Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Michał Ziółkowski
Nr albumu: 209442

Migracja do systemu DB2 i optymalizacja bazy danych eksperymentu "π of the Sky"

Praca magisterska
na kierunku INFORMATYKA

Praca wykonana pod kierunkiem
dr. hab. Jerzego Tyszkiewicza
Instytut Informatyki

Wrzesień 2008

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data Podpis kierującego pracą

Oświadczenie autora (autorów) pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data Podpis autora (autorów) pracy

Streszczenie

W pracy przedstawiono zadania wykonane w ramach migracji bazy danych eksperymentu "π of the Sky" z systemu PostgreSQL do DB2, a także optymalizacji jej działania poprzez opracowanie i wdrożenie strategii tworzenia kopii zapasowych zapewniających bezpieczeństwo danych oraz opracowanie systemu redundancji umożliwiającego pracę bazy w przypadku awarii. Opisano kolejne etapy wykonanej pracy i przedstawiono różnice pomiędzy bazami PostgreSQL i DB2 wpływające na proces migracji danych, a także opisano sposób implementacji partycjonowanej bazy danych DB2 w ramach eksperymentu "π of the Sky" oraz opracowane systemy tworzenia kopii zapasowych i odporności na awarie.

Słowa kluczowe

PostgreSQL, DB2, migracja, π of the Sky, kopia zapasowa, redundancja, katalog gwiazd

Dziedzina pracy (kody wg programu Socrates-Erasmus)

11.3 Informatyka

Klasyfikacja tematyczna

H. Information Systems
H.2 Database management
H.2.4 Systems

Tytuł pracy w języku angielskim

Migration to DB2 and optimalization of "π of the Sky" experiment's database

Spis treści

Wprowadzenie

Eksperyment "π of the Sky" został stworzony w celu poszukiwania ciekawych zjawisk we wszechświecie. Prototypowa wersja eksperymentu znajduje się w Las Campanas w Chile [PI02], gdzie zlokalizowany jest sprzęt służący do fotografowania nieba i przetwarzania wykonanych zdjęć, którego efektem są pomiary jasności i położenia gwiazd umieszczane w bazie danych PostgreSQL. [PI01]

Aktualnie system składa się z dwóch kamer, które dostarczają ok. sześciu milionów pomiarów na dobę, co przekłada się na ok. 200GB danych rocznie. [PI03] Ponieważ planowane jest rozszerzenie systemu do trzydziestu dwóch kamer, w celu uzyskania odpowiedniej wydajności dostępu do danych zdecydowano się na migrację z silnika bazy danych PostgreSQL na silnik bazy danych DB2, który poza funkcją optymalizacji zapytań umożliwia także rozproszenie jednej bazy danych na wiele komputerów.

Oba silniki baz danych oparte są o standardowy język zapytań SQL[1], jednak ze względu na różnice w sposobie reprezentacji danych i składni poleceń konieczne jest przeniesienie aktualnej bazy do nowego rozproszonego systemu, a także dostosowanie aktualnych aplikacji do pracy z nową bazą.

Przedmiotem tej pracy jej wykonanie migracji danych z aktualnej bazy PostgreSQL do bazy DB2, a także optymalizacja jest działania poprzez zapewnienie niezawodności bazy i opracowanie systemu i strategii tworzenia kopii zapasowych, które zapewnią bezpieczeństwo danych i umożliwią szybką reakcję na awarie. Zadanie podzielono na pięć kolejnych etapów:

- stworzenie schematu bazy danych w bazie DB2, który umożliwi migrację danych z wielu ustalonych baz PostgreSQL o podobnych (ale niekoniecznie identycznych) schematach
- stworzenie analogicznych jak w bazach źródłowych indeksów w bazie DB2
- stworzenie skryptów przenoszących dane, które będą służyły zarówno do jednorazowej migracji obecnych dużych baz danych, jak i do conocnego przenoszenia danych z baz podręcznych
- opracowanie strategii wykonywania kopii zapasowych zapewniających bezpieczeństwo danych i ich odtwarzania w przypadku awarii
- opracowanie koniecznych redundancji dla zapewnienia odporności rozproszonej bazy na awarie

Praca składa się z pięciu rozdziałów i dodatków. W rozdziale 1 opisano oprogramowanie używane w czasie realizacji zadania. Rozdział 2 przestawia różnice pomiędzy sposobem reprezentacji danych w rozważanych systemach baz danych, a także opisuje stworzoną strukturę i sposób jej użycia do stworzenia nowej bazy. Skrypty migrujące dane i sposób ich działania

[1] SQL (ang. Structured Query Language) —- strukturalny język zapytań używany do tworzenia i modyfikowania baz danych oraz do umieszczania i pobierania danych z baz danych.

zostały opisane w rozdziale 3. Rozdział 4 przedstawia opracowaną strategię tworzenia kopii zapasowych, sposób ich wykonywania i odtwarzania. W rozdziale 5 przybliżono systemy zabezpieczania się przed awariami systemu i reagowania na nie. W dodatkach umieszczono schemat struktury bazy danych.

Rozdział 1

Opis oprogramowania

W projekcie wykorzystano różne rodzaje oprogramowania, które tworzą środowisko działania części bazodanowej eksperymentu "π of the Sky". Poniżej opisano najważniejsze systemy i programy, których wykorzystanie jest kluczowe dla projektu.

1.1. System operacyjny *Fedora 8*

Fedora to nazwa dystrybucji systemów operacyjnych z rodziny Linux. Powstała w 2003 roku w wyniku podziału projektu Red Hat Linux na część darmową (jaką właśnie jest Fedora) i część komercyjną Red Hat Enterprise Linux. Jest tworzona i rozwijana przez grupę *Fedora Project*, sponsorowaną przez firmę *Red Hat*. Produkty z grupy Fedora opierają się na oprogramowaniu darmowym, o ogólnie dostępnych źródłach. Cechą szczególną jest zastosowanie systemu zarządzania pakietami RPM[1]. [Fed01]

W eksperymencie "π of the Sky" zastosowano najnowszy (w chwili rozpoczęcia projektu migracji) system z rodziny Fedora – *Fedora 8*, wydany 8 listopada 2007 roku. Z uwagi na istniejące w środowisku eksperymentu 32-bitowe aplikacje zdecydowano się na wersję 32-bitową, z dodatkowym rozszerzeniem PAE[2].

1.2. System zarządzania bazą danych *PostgreSQL*

PostgreSQL to wolnodostępny system zarządzania relacyjną bazą danych. Początkowo opracowywany na Uniwersytecie Kalifornijskim w Berkeley i opublikowany pod nazwą Postgres. W miarę rozwoju i zwiększania funkcjonalności, baza danych otrzymała nazwy Postgres95 i ostatecznie PostgreSQL, aby upamiętnić pierwowzór oraz zaznaczyć zgodność ze standardem SQL. PostgreSQL zalicza się do baz typu RDBMS[3] z rozszerzeniami obiektowymi. [PSQL01]

W prototypie eksperymentu "π of the Sky" system PostgreSQL został wybrany jako narzędzie do zarządzania danymi ze względu na fakt, że jest dostępny za darmo bez ograniczeń

[1] RPM (ang. RPM Package Manager, dawniej Red Hat Package Manager) — program służący do instalacji i zarządzania pakietami zawierającymi oprogramowanie. Program ten powstał na potrzeby dystrybucji Red Hat Linux, aktualnie jest używany również w innych dystrybucjach (np. Fedora, SUSE, Mandriva, PLD).

[2] PAE (ang. Physical Address Extension) — funkcjonalność niektórych procesorów x86 i x86-64 umożliwiająca wykorzystanie więcej niż 4GB pamięci fizycznej w 32-bitowych systemach operacyjnych. Polega na dodaniu do procesora dodatkowych czterech bitów służących do wyboru adresu, dzięki czemu możliwa do zaadresowania przestrzeń zostaje zwiększona z 4GB do 64GB. Wymaga wsparcia ze strony systemu operacyjnego.

[3] RDBMS (ang. Relational Database Management System) — zestaw programów służących do korzystania z bazy danych opartej na modelu relacyjnym.

funkcjonalności. W momencie rozszerzenia systemu do 32 kamer konieczne jest uzyskanie większej wydajności bazy danych, z tego względu zdecydowano się na zmianę systemu PostgreSQL na inny, umożliwiający rozproszenie bazy danych na wiele komputerów.

1.3. System zarządzania bazą danych *IBM DB2*

DB2 to system zarządzania relacyjną bazą danych opracowany przez firmę IBM. Pierwszy raz został wydany w 1983 roku na platformę Multiple Virtual Storage. W latach dziewięćdziesiątych firma IBM opracowała wersje systemu DB2 na platformę x86, między innymi dla systemu Linux. [IBM02]

Z uwagi na dostępną w ramach systemu DB2 funkcjonalność partycjonowania bazy danych (ang. *Database Partitioning Feature*), zdecydowano się wybrać DB2 jako środowisko docelowe dla eksperymentu "π of the Sky". W pierwszej fazie wdrożenia systemu planowane jest rozproszenie bazy danych na trzy równorzędne komputery. W środowisku eksperymentu używany jest system DB2 w wersji 9.5 z zainstalowym pakietem poprawek nr 1.

1.4. Środowisko *Java Development Kit* i *Java Virtual Machine*

Java Development Kit to darmowe oprogramowanie firmy Sun Microsystems udostępniające środowisko niezbędne do programowania w języku Java. Produkt dostępny jest dla wielu systemów operacyjnych – najpopularniejsze wersje są dla systemów Solaris, Linux i Microsoft Windows. Tworzone za jego pomocą oprogramowanie można uruchamiać na wirtualnej maszynie Javy.

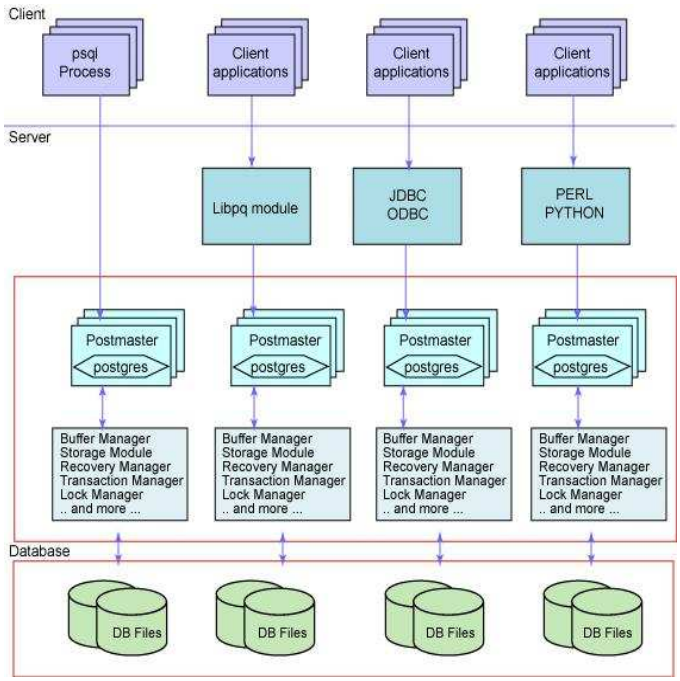
Wirtualna maszyna Javy (ang. *Java Virtual Machine*) to zależny od platformy system uruchomieniowy dla programów. Standardowo służy programom napisanym w języku Java, ale można go używać także jako środowisko uruchomieniowe także dla innych języków, np. Jython (implementacja Pythona właśnie na JVM). Programy napisane dla maszyny wirtualnej Javy są niezależne od platformy dzięki temu, że uruchamiane są bezpośrednio w JVM (po konwersji na kod pośredni, poprzez kompilator (jak javac) dla programów Javy lub w locie jak dla wspomnianego wyżej Jythona), a nie w rodzimym środowisku.

W trakcie realizacji prac potrzebowałem dodatkowego środowiska dającego więcej możliwości, niż wbudowany język systemu operacyjnego czy narzędzia dostępne razem z systemem DB2. Z uwagi na fakt, że stworzone oprogramowanie będzie uruchamiane na wielu różnych komputerach, nie powinno ono wymagać instalacji dodatkowych narzędzi do działania. Ponieważ środowisko języka Java jest wymagane do działania przez system DB2, w projekcie zdecydowałem się na jego wykorzystanie przy migracji danych. Istotny dla mnie był także fakt, że napisane w środowisku Java oprogramowanie może być łatwo uruchamiane na innych systemach operacyjnych bez konieczności rekompilacji.

Rozdział 2

Migracja struktury bazy danych eksperymentu "π of the Sky"

2.1. Różnice pomiędzy systemami *PostgreSQL* i *DB2*



Rysunek 2.1: *Architektura i procesy systemu PostgreSQL* [IBM01]

Systemy PostgreSQL i DB2 różnią się pod wieloma względami – z punktu widzenia architektury systemu, przechowywania danych, typów danych i sposobu zapewniania spójności bazy. Podstawowe różnice pomiędzy PostgreSQL i DB2 to:

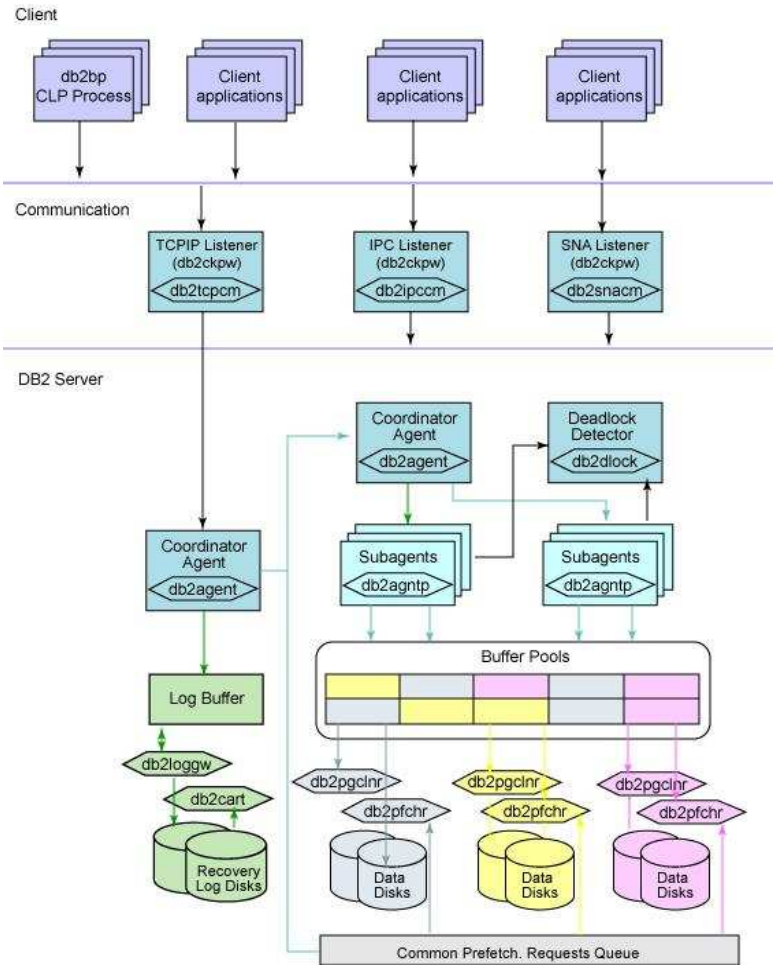
- model obszaru tabel: w PostgreSQL obszar tabel może obejmować wiele baz danych, natomiast w DB2 jest przypisany do określonej bazy danych. Oznacza to, że w bazie PostgreSQL tabele z różnych baz danych mogą być trzymane w tej samej przestrzeni tabel o tych samych parametrach, natomiast DB2 fizycznie oddziela od siebie przestrzenie tabel z różnych baz danych. [PSQL02]

- rozróżnianie wielkości liter w nazwach baz danych: PostgreSQL nie rozróżnia wielkości liter, natomiast DB2 standardowo używa dużych liter przy przechowywaniu nazw baz danych. Możliwe jest jednak użycie także małych liter poprzez umieszczenie nazwy bazy w cudzysłowach.

- kolumny typu tablicowego: DB2 nie wspiera tego typu kolumn, w przeciwieństwie do PostgreSQL

- kolumny o wartościach rosnących automatycznie: PostgreSQL definiuje typy SERIAL i BIGSERIAL, natomiast DB2 umożliwia nadanie kolumnie atrybutu IDENTITY, który umożliwia automatyczne wypełnianie kolumnny kolejnymi wartościami.

Na rysunkach 2.1 i 2.2 przedstawiono schemat architektury i procesów systemów PostgreSQL i DB2.



Rysunek 2.2: *Architektura i procesy systemu DB2* [IBM01]

Z punktu widzenia migracji danych najistotniejsze różnice leżą w typach danych definiowanych przez oba silniki baz danych. Porównanie najważniejszych typów danych i sposób ich migracji z bazy PostgreSQL do DB2 przedstawia tabela 2.1.

Tabela 2.1: Mapowanie typów danych PostgreSQL na DB2 [IBM01]

PostgreSQL	**DB2**	**Uwagi**
Bigserial, Serial8	Bigint	Należy użyć atrybutu IDENTITY
Bit	Char(n) for bit data	Dla długości do 254 bajtów
Bit varying(n) Varbit(n)	Varchar(n) for bit data	Dla wielkości do 32,672 bajtów
Bytea	Blob	Może być używane pomiędzy 32K i 2GB bajtów
Boolean	—	Należy użyć Char(1) lub Smallint
Char(n), Character (n)	Char(n)	Do 254 bajtów
Date	Date	Należy użyć rejestru CURRENT TIMEZONE aby przekształcić datę
Datetime	Timestamp	
Decimal(p,s), Numeric(p,s)	Decimal(p,s)	Dla precyzji większej niż 31 należy użyć Double
Float4, Real	Real	Można użyć także Numeric lub Float
Float8, Double Precision	Double Precision	Dla niedużych liczb i precyzji mniejszej niż 31 można użyć Numeric
Smallint	Smallint	–
Integer	Integer	–
Int8, Bigint	Bigint	–
Varchar(n) Character varying(n) Character varying	Varchar(n)	Jeśli 'n' ma 32K lub mniej DB2 wymaga podania wartości 'n', PostgreSQL tego nie wymusza
Serial, Serial4	Integer	Należy użyć atrybutu IDENTITY
Text	Varchar(n), Clob	Należy użyć CLOB jeśli długość jest większa niż 32K bajtów
Time	Time	Brak reprezentacji strefy czasowej
Timestamp	Timestamp	Brak reprezentacji strefy czasowej

2.2. Struktura bazy danych eksperymentu

W ramach projektu przeanalizowałem struktury różnych baz eksperymentu. Ponieważ bazy te w celu uzyskania większej wydajności nie mają zaimplementowanych powiązań pomiędzy tabelami (logicznie takie powiązania istnieją), możliwe było utworzenie wspólnej struktury docelowej, do której będą migrowane wszystkie istniejące bazy danych (ustaliłem, że kolumny, które nie występują we wszystkich bazach, mogą w przypadku braku danych zostać wypełnione wartościami NULL). Najważniejsze trudności, jakie pojawiły się przy tworzeniu struktury to:

1. Migracja pól typu BOOLEAN - to pole nie ma swojego odpowiednika w bazie DB2, więc konieczne było jego przemigrowanie do jednego z obsługiwanych typów. Jednak zmienne typu BOOLEAN w bazie PostgreSQL mogą przyjmować następujące wartości: TRUE, 't', 'true', 'y', 'yes', '1', FALSE, 'f', 'false', 'n', 'no', '0'. Widać zatem, że nie jest możliwe wybranie uniwersalnego odpowiednika. W celu znalezienia odpowiedniego typu przeanalizowałem bazy danych eksperymentu i sposób zapisu wartości w tych bazach. W związku z tym, że w tych bazach wartości były zapisywane za pomocą jednej litery, wybrałem jako odpowiednik BOOLEAN typ CHAR(1). Dzięki temu uniknąłem także

konieczności konwersji danych podczas migracji, co eliminuje możliwość popełnienia błędu i zmiany znaczenia danych (co w przypadku baz naukowych jest bardzo trudne do wykrycia).

2. Tworzenie kluczy głównych i unikalnych - PostgreSQL nie wymaga, aby kolumna na której tworzymy klucz główny lub unikalny miała adnotację NOT NULL. W DB2 jest to konieczne, dlatego też w wielu miejscach dodałem taką adnotację do migrowanych kolumn.

3. Podwójne indeksy na wartościach - niektóre tabele miały na wartości wyznaczonej jako klucz główny (co implikuje obecność indeksu na tej wartości) stworzony dodatkowy indeks na tej samej wartości. PostgreSQL pozwala na takie dublowanie indeksów, natomiast DB2 nie pozwala na tworzenie dwóch dokładnie takich samych indeksów. Przy migracji usunąłem ze struktury dodatkowe indeksy.

4. Migracja pól typu TIME - w bazie DB2 istnieje wprawdzie dokładny odpowiednik, jednak nie jest on tak precyzyjny jak w PostgreSQL (nie umożliwia przechowywania milisekund). Zmniejszyłem precyzję tych danych, aby dopasować ją do standardu obowiązującego w bazie DB2.

Diagram docelowej bazy został przedstawiony w dodatku A.

2.3. Pliki opisujące strukturę i ich zastosowanie do stworzenia bazy danych DB2

W ramach pracy stworzyłem następujące pliki zawierające polecenia DDL[1]:

- **pidb_empty_db2tables.sql** – plik z poleceniami tworzącymi tabele w bazie danych
- **pidb_empty_db2cons.sql** – plik z poleceniami tworzącymi indeksy w bazie danych
- **pidb_empty_db2drop.sql** – plik z poleceniami usuwajacymi tabele z bazy danych

Pliki opisane powyżej służą jednak do stworzenia obiektów w już istniejącej bazie danych, dlatego też stworzyłem dodatkowy skrypt **create_database** wykonujący następujące czynności:

1. Wywołuje polecenie DB2 *create database* w celu stworzenia bazy o podanej przez użytkownika nazwie, w podanej przez niego lokalizacji

2. Nawiązuje połączenie z nowo stworzoną bazą za pomocą polecenia DB2 *connect*

3. Tworzy pulę buforów o wielkości strony 8K (konieczne dla tabel o dużej wielkości) za pomocą polecenia DB2 *create bufferpool*

4. Tworzy przestrzeń tabel o wielkości strony 8K (konieczne dla tabel o dużej wielkości) za pomocą polecenia DB2 *create tablespace*

5. Wywołuje opisane wyżej skrypty tworzące tabele i indeksy

[1]DDL (ang. Data Definition Laguage) — język programowania służący do definiowania struktur danych, używany m.in. do definiowania struktury bazy danych w relacyjnych systemach zarządzania bazami danych. [MC01]

6. Kończy połączenie z bazą

Opisany skrypt wywołuje się podając dwa parametry: nazwę tworzonej bazy (która ze względu na ograniczenia DB2 nie może mieć więcej niż osiem znaków i musi zaczynać się od litery) i ścieżkę na dysku, gdzie będą przechowywane dane opisujące bazę. Kolejno wykonywane etapy i informacje o napotkanych błędach wypisywane są na standardowe wyjście, natomiast w pliku <*nazwa bazy*>_db2.log zapisywane są szczegółowe informacje o wszystkich czynnościach wykonywanych przez skrypt.

Rozdział 3

Migracja danych

Migracja danych z bazy PostgreSQL musi być procesem zautomatyzowanym i powtarzalnym, jako że w ramach eksperymentu do zmigrowania jest wiele baz, a dane źródłowe będą dalej zapisywane do baz podręcznych PostgreSQL i następnie migrowane do DB2. Dodatkowo konieczne jest umożliwienie użytkownikowi wyboru, z jakiego okresu dane będą migrowane. Proces migracji podzieliłem na cztery części:

1. Ustalenie zakresu danych eksportowanych z bazy źródłowej
2. Eksport danych z bazy źródłowej
3. Import danych do bazy docelowej
4. Dostosowanie atrybutów bazy docelowej do załadowanych danych

W pierwszych dwóch etapach przy eksportowaniu informacji z bazy PostgreSQL wykorzystałem stworzone przez firmę IBM i dostępne za darmo narzędzie *IBMExtract.jar*, dostępne na stronie: http://www.ibm.com/developerworks/db2/library/techarticle/dm-0606khatri/

3.1. Konfiguracja

Wszystkie narzędzia służące do migracji danych korzystają ze wspólnego pliku konfiguracyjnego o nazwie **config**. Zawiera on następujące pozycje:

- **JAVA_HOME** – określa ścieżkę na dysku, pod którą zainstalowano *Java Development Kit*. Parametr ten należy odpowiednio ustawić w zależności od środowiska, w którym się pracuje.
- **DB2SCHEMA** – określa schemat w bazie DB2, pod którym umieszczone są tabele docelowe (standardowo tabele umieszczane są w schemacie MAIN).
- **COLSEP** – definiuje separator kolumn w eksportowanych plikach z danymi
- **NUM_THREADS** – określa liczbę tworzonych wątków, które będą eksportować dane
- **SERVER** – zawiera adres serwera, na którym położona jest źródłowa baza danych
- **PORT** – określa port serwera, pod którym można połączyć się z bazą PostgreSQL
- **DBUID** – określa nazwę użytkownika, który ma dostęp do źródłowej bazy

- **DBPWD** – zawiera hasło użytkownika z pozycji **DBUID**
- **TABLELIST** – definiuje nazwę pliku z listą tabel, z których będą wyciągane dane; plik ten jest tworzony podczas pierwszego etapu migracji
- **GENDDL** – określa, czy podczas eksportu ma zostać stworzony plik z poleceniami DDL tworzącymi tabele źródłowe
- **UNLOAD** – określa, czy podczas eksportu mają zostać pobrane dane
- **FETCHSIZE** – definiuje ilość jednocześnie czytanych wierszy danych.

Wczytanie pliku **config** powoduje dodanie do środowiska uruchomieniowego ścieżek do *Java Development Kit*, sterowników JDBC[1] do bazy PostgreSQL i narzędzia *IBMExtract.jar*.

3.2. Ustalenie zakresu eksportowanych danych

Dla pierwszego etapu migracji stworzyłem skrypt **geninput**, któremu należy podać jako parametr nazwę bazy źródłowej. W wyniku jego działania otrzymujemy plik z danymi wejściowymi dla drugiego etapu migracji, zawierający w każdym wierszu nazwę tabeli i standardowe zapytanie w języku SQL opisujące polecenie pobrania wszystkich danych z tej tabeli. Poprzez modyfikację tego pliku (usuwanie wierszy i modyfikację zapytań SQL) możemy dowolnie definiować zakres eksportowanych danych.

3.3. Eksport danych

Drugi etap migracji polega na pobraniu danych z bazy źródłowej i zapisaniu ich na dysku. W pierwszej fazie oprogramowanie pobiera dane z każdej tabeli i umieszcza je w osobnych plikach, w których kolejne wiersze są zapisywane w kolejnych liniach, a kolumny są oddzielone zdefiniowanym w pliku konfiguracyjnym separatorem. Następnie generowane są skrypty z poleceniami DB2 służącymi do:

- ładowania danych do tabel
- podliczenia ilości wierszy w tabelach
- sprawdzenia stanu tabel po ładowaniu.

Skrypty te są przy używane fazie importu danych.

Do wykonania eksportu danych służy skrypt **export_pg**, który wymaga podania jako parametru nazwy bazy źródłowej. Na tym etapie jest też wpisywana do plików wynikowych nazwa schematu, do którego będą importowane dane (parametr ten definiuje się w pliku konfiguracyjnym).

[1] JDBC (ang. *Java DataBase Connectivity*) — interfejs programowania opracowany w 1996 r. przez Sun Microsystems, umożliwiający niezależnym od platformy aplikacjom napisanym w języku Java porozumiewać się z bazami danych za pomocą języka SQL. Interfejs ten jest odpowiednikiem standardu ODBC opracowanego przez SQL Access Group. [JV01]

3.3.1. Pobieranie danych do dopisania

Skrypt **export_pg** tworzy polecenia ładujące dane do pustej bazy, kasując najpierw istniejące już w tabelach wiersze (jest to wymagane do wygenerowania statystyk). Na potrzeby eksperymentu wymagana jest także możliwość dopisania eksportowanych danych do istniejących baz.

W tym celu stworzyłem drugie narzędzie do eksportu danych – skrypt **export_part_pg**. Zamiast pliku wynikowego z pierwszego etapu, korzysta on ze specjalnie przygotowanego pliku z listą tabel **partial.tables**, który zawiera tylko cztery tabele z pomiarami: FRAME, FRAME_DET, MEASUREMENTS i STARS (po konsultacji z zespołem "π of the Sky" ustaliłem, że tylko te tabele powinny być kopiowane przy częściowej migracji, jako że właśnie w nich przechowywane są nowe pomiary). Skrypt za pomocą opcjonalnych parametrów umożliwia ustalenie zakresu czasowego eksportowanych danych z pierwszych trzech tabel. W celu zachowania spójności danych konieczne jest posiadanie w bazie informacji o wszystkich gwiazdach z tabeli STARS, których pomiary mamy w pozostałych tabelach, dlatego też dodałem tutaj funkcję, która sprawdza czy wszystkie dane z tej tabeli są już przemigrowane i w razie potrzeby przenosi brakujące wiersze. Z tego powodu też przy uruchamianiu narzędzia wymagane jest podanie dodatkowego parametru oznaczającego docelową bazę DB2, ponieważ skrypt pobiera z niej informacje konieczne do ustalenia zakresu eksportowanych danych.

3.4. Import danych

Wspólnym narzędziem do importu dla obu sposobów pobierania danych jest skrypt **import_data**, któremu podaje się jako parametry nazwę źródłowej i docelowej bazy. Ponieważ jeden ze sposobów eksportu danych generuje skrypty przygotowane do wyczyszczenia bazy i ładowania danych do pustej struktury, istotne było dla mnie zabezpieczenie skryptu przed jego przypadkowym wykorzystaniem do usunięcia używanych danych. W związku z tym w pierwszej fazie jego działania sprawdzane jest, czy importowane dane są ustawione w tryb nadpisywania i czy w docelowej bazie są już jakieś dane. Jeśli oba warunki są spełnione, wyświetlane jest ostrzeżenie i program kończy się (sprawdzanie to można pominąć uruchamiając skrypt z opcją -f). W przeciwnym wypadku uruchamiane są stworzone podczas eksportu skrypty ładujące dane do bazy.

Informacje ogólne odnośnie powodzenia migracji każdej z tabel są zapisywane do pliku <*nazwa bazy DB2*>_db2.log, natomiast szczegółowe dane odnośnie etapów ładowania danych, błędów i ostrzeżeń są umieszczane w katalogu <*nazwa bazy DB2*>_msg w miejscu, gdzie przechowywane są pliki z danymi.

3.5. Dostosowanie atrybutów bazy docelowej

Ostatnim etapem migracji jest poprawne ustawienie atrybutów bazy DB2 do przemigrowanych danych typu SERIAL, którym w DB2 odpowiadają pola typu INTEGER z atrybutem IDENTITY, którego wartość niestety nie jest ustawiana automatycznie w momencie importu danych. Zadanie to wykonuje skrypt **fix_identities**, który jest wywoływany automatycznie po zakończeniu migracji. W razie potrzeby można go także uruchamiać ręcznie podając jako parametr nazwę bazy DB2 – zawsze jako wartość IDENTITY zostanie ustawiona największa wartość w danej kolumnie.

3.6. Usuwanie danych z baz PostgreSQL

Niektóre z baz PostgreSQL będą funkcjonować także po przemigrowaniu z nich danych jako bazy podręczne. Po zakończeniu migracji, w celu zmniejszenia ich objętości, istnieje możliwość usunięcia części danych. Jako że ponad 95% objętości bazy stanowi tabela MEASUREMENTS, ustaliłem z zespołem "π of the Sky", że tylko dane z tej tabeli będą usuwane. Do tego celu służy polecenie **delete_measurements**, które wymaga podania w parametrach nazwy bazy PostgreSQL, zakresu danych do usunięcia i nazwy bazy DB2, do której dane zostały zmigrowane. Skrypt przed usunięciem weryfikuje, czy w bazie DB2 są dane z wybranego zakresu. W celu zabezpieczenia przed skasowaniem danych niezmigrowanych, kasowane są tylko wiersze z datą najpóźniej o dzień wcześniejszą, niż dane obecne w DB2.

Rozdział 4

Tworzenie i odtwarzanie kopii zapasowych

Aby ochronić bazę eksperymentu przez utratą danych w wyniku awarii sprzętu, błędu ludzkiego czy innych nieprzewidzianych wydarzeń, konieczne jest opracowanie strategii tworzenia kopii bezpieczeństwa. Poniżej opisałem typowe rodzaje kopii zapasowych, możliwości ich tworzenia, strategię ochrony danych opracowaną dla bazy eksperymentu i sposób jej stosowania.

4.1. Rodzaje kopii zapasowych

4.1.1. Kopia pełna

Kopia pełna polega na utworzeniu zbioru zawierającego komplet informacji znajdujących się w bazie. Kopie pełne są najłatwiejsze w użyciu podczas odzyskiwania danych, ponieważ wymagają jedynie posiadania utworzonego zbioru. Wykonywanie kopii pełnych zajmuje najwięcej przestrzeni na nośnikach i zazwyczaj trwa najdłużej.

4.1.2. Kopia przyrostowa

Kopia przyrostowa polega na utworzeniu zbioru zawierającego jedynie te dane, które zostały utworzone lub zmienione od czasu utworzenia ostatniej (dowolnej) kopii bazy. Pozwala to na maksymalne skrócenie czasu potrzebnego do stworzenia kopii zapasowej. Przed utworzeniem pierwszej kopii przyrostowej trzeba utworzyć kopię pełną. Przy odtwarzaniu z kopii przyrostowej konieczne jest najpierw odtworzenie ostatniej kopii pełnej, a następnie wszystkich wykonanych po niej kopii przyrostowych w porządku chronologicznym, aż do odtworzenia wybranej kopii bazy.

4.1.3. Kopia różnicowa

Kopia różnicowa polega na utworzeniu zbioru zawierającego jedynie te dane, które zostały utworzone lub zmienione od czasu utworzenia ostatniej kopii pełnej. Pozwala to skrócić czas konieczny do jej utworzenia, jednak zwykle jest on dłuższy niż przy tworzeniu kopii przyrostowej (zależy to od wybranej strategii ochrony danych). Przed utworzeniem pierwszej kopii różnicowej konieczne jest wykonanie kopii pełnej. Przy odtwarzaniu z kopii różnicowej konieczne jest najpierw odtworzenie ostatniej kopii pełnej, a następnie wybranej kopii różnicowej.

Różnica pomiędzy kopią przyrostową i różnicową polega na wyborze danych, które są zapisywane. Przy pierwszym rodzaju kopii zapisywane są wszystkie dane zmienione od momentu wykonania ostatniej kopii bezpieczeństwa, natomiast w drugim przypadku nie są brane pod uwagę wszelkie stworzone kopie przyrostowe, a wybierane są dane zmienione od momentu wykonania ostatniej kopii pełnej lub różnicowej.

4.2. Tryby tworzenia kopii zapasowych

Kopia zapasowa może być utworzona w trybie *offline* lub *online*. W pierwszym przypadku w trakcie wykonywania kopii aplikacje i procesy nie mają możliwości uzyskania dostępu do bazy danych i jej obiektów. W przypadku kopii w trybie *online* dostęp do bazy nie jest blokowany i aplikacje mogą swobodnie odczytywać i zapisywać dane w bazie, jednak po odtworzeniu tak wykonanej kopii zapasowej baza znajduje się w stanie niespójnym i konieczne jest odtworzenie dzienników transakcji, które były aktywne w trakcie trwania operacji wykonywania kopii.

4.3. Dzienniki transakcji

Każda operacja wykonywana na bazie danych jest najpierw zapisywana w **dziennikach transakcji**. Umożliwiają one zachowanie spójności bazy danych w przypadku awarii, mogą też służyć do ponownego wykonania kolejnych transakcji w przypadku odtwarzania bazy od stanu po wykonaniu kopii do stanu późniejszego.

Przy odtwarzaniu kopii zapasowych w trybie *online* dzienniki transakcji odgrywają bardzo istotną rolę: w czasie tworzenia kopii użytkownicy mogą korzystać z bazy i zmieniać istniejące w niej dane, dlatego też po odtworzeniu bazy jest ona w stanie niespójnym i konieczne jest wykorzystanie dzienników transakcji do odtworzenia operacji aktywnych w czasie tworzenia kopii i doprowadzenia bazy do stanu spójnego.

W systemie DB2 istnieją trzy rodzaje dzienników transakcji:

- aktywne – zawierają niezatwierdzone lub nieeksternalizowane (nie zapisane fizycznie do bazy) transakcje
- archiwalne online – zawierają zatwierdzone i eksternalizowane transakcje, ale mimo to dzienniki te są jeszcze przechowywane w katalogu aktywnych dzienników transakcji
- archiwalne offline – zawierają zatwierdzone i eksternalizowane transakcje, w przeciwieństwie do dzienników online zostały już skopiowane z katalogu aktywnych dzienników transakcji na inny nośnik

Przy tworzeniu kopii bazy w trybie *online* jest możliwe dołączenie do stworzonej kopii bazy dzienników transakcji aktywnych podczas jej tworzenia, aby umożliwić odtworzenie bazy do stanu spójnego nawet w momencie posiadania samej kopii zapasowej. [Cho06]

4.4. Strategia tworzenia kopii zapasowych

Przy opracowywaniu strategii tworzenia kopii zapasowych starałem się znaleźć schemat, który w jak najmniejszym stopniu wykorzystuje przestrzeń dyskową, jednak zapewnia bezpieczeństwo danych. Istotny był dla mnie także czas potrzebny na stworzenie kopii, który z uwagi na rozmiar baz danych eksperymentu jest dosyć duży. W związku z powyższym zdecydowałem się na maksymalne ograniczenie ilości pełnych kopii zapasowych, dzięki czemu ilość zajętego

miejsca na dyskach nie jest duża, a tworzenie kopii przyrostowych i różnicowych przebiega znacznie szybciej niż w przypadku kopii pełnych.

Opracowana strategia zakłada wykonywanie pełnej kopii zapasowej raz na miesiąc, cotygodniowe tworzenie kopii różnicowej i codzienne wykonywanie kopii przyrostowej. Po każdym poprawnym utworzeniu kopii różnicowej będą usuwane kopie przyrostowe i różnicowe będące starsze od właśnie wykonanej o więcej niż tydzień, natomiast po każdym wykonaniu kopii pełnej będą usuwane wszystkie kopie starsze od niej o więcej niż miesiąc.

Dzięki takiej strategii zawsze będzie dostępna do odtworzenia nie tylko aktualna kopia danych, ale też historia co najmniej na miesiąc wstecz, a ilość kopii pełnych (zajmujących najwięcej przestrzeni dyskowej) zostanie zminimalizowana. Szczegółowy plan tworzenia kopii zapasowych przedstawia tabela 4.1.

Tabela 4.1: Schemat kopii zapasowych w zależności od dnia tygodnia

Dzień tygodnia Numer tygodnia	Pon	Wt	Śr	Czw	Pt	Sob	Ndz
Tydzień 1	Pełna	Przyr.	Przyr.	Przyr.	Przyr.	Przyr.	Przyr.
Tydzień 2	Różnicowa	Przyr.	Przyr.	Przyr.	Przyr.	Przyr.	Przyr.
Tydzień 3	Różnicowa	Przyr.	Przyr.	Przyr.	Przyr.	Przyr.	Przyr.
Tydzień 4	Różnicowa	Przyr.	Przyr.	Przyr.	Przyr.	Przyr.	Przyr.

Cały proces tworzenia kopii zapasowej powinien przebiegać automatycznie, co pozwala zabezpieczyć się przed błędami ludzkimi i zapewnić spójną i możliwą do odtworzenia historię kopii zapasowych. Działanie systemu powinno być także okresowo monitorowane, aby weryfikować jego działanie i reagować w przypadku pojawienia się problemów (np. braku miejsca na dysku). Z uwagi na charakterystyczny dla eksperymentu "π of the Sky" model obciążenia bazy (dane ładowane są jedynie w nocy, w dużych ilościach), proces tworzenia kopii zapasowej powinien rozpoczynać się jak najwcześniej po zakończeniu ładowania danych. Ponieważ tworzenie kopii (szczególnie pełnej) dla dużej bazy może potrwać długo, wszystkie kopie zapasowe będą tworzone w trybie *online*, aby umożliwić użytkownikom nieprzerwane korzystanie z informacji w bazie danych.

4.5. Sposób wykonywania kopii zapasowej

4.5.1. Parametry konieczne do wykonywania kopii bazy

Przed rozpoczęciem procesu tworzenia kopii zapasowych konieczne jest odpowiednie skonfigurowanie bazy, aby możliwe było wykonywanie kopii w trybie *online* i kopii przyrostowych. Wszystkie opisane parametry modyfikuje się poleceniem:

```
update db config for <nazwa bazy> using <parametr> <wartość>
```

Konieczne jest ustawienie następujących parametrów:

- **logarchmeth1** – należy ustawić wartość `disk:<ścieżka>`, gdzie ścieżka oznacza katalog archiwalny dzienników transakcji; to ustawienie powoduje zapisywanie archiwalnych dzienników transakcji do podanego katalogu (standardowo archiwalne dzienniki transakcji są usuwane)

- **trackmod** – należy ustawić wartość `ON`; parametr ten uaktywnia śledzenia zmian i pozwala na wykonywanie kopii przyrostowych

Po ustawieniu powyższych parametrów konieczne jest wykonanie pierwszej kopii zapasowej w trybie *offline*, która posłuży za punkt odniesienia. Wykonuje się ją następującym poleceniem:

```
backup db <nazwa bazy> on all dbpartitionnums to <ścieżka>
```

W powyższym poleceniu <ścieżka> oznacza katalog docelowy, w którym będzie stworzona kopia zapasowa.

4.5.2. Wykonywanie kopii bazy w środowisku rozproszonym

Kopię bazy danych w środowisku rozproszonym wykonuje się wywołując na głównym węźle bazy polecenie:

```
backup db <nazwa bazy> on all dbpartitionnums online <rodzaj kopii> ...
...to <ścieżka> compress include logs
```

Komenda ta powoduje wykonanie kopii zapasowej na wszystkich węzłach należących do bazy. Jako parametr <rodzaj kopii> możemy podać trzy wartości:

- brak wartości – oznacza kopię pełną
- INCREMENTAL – oznacza kopię różnicową
- INCREMENTAL DELTA – oznacza kopię przyrostową

Parametr <ścieżka> oznacza docelowy katalog, w którym będzie stworzona kopia zapasowa.

4.6. Sposób odtwarzania kopii zapasowej

Odtwarzanie kopii zapasowej w środowisku rozproszonym dzieli się na trzy etapy:

1. Odtworzenie kopii zapasowej na serwerze głównym
2. Odtworzenie kopii zapasowej na pozostałych serwerach
3. Odtworzenie zmian zawartych w dziennikach transakcji

Pierwsze dwa kroki wykonuje się za pomocą polecenia:

```
restore db <nazwa bazy> taken at <identyfikator> logtarget <ścieżka>
```

W powyższym poleceniu <identyfikator> oznacza numer unikalny dla kopii zapasowej, który tworzony jest z daty i godziny w momencie wykonywania kopii (numer ten występuje w nazwie pliku kopii zapasowej), natomiast parametr <ścieżka> oznacza katalog, w którym mają być zapisane dzienniki transakcji odtwarzane z kopii zapasowej. Przy odtwarzaniu kopii przyrostowej lub różnicowej po nazwie bazy należy dodać parametr INCREMENTAL AUTOMATIC.

Ostatni krok polega na odtworzeniu dzienników transakcji i wykonuje się go następującym poleceniem:

```
rollforward db <nazwa bazy> to end of logs on all dbpartitionnums and complete
```

Rozdział 5

Odporność na awarie

W ramach eksperymentu "π of the Sky" co noc zbierane są i dodawane do bazy danych nowe pomiary gwiazd. Dlatego też w wypadku awarii jednego z węzłów rozproszonej bazy wymagane jest jak najszybsze przywrócenie funkcjonalności. W ramach pracy przeanalizowałem dostępne możliwości i opracowałem procedurę reakcji na awarie, która maksymalnie zmniejsza czas przestoju bazy. Poniżej opisałem możliwości, jakie istnieją w DB2 w celu zapewnienia odporności na awarie.

5.1. Konfiguracja rozproszonej bazy eksperymentu

Serwery bazy danych eksperymentu znajdują się w miejscu uniemożliwiającym szybką naprawę uszkodzonego sprzętu, dlatego też konieczne jest zapewnienie wysokiej dostępności bazy poprzez zastąpienie uszkodzonego węzła zapasowym. W tym celu istotne jest zrozumienie działania i sposobu komunikacji rozproszonej bazy. Poniżej opisałem konfigurację węzłów bazy:

- Na jednym z węzłów znajduje się katalog domowy użytkownika, który jest właścicielem rozproszonej instancji bazy. Katalog ten jest eksportowany poprzez system NFS do pozostałych węzłów jako katalog domowy użytkownika o tej samej nazwie, co w węźle eksportującym. Dzięki temu na każdym z węzłów istnieje ten sam użytkownik, mający dostęp do tych samych informacji o instancji.

- Wszystkie węzły należące do instancji zapisane są w pliku `db2nodes.cfg` w katalogu domowym instancji.

- Połączenie między węzłami odbywa się poprzez SSH. Usługa jest skonfigurowana tak, aby umożliwiać właścicielowi instancji zdalne wykonywanie poleceń na wszystkich węzłach należących do instancji, bez konieczności podawania hasła użytkownika (uwierzytelniany jest tylko serwer, za pomocą klucza publicznego).

- Do połączenia między węzłami używany jest protokół TCP/IP, z wyznaczonym osobnym portem dla każdego z węzłów.

5.2. Odtworzenie węzła na serwerze zapasowym

Opracowana przeze mnie poniższa procedura działania ma dużą zaletę: nie wymaga stosowania drogiej pamięci masowej, aby umożliwić odtworzenie działania bazy. Jej wadą natomiast

jest nawet kilkugodzinny, w zależności od ilości danych, czas przestoju w razie awarii, ponieważ informacje odtwarzane są z kopii zapasowej dopiero po wystąpieniu awarii.

Założeniem tej metody jest istnienie serwera zapasowego, który zastępuje uszkodzony węzeł. Serwer ten eksportuje poprzez NFS do węzłów bazy dwa katalogi:

- backup – w którym będą przechowywane kopie zapasowe bazy
- logs – w którym będą przechowywane archiwalne dzienniki transakcji

Wymaganiem tej metody jest regularne tworzone kopii zapasowych wszystkich węzłów i zapisywanie ich na serwerze zapasowym (tak jak to opisano w rozdziale 4.5.2), a także ustawienie bazy do przechowywania na nim archiwalnych dzienników transakcji (co zostało opisane w rozdziale 4.5.1).

W poleceniu pierwszym parametr `<ścieżka>` oznacza katalog, w którym przechowywane będą archiwalne dzienniki transakcji. Po wykonaniu powyższej procedury konieczne jest jednorazowe wykonanie kopii zapasowej w trybie offline.

Serwer zapasowy powinien być także skonfigurowany do komunikacji z węzłami bazy tak samo, jak wszystkie pozostałe węzły.

W razie awarii jednego z serwerów bazy procedura odtwarzania jest następująca:

1. Odłączenie od bazy wszystkich aplikacji poleceniem `db2 force application all`
2. Zatrzymanie wszystkich działających węzłów bazy poleceniem `db2stop`
3. Fizyczne usunięcie ze wszystkich działających węzłów plików należących do bazy danych (w tym celu warto przechowywać bazę danych w osobnym katalogu)
4. Edycja pliku `db2nodes.cfg` w celu zamiany adresu uszkodzonego węzła na adres serwera zapasowego
5. Uruchomienie wszystkich węzłów bazy poleceniem `db2start`
6. Usunięcie danych o poprzedniej wersji bazy z katalogu systemowego poleceniem `uncatalog db <nazwa bazy>`
7. Odtworzenie bazy z kopii zapasowej
8. Odtworzenie za pomocą logów archiwalnych transakcji wykonanych od momentu stworzenia kopii zapasowej

Sytuacją szczególną jest awaria węzła, który jest właścicielem instancji rozproszonej - w wyniku tego tracimy dostęp do danych instancji eksportowanych poprzez NFS. W tym celu wymagane jest, aby wcześniej na serwerze zapasowym była stworzona osobna instancja o wszystkich parametrach takich, jak instancja przechowująca bazę. Dzięki temu w wypadku awarii możemy w miejsce eksportowanego katalogu instacji z serwera głównego podłączyć katalog z serwera zapasowego. W takim wypadku możemy też przy odtwarzaniu pominąć usunięcie informacji o bazie z katalogu systemowego.

5.3. Klastry wysokiej dostępności

Alternatywną metodą na zapewnienie odporności na awarie jest stworzenie klastra wysokiej dostępności. Klaster taki składa się z dwóch lub więcej serwerów ze wspólną pamięcią

dyskową i dodatkowym połączeniem sieciowym pomiędzy nimi umożliwiającym ciągłe monitorowanie wzajemnego działania. W takiej konfiguracji jeden z serwerów jest awaryjnym, natomiast pozostałe odpowiadają za pracę bazy danych. W wypadku awarii serwer awaryjny automatycznie przejmuje adres IP uszkodzonego serwera i umożliwia dostęp do bazy przy minimalnym wpływie na korzystające z niej aplikacje. [Red01]

Z uwagi na niewielką przerwę w działaniu i funkcjonalność przejmowania zasobów serwera uszkodzonego przez serwer zapasowy, rozwiązanie takie jest bardzo skuteczne, ponieważ umożliwia praktycznie nieprzerwany dostęp do bazy danych i nie wymusza konieczności zmiany adresu serwera w aplikacjach korzystających z bazy. Dodatkowo mechanizm wysokiej dostępności jest natywnie wspierany przez system operacyjny *Fedora 8* i przez system DB2. [HA01] [Red01]

Wadą tego rozwiązania jest konieczność posiadania wspólnej pamięci dyskowej, co znacznie zwiększa koszt sprzętu koniecznego do jego wdrożenia. Zmniejsza to także skalowalność rozwiązania, ponieważ wąskim gardłem staje się pamięć dyskowa, która posiada określoną przepustowość do przesyłania danych do serwerów.

5.4. Odporność na awarie w "π of the Sky"

Metodą wybraną do stosowania w ramach bazy eksperymentu jest odtwarzanie uszkodzonego węzła na serwerze zapasowym. Mimo dużo dłuższego czasu, jaki jest konieczny do przywrócenia bazy do stanu sprzed awarii jest to metoda zdecydowanie bardziej ekonomiczna. Za jej wdrożeniem przemawiają następujące fakty:

1. Na miejscu działania eksperymentu działają bazy podręczne, które w razie awarii głównej bazy mogą przez jakiś czas kumulować dane aż do przywrócenia funkcjonalności
2. Brak możliwości wykonywania obliczeń w sytuacjach krytycznych przez okres kilku godzin nie wpłynie znacząco na powodzenie eksperymentu
3. Metoda ta zapewnia dużo większą skalowalność niż klaster wysokiej dostępności, ponieważ do zwiększenia wydajności konieczne jest tylko dodanie kolejnego węzła.

Rozdział 6

Podsumowanie

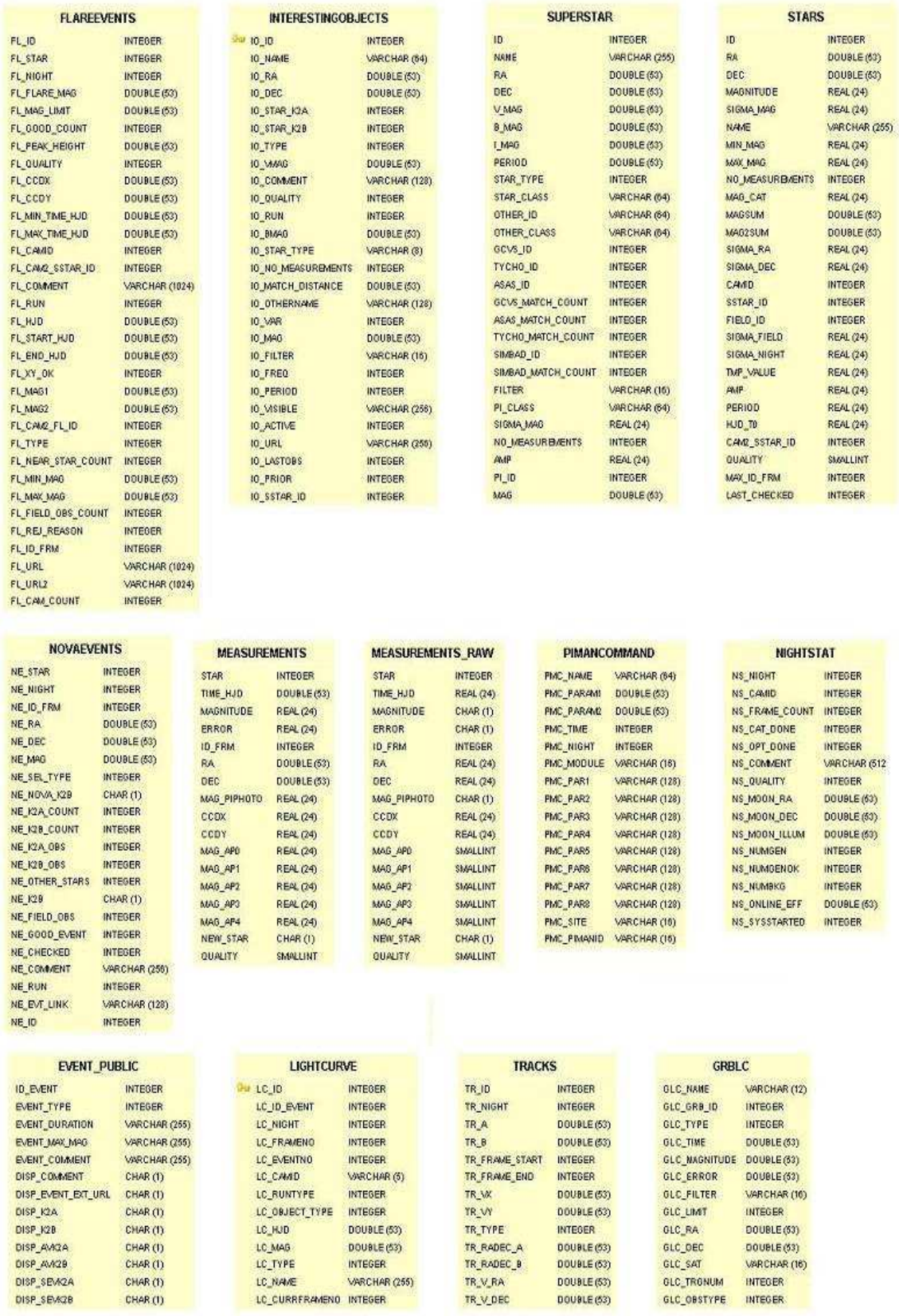
W pracy przedstawiono opis zadań wykonanych w ramach migracji bazy eksperymentu "π of the Sky" z systemu PostgreSQL do środowiska rozproszonego DB2. Dzięki przeniesieniu bazy do wydajniejszej architektury, umożliwiono obsługę danych ze znacznie większego systemu złożonego z 32 kamer. Taki system ma dużo większe wymagania wydajnościowe, którym nie mogłaby sprostać baza danych oparta o system PostgreSQL.

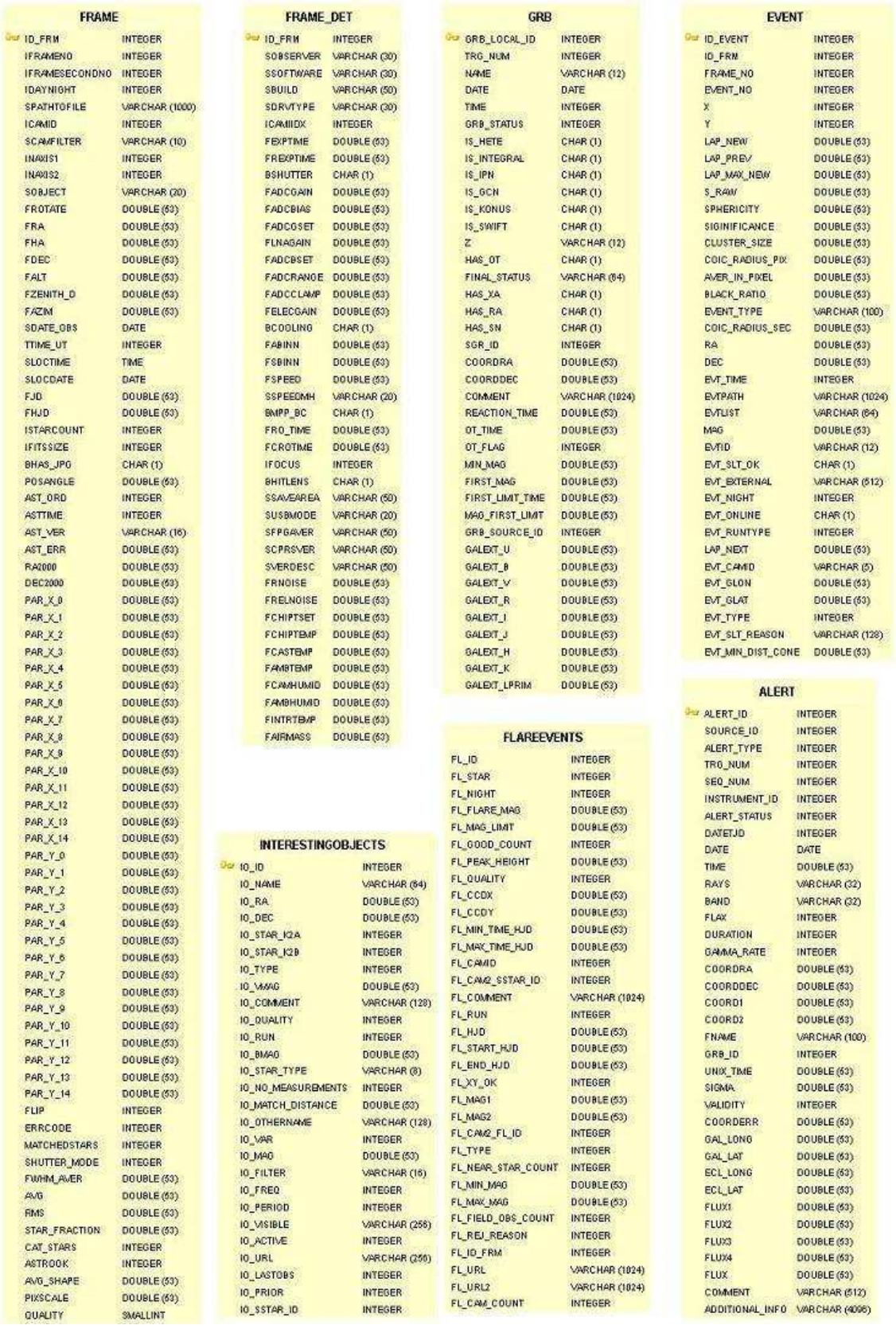
Dzięki efektom tej pracy będzie możliwe wykonywanie dokładnej analizy dużego obszaru nieba. Umożliwi to dokonywanie interesujących odkryć w dziedzinie astrofizyki procesów szybkozmiennych. Do korzyści, które będą efektem wdrożenia nowej architektury można zaliczyć:

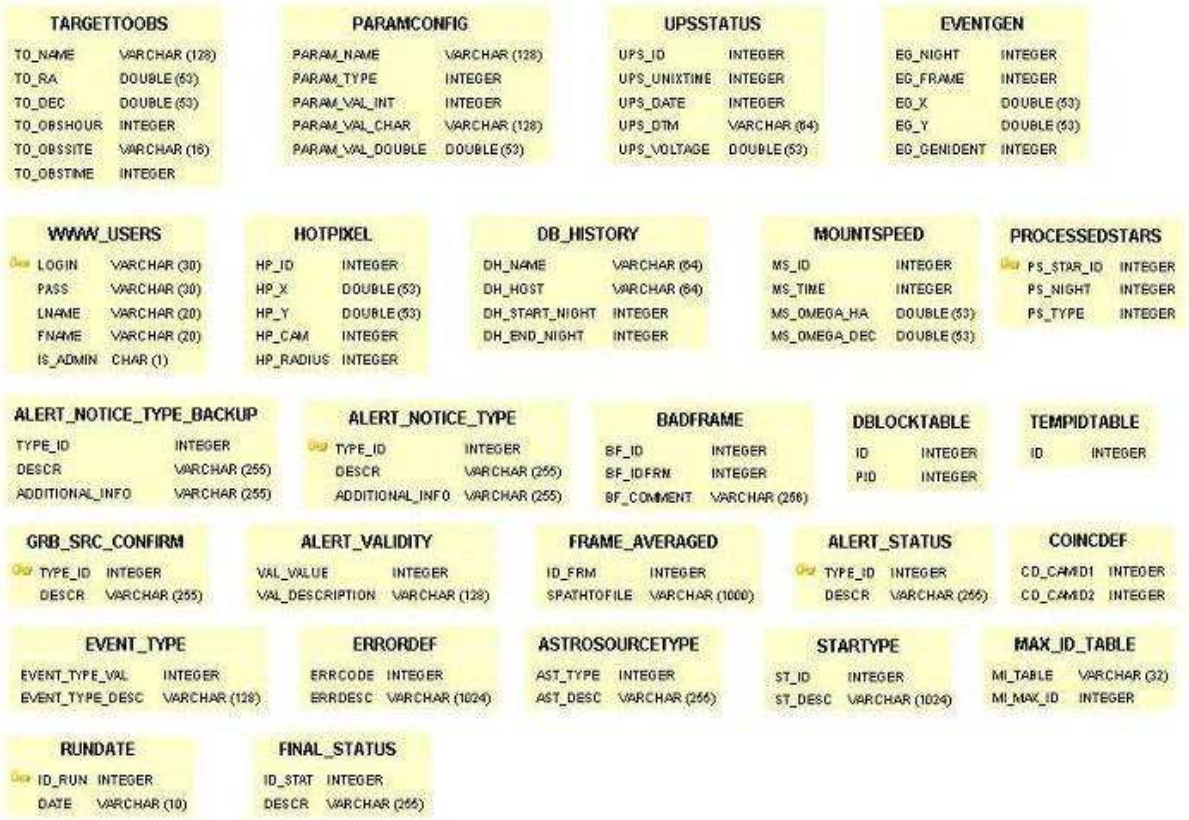
- odkrywanie gwiazd nowych
- możliwość automatycznego wykrywania krótkotrwałych zdarzeń kosmicznych, jak np. narodziny czarnej dziury
- stworzenie pełnej bazy danych gwiazd obserwowanych przez system "π of the Sky"
- możliwość łatwego i efektywnego udostępnienia pomiarów jasności do badań naukowych ogromnej liczby obiektów astrofizycznych.

Dodatek A

Diagram docelowej struktury bazy eksperymentu


FLAREEVENTS
FL_ID INTEGER
FL_STAR INTEGER
FL_NIGHT INTEGER
FL_FLARE_MAG DOUBLE (53)
FL_MAG_LIMIT DOUBLE (53)
FL_GOOD_COUNT INTEGER
FL_PEAK_HEIGHT DOUBLE (53)
FL_QUALITY INTEGER
FL_CCDX DOUBLE (53)
FL_CCDY DOUBLE (53)
FL_MIN_TIME_HJD DOUBLE (53)
FL_MAX_TIME_HJD DOUBLE (53)
FL_CAMID INTEGER
FL_CAM2_SSTAR_ID INTEGER
FL_COMMENT VARCHAR (1024)
FL_RUN INTEGER
FL_HJD DOUBLE (53)
FL_START_HJD DOUBLE (53)
FL_END_HJD DOUBLE (53)
FL_XY_OK INTEGER
FL_MAG1 DOUBLE (53)
FL_MAG2 DOUBLE (53)
FL_CAM2_FL_ID INTEGER
FL_TYPE INTEGER
FL_NEAR_STAR_COUNT INTEGER
FL_MIN_MAG DOUBLE (53)
FL_MAX_MAG DOUBLE (53)
FL_FIELD_OBS_COUNT INTEGER
FL_REJ_REASON INTEGER
FL_ID_FRM INTEGER
FL_URL VARCHAR (1024)
FL_URL2 VARCHAR (1024)
FL_CAM_COUNT INTEGER
INTERESTINGOBJECTS
IO_ID INTEGER
IO_NAME VARCHAR (64)
IO_RA DOUBLE (53)
IO_DEC DOUBLE (53)
IO_STAR_K2A INTEGER
IO_STAR_K2B INTEGER
IO_TYPE INTEGER
IO_VMAG DOUBLE (53)
IO_COMMENT VARCHAR (128)
IO_QUALITY INTEGER
IO_RUN INTEGER
IO_BMAG DOUBLE (53)
IO_STAR_TYPE VARCHAR (8)
IO_NO_MEASUREMENTS INTEGER
IO_MATCH_DISTANCE DOUBLE (53)
IO_OTHERNAME VARCHAR (128)
IO_VAR INTEGER
IO_IMAG DOUBLE (53)
IO_FILTER VARCHAR (16)
IO_FREQ INTEGER
IO_PERIOD INTEGER
IO_VISIBLE VARCHAR (256)
IO_ACTIVE INTEGER
IO_URL VARCHAR (256)
IO_LASTOBS INTEGER
IO_PRIOR INTEGER
IO_SSTAR_ID INTEGER
SUPERSTAR
ID INTEGER
NAME VARCHAR (255)
RA DOUBLE (53)
DEC DOUBLE (53)
V_MAG DOUBLE (53)
B_MAG DOUBLE (53)
I_MAG DOUBLE (53)
PERIOD DOUBLE (53)
STAR_TYPE INTEGER
STAR_CLASS VARCHAR (64)
OTHER_ID VARCHAR (64)
OTHER_CLASS VARCHAR (64)
GCVS_ID INTEGER
TYCHO_ID INTEGER
ASAS_ID INTEGER
GCVS_MATCH_COUNT INTEGER
ASAS_MATCH_COUNT INTEGER
TYCHO_MATCH_COUNT INTEGER
SIMBAD_ID INTEGER
SIMBAD_MATCH_COUNT INTEGER
FILTER VARCHAR (16)
PI_CLASS VARCHAR (64)
SIGMA_MAG REAL (24)
NO_MEASUREMENTS INTEGER
AMP REAL (24)
PI_ID INTEGER
MAG DOUBLE (53)
STARS
ID INTEGER
RA DOUBLE (53)
DEC DOUBLE (53)
MAGNITUDE REAL (24)
SIGMA_MAG REAL (24)
NAME VARCHAR (255)
MIN_MAG REAL (24)
MAX_MAG REAL (24)
NO_MEASUREMENTS INTEGER
MAG_CAT REAL (24)
MAGSUM DOUBLE (53)
MAG2SUM DOUBLE (53)
SIGMA_RA REAL (24)
SIGMA_DEC REAL (24)
CAMID INTEGER
SSTAR_ID INTEGER
FIELD_ID INTEGER
SIGMA_FIELD REAL (24)
SIGMA_NIGHT REAL (24)
TMP_VALUE REAL (24)
AMP REAL (24)
PERIOD REAL (24)
HJD_T0 REAL (24)
CAM2_SSTAR_ID INTEGER
QUALITY SMALLINT
MAX_ID_FRM INTEGER
LAST_CHECKED INTEGER
NOVAEVENTS
NE_STAR INTEGER
NE_NIGHT INTEGER
NE_ID_FRM INTEGER
NE_RA DOUBLE (53)
NE_DEC DOUBLE (53)
NE_MAG DOUBLE (53)
NE_SEL_TYPE INTEGER
NE_NOVA_K2B CHAR (1)
NE_K2A_COUNT INTEGER
NE_K2B_COUNT INTEGER
NE_K2A_OBS INTEGER
NE_K2B_OBS INTEGER
NE_OTHER_STARS INTEGER
NE_K2B CHAR (1)
NE_FIELD_OBS INTEGER
NE_GOOD_EVENT INTEGER
NE_CHECKED INTEGER
NE_COMMENT VARCHAR (256)
NE_RUN INTEGER
NE_EVT_LINK VARCHAR (128)
NE_ID INTEGER
MEASUREMENTS
STAR INTEGER
TIME_HJD DOUBLE (53)
MAGNITUDE REAL (24)
ERROR REAL (24)
ID_FRM INTEGER
RA DOUBLE (53)
DEC DOUBLE (53)
MAG_PIPHOTO REAL (24)
CCDX REAL (24)
CCDY REAL (24)
MAG_AP0 REAL (24)
MAG_AP1 REAL (24)
MAG_AP2 REAL (24)
MAG_AP3 REAL (24)
MAG_AP4 REAL (24)
NEW_STAR CHAR (1)
QUALITY SMALLINT
MEASUREMENTS_RAW
STAR INTEGER
TIME_HJD REAL (24)
MAGNITUDE CHAR (1)
ERROR CHAR (1)
ID_FRM INTEGER
RA REAL (24)
DEC REAL (24)
MAG_PIPHOTO CHAR (1)
CCDX REAL (24)
CCDY REAL (24)
MAG_AP0 SMALLINT
MAG_AP1 SMALLINT
MAG_AP2 SMALLINT
MAG_AP3 SMALLINT
MAG_AP4 SMALLINT
NEW_STAR CHAR (1)
QUALITY SMALLINT
PIMANCOMMAND
PMC_NAME VARCHAR (64)
PMC_PARAM1 DOUBLE (53)
PMC_PARAM2 DOUBLE (53)
PMC_TIME INTEGER
PMC_NIGHT INTEGER
PMC_MODULE VARCHAR (16)
PMC_PAR1 VARCHAR (128)
PMC_PAR2 VARCHAR (128)
PMC_PAR3 VARCHAR (128)
PMC_PAR4 VARCHAR (128)
PMC_PAR5 VARCHAR (128)
PMC_PAR6 VARCHAR (128)
PMC_PAR7 VARCHAR (128)
PMC_PAR8 VARCHAR (128)
PMC_SITE VARCHAR (16)
PMC_PIMANID VARCHAR (16)
NIGHTSTAT
NS_NIGHT INTEGER
NS_CAMID INTEGER
NS_FRAME_COUNT INTEGER
NS_CAT_DONE INTEGER
NS_OPT_DONE INTEGER
NS_COMMENT VARCHAR (512
NS_QUALITY INTEGER
NS_MOON_RA DOUBLE (53)
NS_MOON_DEC DOUBLE (53)
NS_MOON_ILLUM DOUBLE (53)
NS_NUMGEN INTEGER
NS_NUMGENOK INTEGER
NS_NUMBKG INTEGER
NS_ONLINE_EFF DOUBLE (53)
NS_SYSSTARTED INTEGER
EVENT_PUBLIC
ID_EVENT INTEGER
EVENT_TYPE INTEGER
EVENT_DURATION VARCHAR (255)
EVENT_MAX_MAG VARCHAR (255)
EVENT_COMMENT VARCHAR (255)
DISP_COMMENT CHAR (1)
DISP_EVENT_EXT_URL CHAR (1)
DISP_K2A CHAR (1)
DISP_K2B CHAR (1)
DISP_AVK2A CHAR (1)
DISP_AVK2B CHAR (1)
DISP_SEVK2A CHAR (1)
DISP_SEVK2B CHAR (1)
LIGHTCURVE
LC_ID INTEGER
LC_ID_EVENT INTEGER
LC_NIGHT INTEGER
LC_FRAMENO INTEGER
LC_EVENTNO INTEGER
LC_CAMID VARCHAR (5)
LC_RUNTYPE INTEGER
LC_OBJECT_TYPE INTEGER
LC_HJD DOUBLE (53)
LC_MAG DOUBLE (53)
LC_TYPE INTEGER
LC_NAME VARCHAR (255)
LC_CURRFRAMENO INTEGER
TRACKS
TR_ID INTEGER
TR_NIGHT INTEGER
TR_A DOUBLE (53)
TR_B DOUBLE (53)
TR_FRAME_START INTEGER
TR_FRAME_END INTEGER
TR_VX DOUBLE (53)
TR_VY DOUBLE (53)
TR_TYPE INTEGER
TR_RADEC_A DOUBLE (53)
TR_RADEC_B DOUBLE (53)
TR_V_RA DOUBLE (53)
TR_V_DEC DOUBLE (53)
GRBLC
GLC_NAME VARCHAR (12)
GLC_GRB_ID INTEGER
GLC_TYPE INTEGER
GLC_TIME DOUBLE (53)
GLC_MAGNITUDE DOUBLE (53)
GLC_ERROR DOUBLE (53)
GLC_FILTER VARCHAR (16)
GLC_LIMIT INTEGER
GLC_RA DOUBLE (53)
GLC_DEC DOUBLE (53)
GLC_SAT VARCHAR (16)
GLC_TRONUM INTEGER
GLC_OBSTYPE INTEGER


FRAME
ID_FRM INTEGER
IFRAMENO INTEGER
IFRAMESECONDNO INTEGER
IDAYNIGHT INTEGER
SPATHTOFILE VARCHAR (1000)
ICAMID INTEGER
SCAMFILTER VARCHAR (10)
INAXIS1 INTEGER
INAXIS2 INTEGER
SOBJECT VARCHAR (20)
FROTATE DOUBLE (53)
FRA DOUBLE (53)
FHA DOUBLE (53)
FDEC DOUBLE (53)
FALT DOUBLE (53)
FZENITH_D DOUBLE (53)
FAZIM DOUBLE (53)
SDATE_OBS DATE
TTIME_UT INTEGER
SLOCTIME TIME
SLOCDATE DATE
FJD DOUBLE (53)
FHJD DOUBLE (53)
ISTARCOUNT INTEGER
IFITSSIZE INTEGER
BHAS_JPG CHAR (1)
POSANGLE DOUBLE (53)
AST_ORD INTEGER
ASTTIME INTEGER
AST_VER VARCHAR (16)
AST_ERR DOUBLE (53)
RA2000 DOUBLE (53)
DEC2000 DOUBLE (53)
PAR_X_0 DOUBLE (53)
PAR_X_1 DOUBLE (53)
PAR_X_2 DOUBLE (53)
PAR_X_3 DOUBLE (53)
PAR_X_4 DOUBLE (53)
PAR_X_5 DOUBLE (53)
PAR_X_6 DOUBLE (53)
PAR_X_7 DOUBLE (53)
PAR_X_8 DOUBLE (53)
PAR_X_9 DOUBLE (53)
PAR_X_10 DOUBLE (53)
PAR_X_11 DOUBLE (53)
PAR_X_12 DOUBLE (53)
PAR_X_13 DOUBLE (53)
PAR_X_14 DOUBLE (53)
PAR_Y_0 DOUBLE (53)
PAR_Y_1 DOUBLE (53)
PAR_Y_2 DOUBLE (53)
PAR_Y_3 DOUBLE (53)
PAR_Y_4 DOUBLE (53)
PAR_Y_5 DOUBLE (53)
PAR_Y_6 DOUBLE (53)
PAR_Y_7 DOUBLE (53)
PAR_Y_8 DOUBLE (53)
PAR_Y_9 DOUBLE (53)
PAR_Y_10 DOUBLE (53)
PAR_Y_11 DOUBLE (53)
PAR_Y_12 DOUBLE (53)
PAR_Y_13 DOUBLE (53)
PAR_Y_14 DOUBLE (53)
FLIP INTEGER
ERRCODE INTEGER
MATCHEDSTARS INTEGER
SHUTTER_MODE INTEGER
FWHM_AVER DOUBLE (53)
AVG DOUBLE (53)
RMS DOUBLE (53)
STAR_FRACTION DOUBLE (53)
CAT_STARS INTEGER
ASTROOK INTEGER
AVG_SHAPE DOUBLE (53)
PIXSCALE DOUBLE (53)
QUALITY SMALLINT
FRAME_DET
ID_FRM INTEGER
SOBSERVER VARCHAR (30)
SSOFTWARE VARCHAR (30)
SBUILD VARCHAR (50)
SDRVTYPE VARCHAR (30)
ICAMIIDX INTEGER
FEXPTIME DOUBLE (53)
FREXPTIME DOUBLE (53)
BSHUTTER CHAR (1)
FADCGAIN DOUBLE (53)
FADCBIAS DOUBLE (53)
FADCOSET DOUBLE (53)
FLNAGAIN DOUBLE (53)
FADCBSET DOUBLE (53)
FADCRANGE DOUBLE (53)
FADCCLAMP DOUBLE (53)
FELECGAIN DOUBLE (53)
BCOOLING CHAR (1)
FABINN DOUBLE (53)
FSBINN DOUBLE (53)
FSPEED DOUBLE (53)
SSPEEDMH VARCHAR (20)
BMPP_BC CHAR (1)
FRO_TIME DOUBLE (53)
FCROTIME DOUBLE (53)
IFOCUS INTEGER
BHITLENS CHAR (1)
SSAVEAREA VARCHAR (50)
SUSBMODE VARCHAR (20)
SFPGAVER VARCHAR (50)
SCPRSVER VARCHAR (50)
SVERDESC VARCHAR (50)
FRNOISE DOUBLE (53)
FRELNOISE DOUBLE (53)
FCHIPTSET DOUBLE (53)
FCHIPTEMP DOUBLE (53)
FCASTEMP DOUBLE (53)
FAMBTEMP DOUBLE (53)
FCAMHUMID DOUBLE (53)
FAMBHUMID DOUBLE (53)
FINTRTEMP DOUBLE (53)
FAIRMASS DOUBLE (53)
INTERESTINGOBJECTS
IO_ID INTEGER
IO_NAME VARCHAR (64)
IO_RA DOUBLE (53)
IO_DEC DOUBLE (53)
IO_STAR_K2A INTEGER
IO_STAR_K2B INTEGER
IO_TYPE INTEGER
IO_VMAG DOUBLE (53)
IO_COMMENT VARCHAR (128)
IO_QUALITY INTEGER
IO_RUN INTEGER
IO_BMAG DOUBLE (53)
IO_STAR_TYPE VARCHAR (8)
IO_NO_MEASUREMENTS INTEGER
IO_MATCH_DISTANCE DOUBLE (53)
IO_OTHERNAME VARCHAR (128)
IO_VAR INTEGER
IO_MAG DOUBLE (53)
IO_FILTER VARCHAR (16)
IO_FREQ INTEGER
IO_PERIOD INTEGER
IO_VISIBLE VARCHAR (256)
IO_ACTIVE INTEGER
IO_URL VARCHAR (256)
IO_LASTOBS INTEGER
IO_PRIOR INTEGER
IO_SSTAR_ID INTEGER
GRB
GRB_LOCAL_ID INTEGER
TRG_NUM INTEGER
NAME VARCHAR (12)
DATE DATE
TIME INTEGER
GRB_STATUS INTEGER
IS_HETE CHAR (1)
IS_INTEGRAL CHAR (1)
IS_IPN CHAR (1)
IS_GCN CHAR (1)
IS_KONUS CHAR (1)
IS_SWIFT CHAR (1)
Z VARCHAR (12)
HAS_OT CHAR (1)
FINAL_STATUS VARCHAR (64)
HAS_XA CHAR (1)
HAS_RA CHAR (1)
HAS_SN CHAR (1)
SGR_ID INTEGER
COORDRA DOUBLE (53)
COORDDEC DOUBLE (53)
COMMENT VARCHAR (1024)
REACTION_TIME DOUBLE (53)
OT_TIME DOUBLE (53)
OT_FLAG INTEGER
MIN_MAG DOUBLE (53)
FIRST_MAG DOUBLE (53)
FIRST_LIMIT_TIME DOUBLE (53)
MAG_FIRST_LIMIT DOUBLE (53)
GRB_SOURCE_ID INTEGER
GALEXT_U DOUBLE (53)
GALEXT_B DOUBLE (53)
GALEXT_V DOUBLE (53)
GALEXT_R DOUBLE (53)
GALEXT_I DOUBLE (53)
GALEXT_J DOUBLE (53)
GALEXT_H DOUBLE (53)
GALEXT_K DOUBLE (53)
GALEXT_LPRIM DOUBLE (53)
FLAREEVENTS
FL_ID INTEGER
FL_STAR INTEGER
FL_NIGHT INTEGER
FL_FLARE_MAG DOUBLE (53)
FL_MAG_LIMIT DOUBLE (53)
FL_GOOD_COUNT INTEGER
FL_PEAK_HEIGHT DOUBLE (53)
FL_QUALITY INTEGER
FL_CCDX DOUBLE (53)
FL_CCDY DOUBLE (53)
FL_MIN_TIME_HJD DOUBLE (53)
FL_MAX_TIME_HJD DOUBLE (53)
FL_CAMID INTEGER
FL_CAM2_SSTAR_ID INTEGER
FL_COMMENT VARCHAR (1024)
FL_RUN INTEGER
FL_HJD DOUBLE (53)
FL_START_HJD DOUBLE (53)
FL_END_HJD DOUBLE (53)
FL_XY_OK INTEGER
FL_MAG1 DOUBLE (53)
FL_MAG2 DOUBLE (53)
FL_CAM2_FL_ID INTEGER
FL_TYPE INTEGER
FL_NEAR_STAR_COUNT INTEGER
FL_MIN_MAG DOUBLE (53)
FL_MAX_MAG DOUBLE (53)
FL_FIELD_OBS_COUNT INTEGER
FL_REJ_REASON INTEGER
FL_ID_FRM INTEGER
FL_URL VARCHAR (1024)
FL_URL2 VARCHAR (1024)
FL_CAM_COUNT INTEGER
EVENT
ID_EVENT INTEGER
ID_FRM INTEGER
FRAME_NO INTEGER
EVENT_NO INTEGER
X INTEGER
Y INTEGER
LAP_NEW DOUBLE (53)
LAP_PREV DOUBLE (53)
LAP_MAX_NEW DOUBLE (53)
S_RAW DOUBLE (53)
SPHERICITY DOUBLE (53)
SIGINIFICANCE DOUBLE (53)
CLUSTER_SIZE DOUBLE (53)
COIC_RADIUS_PIX DOUBLE (53)
AVER_IN_PIXEL DOUBLE (53)
BLACK_RATIO DOUBLE (53)
EVENT_TYPE VARCHAR (100)
COIC_RADIUS_SEC DOUBLE (53)
RA DOUBLE (53)
DEC DOUBLE (53)
EVT_TIME INTEGER
EVTPATH VARCHAR (1024)
EVTLIST VARCHAR (64)
MAG DOUBLE (53)
EVTID VARCHAR (12)
EVT_SLT_OK CHAR (1)
EVT_EXTERNAL VARCHAR (512)
EVT_NIGHT INTEGER
EVT_ONLINE CHAR (1)
EVT_RUNTYPE INTEGER
LAP_NEXT DOUBLE (53)
EVT_CAMID VARCHAR (5)
EVT_GLON DOUBLE (53)
EVT_GLAT DOUBLE (53)
EVT_TYPE INTEGER
EVT_SLT_REASON VARCHAR (128)
EVT_MIN_DIST_CONE DOUBLE (53)
ALERT
ALERT_ID INTEGER
SOURCE_ID INTEGER
ALERT_TYPE INTEGER
TRG_NUM INTEGER
SEQ_NUM INTEGER
INSTRUMENT_ID INTEGER
ALERT_STATUS INTEGER
DATETJD INTEGER
DATE DATE
TIME DOUBLE (53)
RAYS VARCHAR (32)
BAND VARCHAR (32)
FLAX INTEGER
DURATION INTEGER
GAMMA_RATE INTEGER
COORDRA DOUBLE (53)
COORDDEC DOUBLE (53)
COORD1 DOUBLE (53)
COORD2 DOUBLE (53)
FNAME VARCHAR (100)
GRB_ID INTEGER
UNIX_TIME DOUBLE (53)
SIGMA DOUBLE (53)
VALIDITY INTEGER
COORDERR DOUBLE (53)
GAL_LONG DOUBLE (53)
GAL_LAT DOUBLE (53)
ECL_LONG DOUBLE (53)
ECL_LAT DOUBLE (53)
FLUX1 DOUBLE (53)
FLUX2 DOUBLE (53)
FLUX3 DOUBLE (53)
FLUX4 DOUBLE (53)
FLUX DOUBLE (53)
COMMENT VARCHAR (512)
ADDITIONAL_INFO VARCHAR (4096)


TARGETTOOBS
TO_NAME VARCHAR (128)
TO_RA DOUBLE (53)
TO_DEC DOUBLE (53)
TO_OBSHOUR INTEGER
TO_OBSSITE VARCHAR (16)
TO_OBSTIME INTEGER
PARAMCONFIG
PARAM_NAME VARCHAR (128)
PARAM_TYPE INTEGER
PARAM_VAL_INT INTEGER
PARAM_VAL_CHAR VARCHAR (128)
PARAM_VAL_DOUBLE DOUBLE (53)
UPSSTATUS
UPS_ID INTEGER
UPS_UNIXTIME INTEGER
UPS_DATE INTEGER
UPS_DTM VARCHAR (64)
UPS_VOLTAGE DOUBLE (53)
EVENTGEN
EG_NIGHT INTEGER
EG_FRAME INTEGER
EG_X DOUBLE (53)
EG_Y DOUBLE (53)
EG_GENIDENT INTEGER
WWW_USERS
LOGIN VARCHAR (30)
PASS VARCHAR (30)
LNAME VARCHAR (20)
FNAME VARCHAR (20)
IS_ADMIN CHAR (1)
HOTPIXEL
HP_ID INTEGER
HP_X DOUBLE (53)
HP_Y DOUBLE (53)
HP_CAM INTEGER
HP_RADIUS INTEGER
DB_HISTORY
DH_NAME VARCHAR (64)
DH_HOST VARCHAR (64)
DH_START_NIGHT INTEGER
DH_END_NIGHT INTEGER
MOUNTSPEED
MS_ID INTEGER
MS_TIME INTEGER
MS_OMEGA_HA DOUBLE (53)
MS_OMEGA_DEC DOUBLE (53)
PROCESSEDSTARS
PS_STAR_ID INTEGER
PS_NIGHT INTEGER
PS_TYPE INTEGER
ALERT_NOTICE_TYPE_BACKUP
TYPE_ID INTEGER
DESCR VARCHAR (255)
ADDITIONAL_INFO VARCHAR (255)
ALERT_NOTICE_TYPE
TYPE_ID INTEGER
DESCR VARCHAR (255)
ADDITIONAL_INFO VARCHAR (255)
BADFRAME
BF_ID INTEGER
BF_IDFRM INTEGER
BF_COMMENT VARCHAR (256)
DBLOCKTABLE
ID INTEGER
PID INTEGER
TEMPIDTABLE
ID INTEGER
GRB_SRC_CONFIRM
TYPE_ID INTEGER
DESCR VARCHAR (255)
ALERT_VALIDITY
VAL_VALUE INTEGER
VAL_DESCRIPTION VARCHAR (128)
FRAME_AVERAGED
ID_FRM INTEGER
SPATHTOFILE VARCHAR (1000)
ALERT_STATUS
TYPE_ID INTEGER
DESCR VARCHAR (256)
COINCDEF
CD_CAMID1 INTEGER
CD_CAMID2 INTEGER
EVENT_TYPE
EVENT_TYPE_VAL INTEGER
EVENT_TYPE_DESC VARCHAR (128)
ERRORDEF
ERRCODE INTEGER
ERRDESC VARCHAR (1024)
ASTROSOURCETYPE
AST_TYPE INTEGER
AST_DESC VARCHAR (256)
STARTYPE
ST_ID INTEGER
ST_DESC VARCHAR (1024)
MAX_ID_TABLE
MI_TABLE VARCHAR (32)
MI_MAX_ID INTEGER
RUNDATE
ID_RUN INTEGER
DATE VARCHAR (10)
FINAL_STATUS
ID_STAT INTEGER
DESCR VARCHAR (255)

Dodatek B

Oprogramowanie stworzone w ramach pracy

Do pracy dołączono płytę CD, która zawiera:

- w katalogu "migracja" – stworzone w ramach pracy narzędzia służące do tworzenia struktury bazy i migracji
- w katalogu "praca" – wersję elektroniczną tego dokumentu
- w katalogu "strony" – stworzone w ramach pracy strony WWW zawierające instrukcję obsługi opracowanych narzędzi.

Bibliografia

[IBM02] International Business Machines Corp., *IBM DB2 Turns 25!*, http://www-01.ibm.com/software/data/db2/25th-birthday/?wm=7115001f1779&cm_sp=ZZ999-_-SWB00-_-1779

[HA01] Linux High Availability Project, /textitHomePage: Linux HA, http://www.linux-ha.org/HomePage

[PI01] M. Biskup, L. Mankiewicz, M. Sokolowski, G. Wrochna, *Databases for the "Pi of the Sky" experiment*, http://grb.fuw.edu.pl/pi/papers/wilga06/MarekBiskupWilga06.pdf

[PI02] M. Biskup, M. Cwiok, H. Czyrkowski, R. Dabrowski, M. Denis, W. Dominik, J. Juchniewicz, G. Kasprowicz, A. Majcher, A. Majczyna, K. Malek, L. Mankiewicz, K. Nawrocki, R. Pietrzak, L. W. Piotrowski, M. Sokolowski, J. Uzycki, R. Wawrzaszek, G. Wrochna, M. Zaremba, A.F. Zarnecki, *"Pi of the Sky" – an innovative approach to astrophysical optical transients detection*, http://grb.fuw.edu.pl/pi/papers/2007/supernova_020.pdf

[PI03] M. Biskup, *Bazy danych "Pi of the Sky"*, http://www.mimuw.edu.pl/∼mbiskup/presentations/BazyDanychPi.pdf

[MC01] M. Chapple, *Data Definition Language*, http://databases.about.com/od/sql/a/sqlfundamentals_2.htm

[PSQL01] PostgreSQL Global Development Group, *PostgreSQL: About*, http://www.postgresql.org/about/

[PSQL02] PostgreSQL Global Development Group, *PostgreSQL: Documentation: Manuals: PostgreSQL 8.1: Tablespaces*, http://www.postgresql.org/docs/8.1/static/manage-ag-tablespaces.html

[Cho06] Raul F. Chong, Clara Liu, Sylvia F. Qi, Dwaine R. Snow, *Zrozumieć DB2 Nauka na przykładach Ilustrowany przewodnik po IBM DB2 + CD*, Wydawnictwo Naukowe PWN, Warszawa, 2006 r.

[Fed01] Red Hat, Inc., *Fedora Project*, http://fedoraproject.org

[JV01] Sun Microsystems Inc., *Java SE Technologies - Database*, http://java.sun.com/javase/technologies/database/

[IBM01] Vikram S. Khatri, Nora Sokolof, Manas Dadarkar, *Migrate from MySQL or PostgreSQL to DB2 Express-C*, http://www.ibm.com/developerworks/db2/library/techarticle/dm-0606khatri/

[Red01] Whei-Jen Chen, Rav Ahuja, Yong Jun Bi, Robert Borovsky, Patrik Fürer, Craig Maddux, Ichiro Ohta, Mark Talens, *DB2 Integrated Cluster Environment Deployment Guide*, IBM Redbooks, 2004 r.,
http://www.redbooks.ibm.com/redbooks/pdfs/sg246376.pdf