    <xs:element name="query" type="queryType"/>
</xs:schema>
```

A.3 XML schéma pro export historického pohledu

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns="http://www.matasek.net/DatabaseLogger"
    targetNamespace="http://www.matasek.net/DatabaseLogger"
    elementFormDefault="qualified">

    <xs:simpleType name="nameType">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>

    <xs:simpleType name="dateTimeType">
        <xs:restriction base="xs:dateTime"/>
```

```
    </xs:simpleType>

    <xs:complexType name="columnType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="name" type="nameType"
                    use="required"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

    <xs:complexType name="rowType">
        <xs:sequence>
            <xs:element name="col" type="columnType" minOccurs="0"
                maxOccurs="unbounded" nillable="true"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="tableType">
        <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:element name="row" type="rowType"/>
        </xs:sequence>
        <xs:attribute name="name" type="nameType"/>
        <xs:attribute name="dateTime" type="dateTimeType"/>
    </xs:complexType>

    <xs:element name="table" type="tableType"/>
</xs:schema>
```

Příloha B

Klíčová slova pro nahrazování v SQL

COLUMN_NAME název sloupce

COLUMN_NAMES seznam názvu sloupců oddělených čárkou

COLUMN_VALUES seznam hodnot sloupců oddělených čárkou

HISTORY_TABLE_COLUMNS názvy sloupců historické tabulky oddělené čárkou

HISTORY_TRIGGER příkaz pro vložení záznamu do historické tabulky

HISTORY_TABLE_NAME název historické tabulky

LOGGING_TABLE_NAME název logovací tabulky

LOGGING_TRIGGER_LOOP_NEW kód, který zpracuje jeden záznam z nově přidaného sloupce (vytvoří fragment XML dokumentu reprezentujícího řádek tabulky)

LOGGING_TRIGGER_LOOP_OLD kód, který zpracuje jeden záznam ze smazaného sloupce (vytvoří fragment XML dokumentu reprezentujícího řádek tabulky)

PREFIX prefix logovacích objektů

PREFIX_FUNCTION_NAME název prefixové funkce

SEQUENCE_NAME název logovací sekvence

TABLE_NAME název tabulky

TRIGGER_EVENTS události spouštějící trigger

TRIGGER_NAME název triggeru

TYPE typ objektu

WHERE_CLAUSE where klauzule příkazu select

Příloha C

Popis použitých SQL dotazů

create_history_columns.sql vytvoří v tabulce historické sloupce

create_history_table.sql vytvoří historickou tabulku

create_history_trigger.sql vytvoří trigger pro logování do historické tabulky

create_logging_sequence.sql vytvoří logovací sekvenci

create_logging_table.sql vytvoří logovací tabulku

create_logging_trigger.sql vytvoří logovací trigger

create_prefix_function.sql vytvoří prefixovou funkci

drop_history_columns.sql smaže v tabulce historické sloupce

drop_history_table.sql smaže historickou tabulku

drop_history_trigger.sql smaže trigger pro logování do historické tabulky

drop_logging_sequence.sql smaže logovací sekvenci

drop_logging_table.sql smaže logovací tabulku

drop_logging_trigger.sql smaže logovací trigger

drop_prefix_function.sql smaže prefixovou funkci

get_columns.sql vrátí seznam sloupců v tabulce

get_historical_table_content.sql vrátí historická data

get_logged_columns.sql vrátí seznam logovaných sloupců

get_logging_objects.sql vrátí seznam všech objektů vytvořených logovací aplikací

get_logging_table_content.sql vrátí data z logovací tabulky

get_prefix.sql vrátí prefix logovacích objektů

get_tables.sql vrátí seznam tabulek

get_trigger_events.sql vrátí operace, které spouští logování

get_users.sql vrátí seznam uživatelů

has_column.sql vrátí sloupců daného jména

has_function.sql vrátí funkcí daného jména

has_sequence.sql vrátí počet sekvencí daného jména

has_table.sql vrátí počet tabulek daného jména

history_trigger.sql obsahuje kód pro vložení záznamu do historické tabulky

logging_trigger_loop.sql obsahuje kód pro převod hodnoty sloupce do XML reprezentace

Příloha D

Obsah přiloženého CD

- /
  - app ..... Složka s aplikací
    - DatabaseLogger.jar ..... Spustitelný JAR archiv s aplikací
  - doc ..... Složka s JavaDoc dokumentací
  - netbeans ..... Složka s projektem aplikace pro NetBeans
  - src ..... Složka se zdrojovými kódy
  - text
    - tex ..... Složka se zdrojovými soubory pro LaTeX
    - Matasek_BP.xml ..... Bakalářská práce ve formátu DocBook
    - Matasek_BP.pdf ..... Bakalářská práce ve formátu PDF