Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра математической кибернетики
и компьютерных наук

ПЕРЕНОС МЕТАДАННЫХ МЕЖДУ БАЗАМИ ДАННЫХ

АВТОРЕФЕРАТ ДИПЛОМНОЙ РАБОТЫ

Студентки 6 курса 611группы
Специальности 010501 – Прикладная математичка и информатика
факультета КНиИТ
Романова Александра Георгиевича

Научный руководитель
доцент, к.ф.-м.н. _____ И. А. Батраева

Заведующий кафедрой
к. ф.-м. н. _____ С. В. Миронов

Саратов 2016

ВВЕДЕНИЕ

Акутальность темы. Любая современная система управления базами данных имеет ресурсы для описания, вввода и хранения не только данных но и их описаний, т.е. структуры данных или метаданных. Иногда при пргоектировании приложений работающих с базами данных складывается ситации появления многоверсионности схемы одной и той же базы данных на различных устройствах. Эта проблема може возникать лбо при неаккуратной работе одного разработчика, либо при частой их смене и плохом документировании проекта. Поэтому иногда возникает необходимость сравнить внутреннюю стурктуру нескольких баз данных, определить их сходства и различия, и принять решение о синхронизации.

Цель. Целью дипломной работы является исследование внутренней структуры СУБД, схемы организации и взаимодействия системных объектов, поиск путей синхронизации метаданных. **Задачи.** Разработка метода сравнения и приведения внутренней структуры нескольких баз данных к одному состоянию. Разработка программы позволяющей выполнить эту процедуру.

Общая характеристика материала исследования. Системные таблицы в большинстве СУБД имеют такую же структуру, как и определенные пользователем таблицы и расположены в той же самой базе. Однако в силу различности версий СУБД, возможности разными способами описывать одни и те же смысловые конструкции языка могут возникать разночтения в коде базы данных. Необходимо определять эти ситуации.

Структура ВКР. Выпускная квалификационная работа имеет 4 главы.

Глава 1. Основы реляционной модели и ее использование в СУБД Firebird.

Глава 2. Внутренняя структура СУБД Firebird

Глава 3. Алгоритм синхронизации метаданных для СУБД Firebird

Глава 4. Разработка приложения для синхронизации метаданных.

1.Основное содержание работы

Базы данных называются *реляционными*, если управление ими основано на математической модели, использующей методы реляционной алгебры и реляционного исчисления. Чтобы считаться реляционной система управления базами данных должна, в частности [2] : представлять всю информацию в виде таблиц. поддерживать логическую структуру данных, независимо от их физического представления, использовать язык высокого уровня для структурирования, выполнения запросов и изменения информации в базах данных, поддерживать основные реляционные операции (выбор, проектирование и объединение), а также теоретико-множественные операции, такие как объединение, пересечение и дополнение, поддерживать временные таблицы, обеспечивая пользователям альтернативный способ просмотра данных в таблицах, различать в таблицах неизвестные значения (nulls), нулевые значения и данные, обеспечивать механизмы для поддержки целостности, авторизации, транзакций и восстановления данных.

СУБД Firebird является реляционной базой данных. Все данные в Firebird хранятся в виде таблиц. База данных состоит из различных объектов, таких как таблицы, виды, домены, хранимые процедуры, триггеры. Объекты базы данных содержат всю информацию о ее структуре. Объекты базы данных называются *метаданными*.Для исследования необходима информация о следующих внутренних объектах СУБД: Таблицы (Tables), Столбцы (Columns), Типы данных (Data types), Домены (Domains), Справочные ограничения целостности (Referential integrity constraints), Индексы (Indexes), Представления (Views), Хранимые процедуры (Stored procedures) , Триггеры (Triggers) , Генераторы (Generators) .

Задача синхронизации метаданных состоит в следующем. Существует база-*источник* (Master) и *синхронизируемая* база (Target). Необходимо привести Target к состоянию Master. То есть, все объекты - таблицы, ограничения индексы, домены, генераторы, процедуры и триггеры, должны

иметь одну структуру и одно описание на языке БД. Под структурой объекта понимается его строение и связь с другими объектами. Для того чтобы выяснить структуру объекта Firebird, необходимо обратиться к системным таблицам (system tables). Системные таблицы Firebird содержат метаданные базы данных. Они создаются автоматически сервером Firebird, когда создается сама база данных. Информация, содержащаяся в этих таблицах, определяет типы полей таблиц, их названия, связи между таблицами и пр. Эти таблицы сопровождаются сервером. Каждый раз, когда пользователь создает или изменяет объект базы, Firebird автоматически обновляет и добавляет записи в соответствующие системные таблицы. При извлечении метаданных могут возникать неоднозначности, связанные с особенностями структуры СУБД и способами задания схемы базы. Например, можно задать домен явно, а можно перечислить ряд свойств поля таких как тип, длину, набор символов и другие. В этом случае создается неявный домен, который не является системным объектом, но в таблице имеет название начинающееся с «RDB$». Хранимые процедуры не поддерживают домены, но используют их спецификацию и их таблицу (rdb$fields) для своих параметров (длина поля, тип и набор символов). Так как поля таблиц и процедур могут отменять спецификацию домена, происходят такие неоднозначности. Следующая проблема – системные таблицы не передают всю информацию необходимую для получения полного скрипта создания, принимающего во внимание все отношения в рабочей базе данных. Существует возможность удалить объекты, которые используются другими объектами базы данных. И сообщение об ошибке может быть получено только при обращении к уже удаленным объектам, используя ссылку из существующих (проблема «зависших ссылок»). Информация о зависимости объектов скрыта не только в нескольких системных таблицах (rdb$dependencies, rdb$ref_constraints) но и в BLR(язык двоичного представления) процедур и триггеров. Также Firebird не поддерживает тип DECIMAL в соответствии со стандартом. Типы DECIMAL и NUMERIC

практически идентичны, и оба могут являться машинозависимыми. До версии ODS9 (IB5.X), Firebird не определял различия при обозначении типа поля NUMERIC или DECIMAL, таким образом, после извлечения в скрипт, он конвертировался в SMALLINT, INTEGER или DOUBLE PRECISION, или же в NUMERIC максимальной точности, что зачастую было намного больше, чем ранее объявлено. В тоже время база, созданная под IB6 (ODS10) может использовать новое поле системной таблицы, rdb$precision, для сохранения информации о точности поля типа NUMERIC и DECIMAL. Информация же о типе этих полей доступна только из документации, а не из rdb$types. Следующую проблему представляют так называемые COMPUTED поля – особый тип. Значения такого поля подсчитываются непосредственно во время заполнения данными других полей и объектов. Обычно это колонки в пределах соответствующей строки таблицы, но возможно ссылки на другую таблицу, представление или даже процедуру. Такие ссылки не отслеживаются Firebird и, например, при успешном удалении или изменении такой процедуры, ошибка проявится позже. Существует 32 системные таблицы. Все они начинаются с индекса «RDB$» [4,5].

Чтобы получить информацию об объекте необходимо обратиться к соответствующей таблице, или нескольким. Например, для синхронизации таблиц необходимо получить списки полей каждой синхронизируемой таблицы, а также атрибуты – тип поля, используемый домен, ограничения на NULL значение. Для того, что бы получить список только таблиц можно использовать SQL запрос:

```
select rdb$relation_name from rdb$relations
where (rdb$system_flag = 0) and (rdb$view_source is null)
order by rdb$relation_name asc;
```

Запрос, возвращающий структуру конкретной таблицы EMPLOYEE

```
select  a.rdb$field_name, a.rdb$field_position,
a.rdb$null_flag, a.rdb$field_source,
a.rdb$default_source,
b.rdb$computed_source,b.rdb$field_length,
```

```
b.rdb$field_scale,    b.rdb$field_type,
b.rdb$field_sub_type   from  rdb$relation_fields a,
rdb$fields b   where a.rdb$field_source = b.rdb$field_name
and A.rdb$relation_name = "EMPLOYEE"  order by
a.rdb$field_position asc;
```

Вся информация о доменых типах содержится в таблице rdb$fields.

```
select rdb$field_name from rdb$fields
where rdb$system_flag = 0
order by rdb$field_name asc;
```

Этот SQL запрос выбирает все псевдонимы типов, которые были созданы для пользовательских таблиц (where rdb$system_flag = 0).

```
select * from rdb$fields where rdb$field_name = "BUDGET";
```

Этот SQL запрос выдает полную информацию о домене с именем BUDGET.

Важное значение для сохрания правильной схемы базы при синхронизации имеют различные ограничения, которые можно узнать для каждой таблицы в базе запросом примерно следующего вида:

```
select RDB$CONSTRAINT_NAME, RDB$CONSTRAINT_TYPE,
RDB$INDEX_NAME from rdb$relation_constraints where
rdb$relation_name = "EMPLOYEE";
```

Необходимо помнить о том, что часть ограничений реализуется и проверятся помошью триггеров, поэтому при выяснении структуры метаданных необходимо задать запрос о задействованных триггерах.

```
select rdb$trigger_name  from rdb$check_constraints  where
rdb$constraint_name = "INTEG_30";
```

В условии отбора указывается имя ограничения, полученное предыдущим SQL запросом. Для ограничений этого типа создается два триггера типа BEFORE INSERT и BEFORE UPDATE.

Индексы FOREIGN KEY поддерживают ссылочную целостность между таблицами в базе данных. При помощи их устанавливается связь, и им соответствуют индексы PRIMARY KEY. Среди служебных таблиц базы данных есть таблица, которая связывает пары FOREIGN KEY и PRIMARY KEY. Это RDB$INDICES. Учитывая все это можно определить, с какой

таблицей связана данная таблица по выбранному индексу FOREIGN KEY. Следующий SQL возвращает имя таблицы и индекс PRIMARY KEY, с которым осуществляет связь индекс FOREIGN KEY. В данном случае это RDB$FOREIGN9.

```
select rdb$relation_name , rdb$index_name from rdb$indices
where rdb$index_name in (select rdb$foreign_key from
rdb$indices where rdb$index_name = "RDB$FOREIGN9");
```

По такому же принципу строятся и остальные запросы для выяснения внутреней структуры метаданных схемы базы данных. Для проведения синхронизации данных между двумя базами, необходимо извлечь в скрипт все объекты, как одной, так и другой базы. Для этого нужно воспользоваться информацией из системных таблиц.

Для синхронизации такх объектов как таблицы и домены, необходимо сравнение каждого элемента объекта исходной базы с каждым синхронизируемой. Это делается напрямую, используя соответствующие значения из системных таблиц. Прямое обращение к системным таблицам дает неоспоримое преимущество перед стандартными конструкциями языка DDL. Так стандартной конструкцией невозможно изменить тип поля или домен, не удалив перед этим все поле целиком. Манипуляции с системными таблицами позволяют это сделать без удаления. Например:

```
Alter Table JOB ADD IBCTEMP NUMERIC(9,2);
COMMIT WORK;
Update Rdb$Relation_Fields Set Rdb$Field_Source=
(Select R.Rdb$Field_Source from Rdb$Relation_Fields
where R.Rdb$Relation_Name = 'JOB' and R.Rdb$Field_Name =
'IBCTEMP') WHERE RDB$FIELD_NAME='MIN_SALARY' AND
RDB$RELATION_NAME = 'JOB';
COMMIT WORK;
Alter Table JOB DROP IBCTEMP;
```

Здесь используется следующий прием: для того чтобы изменить характеристики поля, не удаляя его, можно создать в таблице временное поле. Как было показано выше, в RDB$FIELDS создается неявный домен.

Ссылка на этот домен передается в соответствующую характеристику изменяемого поля. Временное поле удаляется, но неявный домен остается, так как на него уже имеется ссылка. Для изменения остальных характеристик можно просто напрямую обращаться к соответствующим полям системной таблицы. Например, изменение значения NULL на NOT NULL:

```
update RDB$RELATION_FIELDS set RDB$NULL_FLAG = 1 where
(RDB$FIELD_NAME = 'MFORKC') and (RDB$RELATION_NAME =
'ORGANIZATION')
```

Остальные виды объектов (триггеры, процедуры, генераторы и т.п.) можно сравнивать целиком. Для этого для каждого из объектов генерируется несколько видов скриптов: удаление объекта, модификация на пустой, генерация, создание с пустым телом.

Скрипт удаления состоит из конструкции SQL стандартной для каждого объекта. Модификация на пустой – обновление процедуры или триггера, когда тело заменяется фиктивным и не выполняющим никаких полезных действий.

Пример для процедуры:

```
CREATE PROCEDURE ALL_LANGS RETURNS ( CODE VARCHAR(5)
CHARACTER SET NONE,   GRADE VARCHAR(5) CHARACTER SET NONE,
COUNTRY VARCHAR(15) CHARACTER SET NONE,     LANG VARCHAR(15)
CHARACTER SET NONE) AS BEGIN       EXIT; END
```

Пример для триггера:

```
ALTER TRIGGER SET_CUST_NO AS  Declare variable I integer;
BEGIN  I = 0;  END
```

Сравнение происходит по скрипту генерации, так как он содержит всю информацию об объекте. Если сравнение прошло удачно ограничение, индекс или генератор сохраняются, в противном случае удаляются. Модификация объектов на пустые необходима для разрыва связей между ними. После удаления объектов они создаются уже с измененной структурой. Однако Firebird запрещает производить операцию удаления, если объект используется другими объектами. Для того чтобы удалить или изменить

элемент, необходимо убрать ссылки на него от остальных объектов. При этом, после изменения, эти связи необходимо восстановить, причем выдерживая определенную последовательность.

Алгоритм был реализован с использованием среды разработки Lazarus. Каждому типу объекта была сопоставлена собственная структура. После заполнения структура добавлялась в список объектов своего типа. Структура заполняется один раз при извлечении метаданных из базы источника и синхронизируемой базы. Для каждой таблице исходной базы данных ищется соответствующая ей по имени таблица синхронизируемой. При нахождении сравниваются поля таблиц. Дальше специальная функция обрабатывает два соответствующих по названию поля и генерирует для них скрипт несоответствия. Если поля таблицы исходной базы отсутствуют в таблице синхронизируемой, поля добавляются. Если наоборот – удаляются из синхронизируемой. Аналогичный метод применяется для сравнения доменов.

При сравнении процедур, так же как и таблиц, устанавливается соответствие между именами процедур исходной и синхронизируемой базы. Так как алгоритм генерации скрипта один, то процедуры идентичны, если их скрипты обновления посимвольно равны. Такое сравнение верно для всех остальных объектов.

Полный скрипт несоответствия получается после обработки всех объектов базы. Самое важное условие выполнения процедуры синхронизации - базы должны находиться в целостном состоянии. Обязательным условием работы программы является открытость исходных кодов процедур, триггеров, представлений. Так как в работе рассматривается синхронизация только метаданных, то возможна ситуация когда изменение структуры противоречит данным, которые находятся в файле БД. Например, невозможно наложение первичных и уникальных ключей, когда данные столбца таблицы дублируются. Даже успешно синхронизированная база может оказаться неработоспособной.

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломной работы была изучена внутренняя структура объектов СУБД Firebird и отличительные особенности различных версий. Исследованы структура метаданных, строение системных таблиц Firebird и способам обращения, манипуляции и извлечения необходимой информации. Применены навыки использования языка SQL и создания сложных многоуровневых запросов к таблицам БД. Для синхронизации метаданных был реализован алгоритм и протестирована программа его реализующая.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Дейт К.Дж.* Введение в системы баз данных/К.Дж.Дейт.- М.:Вильямс, 2006. -1328 с., ил.
2. *Codd E.F.* Relation Model of Data for Large Shared Data Banks //Comm. ACM. - 1970. - V.13, №.6. - P.377-383.
3. *Бондарь А.Г.* InterBase и FireBird. Практическое руководство для умных пользователей и начинающих разработчиков./ А.Г.Бондарь. - Спб.- БХВ - Петербург. 2012. —-592с. ил.
4. *Кузьменко Д.* Полезные запросы к системным таблицам Interbase и Firebird[Электронный ресурс]/URL: http://www.ibase.ru//sysqry.htm (Дата обращения 10.05.2016). Загл. с экр. Яз.рус.
5. Смирнов А. О системных таблицах Interbase [Электронный ресурс]/URL: http://citforum.ru/database/interbase/new_systab/ (Дата обращения 11.05.2016). Загл. с экр. Яз.рус.
6. Основная документация по Firebird[Электронный ресурс]/ URL: http://www.ibase.ru/develop/ (Дата обращения 12.05.2016). Загл. с экр. Яз.рус.