Ходжаев Р.Д.

РАЗРАБОТКА МЕТОДОВ ПРЕДОТВРАЩЕНИЯ ОШИБОК ИНКРЕМЕНТАЛЬНОЙ ЗАГРУЗКИ В ХРАНИЛИЩЕ ДАННЫХ

R.D. Khodzhaev

DEVELOPMENT OF METHODS OF PREVENTING ERRORS IN INCREMENTAL LOADING OF DATA WAREHOUSES

УДК: 668.077/532.

В статье рассматривается разработка методов инкрементальной загрузки, предотвращающих возникновения ошибок в хранилище для трех видов операционных источников.

Ключевые слова: *банк, хранилища данных, инкрементальная загрузка, предотвращения ошибок в хранилище.*

The article considers the development of methods for the incremental load which prevent the occurrence of errors in a warehouse for three types of operating sources.

Key words: *bank, storage, incremental loading, prevent errors in storage.*

Существуют две основные причины, вызывающие ошибки при обновлении Хранилища Данных (ХД). Первая причина обусловлена постоянным изменением источника во время загрузки данных в ХД, из-за чего может иметь место задержка между изменением базовых данных и захватом этих изменений. Вторая причина возникновения ошибок заключается в том, что при проектировании ETL задач используются соединения таблиц, данные которых определены для разных моментов времени, то есть используются соединения таблиц, находящихся в разных состояниях. В данной статье предлагаются методы предотвращения ошибок обновления ХД безошибочно функционирующие, несмотря на вышеуказанные причины.

Используемый метод в значительной степени определяется свойствами операционных источников. В статье рассмотрены варианты для промежуточного, логируемого источников и источника с метками времени. При разработке методов необходимо учитывать, что банки Таджикистана заинтересованы в решениях, не блокирующих операционные источники.

Разработка метода инкрементальной загрузки для промежуточного источника.

Для промежуточных источников проблема предотвращения несоответствия измененных данных решается следующим образом. В каждом цикле загрузки система ETL запрашивает снимок текущего состояния источника, то есть полностью извлекаются исходные данные. При этом необходимо осуществлять полную блокировку операционного источника данных. Это обусловлено необходимостью обеспечения консистентности данных в ходе формирования снимков. Например, существуют две таблицы **A** и **B** на операционном источнике, где **A** ссылается на таблицу **B**. Предполагается, что таблица **B** содержит большой объем данных. В процессе создания снимков сначала формируется снимок **A**, затем снимок **B**. Формирование снимка **B** может занять значительное время, в течение которого в **A** могут произойти изменения, в результате чего получается неконсистентность снимков.

Снимки хранятся в рабочей зоне ETL инструмента, часто называемой как stage область. Снимки, сделанные во время предыдущего цикла загрузки, так же хранятся в stage области и система ETL может вычислить промежуточный дифференциал, сравнивая последующие снимки. Данный процесс изображен на рис.1.

Для инкрементальной загрузки система ETL не запрашивает непосредственно операционный источник. Запросы отправляются к снимкам в stage области. После создания снимки остаются неизменными. Поэтому ETL задачи не обнаружат непоследовательности изменения данных и соответственно не будет ошибок обновления ХД.

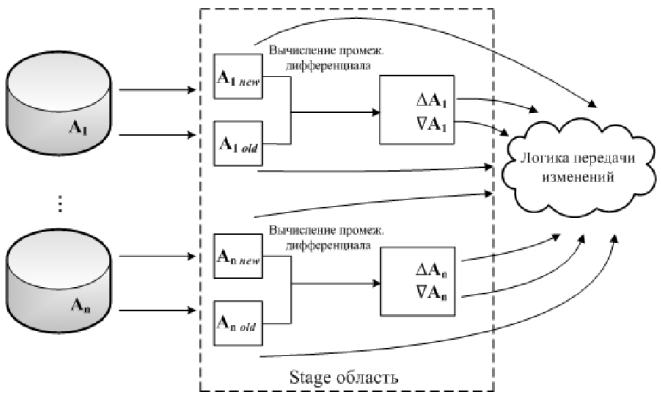

Рис. 1. Расчет промежуточных дифференциалов в stage области.

Во время инкрементальной загрузки базовые таблицы доступны в их текущем состоянии. Следовательно, необходимо разрабатывать ETL задачи таким образом, чтобы не требовался доступ к исходному состоянию. В этом случае снимки текущего и исходного состояния доступны в stage области и ETL задачи для инкрементальной загрузки разрабатываются на основе обоих состояний. Преимущество заключается в том, что необходимая логика передачи измененных данных проще, то есть ETL задача может быть выполнена с использованием меньшего количества операторов. Недостатком является то, что необходимо осуществлять полную блокировку операционного источника во время инкрементальной загрузки данных.

Проблема использования соединенных таблиц в ETL задачах, данные которых определены для разных моментов времени, не вызывает возникновения ошибок загрузки данных в ХД для промежуточных источников, так как вся необходимая информация хранится в снимках, которые находятся в stage области. Во время инкрементальной загрузки запросы выполняются только к снимкам в stage области и не затрагивают источник. Далее этот случай описывается в примере.

Транзакции вставки и удаления вычисляются следующим образом[1]:

$\Delta V = (\mathbf{A}_{curr} \triangleright\triangleleft \Delta\mathbf{B}) \cup (\Delta\mathbf{A} \triangleright\triangleleft \mathbf{B}_{curr}) \cup (\Delta\mathbf{A} \triangleright\triangleleft \Delta\mathbf{B})$ и $\nabla V = (\mathbf{A}_{curr} \triangleright\triangleleft \nabla\mathbf{B}) \cup (\nabla\mathbf{A} \triangleright\triangleleft \mathbf{B}_{curr}) \cup (\nabla\mathbf{A} \triangleright\triangleleft \nabla\mathbf{B})$, где $\mathbf{A}_{curr}$ и $\mathbf{B}_{curr}$ означают текущее состояние **A** и **B**; $\mathbf{A}_{new}$ и $\mathbf{B}_{new}$ - снимки, полученные во время очередной инкрементальной загрузки; $\mathbf{A}_{old}$ и $\mathbf{B}_{old}$ – снимки от предыдущей загрузки; $\Delta\mathbf{A}$, $\Delta\mathbf{B}$, $\nabla\mathbf{A}$, $\nabla\mathbf{B}$ - это результаты операций над старыми и новыми снимками:

$\Delta\mathbf{A}$: $\mathbf{A}_{new}$MINUS$\mathbf{A}_{old}$
$\nabla\mathbf{A}$: $\mathbf{A}_{old}$MINUS$\mathbf{A}_{new}$
$\Delta\mathbf{B}$: $\mathbf{B}_{new}$MINUS$\mathbf{B}_{old}$
$\nabla\mathbf{B}$: $\mathbf{B}_{old}$MINUS$\mathbf{B}_{new}$

Таким образом всю необходимую информацию для инкрементальной загрузки можно извлечь из stage области без обращения к источнику. Соответственно ошибки загрузки в данном случае не возникнут.

Вычисление промежуточных дифференциалов является простым и предотвращает ошибки при обновлении. Недостатками данного метода являются:

- требование больших вычислительных ресурсов для создания снимков;
- необходимость извлечения и отправления по сети значительных объемов данных.

Таким образом, метод промежуточных дифференциалов недостаточно соответствует коротким циклами загрузки, требуемым для обновления ХД, наполняемого в режиме, близком к реальному времени. Кроме того существует необходимость выполнения полной блокировки операционного источника данных во время инкрементальной загрузки.

Разработка метода инкрементальной загрузки для логируемого источника.

Логируемые источники поддерживают лог изменений, который может быть запрошен системой ETL. В результате система ETL может извлечь изменения, произошедшие после предыдущей загрузки.

В момент начала загрузки ETL система запрашивает в источнике измененные данные. В оставшееся время цикла загрузки никакие другие запросы не выполняются на стороне источника. Измененные данные используются ETL системой двумя способами (Рис. 2). Первый способ заключается в том, что данные выступают в качестве входной информации для ETL задач инкрементальной загрузки. Второй способ использует данные для поддержания локальной копии базовых таблиц.

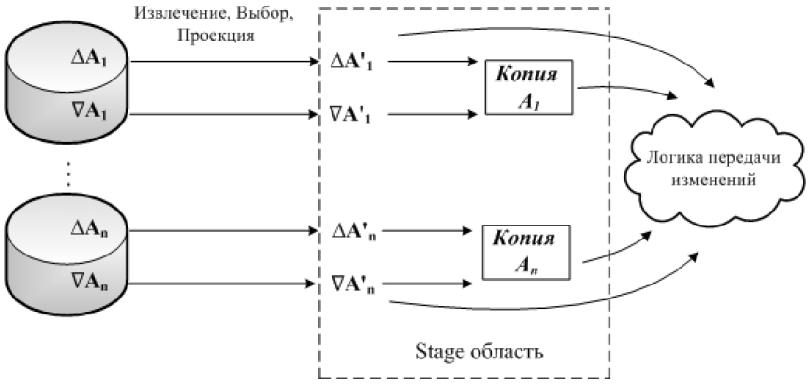

Рис. 2. Stage копии базовых таблиц

Это действие может выполняться непосредственно перед началом или же после окончания ETL задач. В первом случае ETL задачи используют копию исходного состояния базовых таблиц, во втором – копию текущего состояния базовых таблиц.

Для устранения проблемы задержки между изменением и захватом данных необходимо реализовать алгоритм, позволяющий зафиксировать момент времени t_0 (в дальнейшем точка отсчёта), после которого произведённые изменения на источнике не попадут в процесс загрузки данных. Начиная с этого момента времени t_0, передаваемые логи изменений источника не будут обрабатываться механизмом CDC (Change Data

Capture– Захват Изменения Данных), но будут накапливаться для обработки в последующих загрузках.

Пример

Имеются таблицы **A** и **B** на операционном источнике, где **A** ссылается на таблицу **B**. Предполагается, что данные **A** загружаются в таблицу ХД $\mathbf{A}_{stg}$, а **B** – в $\mathbf{B}_{stg}$. Процесс обновления данных следующий:

1. Обновляется таблица $\mathbf{A}_{stg}$:

- Считывание изменений из логов таблицы **A** (период выполнения операции *t1 –t2*)

- Обновление $\mathbf{A}_{stg}$ (период выполнения операции *t2 – t3*)

Предполагается, что в период времени *t2–t3* произошли изменения в таблицах **A** и **B**, а именно добавили новую запись в $\mathbf{A}(\mathbf{A}_{new})$ и новую запись в $\mathbf{B}(\mathbf{B}_{new})$, причём $\mathbf{B}_{new}$ ссылается на $\mathbf{A}_{new}$. Т.е. $\mathbf{A}_{new}$ не попала в процесс загрузки, так как её ввели в период времени *t2-t3*.

2. Обновляется таблица $\mathbf{B}_{stg}$:

В логах изменений таблицы **B** произошло изменение данных, в том числе и добавили запись $\mathbf{B}_{new}$, но так как $\mathbf{A}_{new}$ не попала в $\mathbf{A}_{stg}$, то данная запись ($\mathbf{B}_{new}$) приведёт к неконсистентности данных.

Далее данный пример рассматривается с учётом наличия точки отсчёта.

Перед процессом обновления устанавливается точка отсчёта t_0., после которой не обрабатываются изменения источника. Тогда получается:

1. Обновляется таблица$\mathbf{A}_{stg}$:

- Считывание изменений из логов таблицы **A** (период выполнения операции *t1 – t2*)

- Обновление $\mathbf{A}_{stg}$ (период выполнения операции *t2 – t3*)

Как и в первом случае предполагается, что в период времени *t2-t3* произошли изменения в таблицах **A** и **B**, а именно добавили новую запись в $\mathbf{A}(\mathbf{A}_{new})$ и новую запись в $\mathbf{B}(\mathbf{B}_{new})$, причём $\mathbf{B}_{new}$ ссылается на $\mathbf{A}_{new}$. Т.е. $\mathbf{A}_{new}$ не попала в процесс загрузки, так как её ввели не в период времени *t1-t2*, а в период *t2-t3*.

2. Обновляется таблица $\mathbf{B}_{stg}$:

В логах изменений таблицы **B** будут находиться новые измененные данные, но новая запись $\mathbf{B}_{new}$ не захватывается, так как время её изменения больше чем t_0. Тем самым исключается возможность появления неконсистентных данных.

Таким образом, для решения проблемы задержки захвата данных при инкрементальной загрузке необходимо использовать точку отсчета, то есть некоторый момент времени, после которого обработка лог-файлов приостанавливается до завершения инкрементальной загрузки.

Как было рассмотрено выше ошибки обновления возникают из-за несоответствия между состоянием базовых таблиц и данными в логах изменений.

Наиболее распространены следующие два метода устранения данного несоответствия:

1) блокировка операционного источника на время инкрементальной загрузки

2) создание копий базовых таблиц в stage области.

Недостаток блокируемых операционных источников очевиден: во время инкрементальной загрузки, все записывающиеся транзакции блокируются в источниках. Такой режим не может быть применен за исключением периодов, в которых отсутствуют значительные нагрузки.

Второй метод реализуется за счет потребления дополнительного объема памяти, при этом снижается воздействие на операционные источники.

Организация хранения копий базовых таблиц в stage области препятствует возникновению ошибок обновления, как для синхронных, так и для асинхронных режимов логирующих источников. В асинхронном режиме логирования может присутствовать некоторая задержка между изменением базовых таблиц и соответствующей записи в логе. Следовательно, изменения, не попавшие в лог на момент начала цикла загрузки, не рассматриваются для поддержания stage копии. То есть, состояние копии может отставать от состояния базовых таблиц. Тем не менее, копии всегда соответствуют извлеченному изменению данных. Таким образом исключается возникновение несоответствия изменения данных.

По сравнению с другими рассмотренными ранее методами stage копии базовых таблиц имеют несколько преимуществ:

- минимальное воздействие на операционные источники, то есть меньший объем данных извлекается из операционного источника в каждом цикле загрузки.

Недостатком этого подхода является дополнительный объем памяти необходимый для stage области.

Разработка метода инкрементальной загрузки для источника с метками времени.

В источниках с метками времени изменения захватываются запросом записей с более поздними метками времени, чем самая последняя метка времени, обнаруженная после последней загрузки. Удаления не могут быть обнаружены таким способом. Следовательно, в ХД могут быть переданы только добавления и обновления.

Для решения проблемы задержки между изменением и захватом данных механизм CDC должен определять новые изменения, которые произошли с момента начала последней инкрементальной загрузки. Для этого вводится так называемая точка отсчета, то есть метка времени t_0, которая представляет собой запись в журнале, указывающую на начало инкрементальной загрузки. При следующей загрузке механизм CDC будет начинать захват изменений, произошедших после ранее созданной точки отсчета времени, указанной в журнале.

Проблема несоответствия измененных данных может возникнуть когда системе ETL требуется отправить запрос в операционный источник во время инкрементальной загрузки. После этого система ETL может обнаружить изменения, внесенные в базовые таблицы и произошедшие после того, как были извлечены измененные данные, т.е. имеет место связывание таблиц,

находящихся в разных состояниях. В случае возможной блокировки источника с отметками времени, несоответствия измененных данных можно избежать, блокируя базовые таблицы при выполнении инкрементальной загрузки. Блокировки должны быть запрошены перед извлечением измененных данных и не должны освобождаться до тех пор, пока не будут обработаны все запросы в сторону соответствующей базовой таблицы.

Для уменьшения воздействия на операционную систему и предотвращения ошибок при обновлении предлагается использование stage копии базовых таблиц в логируемых источниках. Но при использовании данного метода имеются проблемы для источников с метками времени. Удаления не обнаруживаются, когда колонки аудита используются для захвата изменения. Следовательно, удаления не могут быть переданы в stage область и из-за этого объем памяти stage области будет постепенно расти. Кроме того не передаются удаления базовых таблиц целиком и они остаются в stage копиях и, следовательно, воздействуют на передачу изменений. Таким способом изменения, переданные в ХД, могут частично возникать из записей, уже не существующих в источнике.

Таким образом, при использовании stage копии источника с отметками времени необходимо учитывать следующие факторы:

- stage копии со временем увеличиваются в размерах;
- передача изменения может быть отклонена искусственным образом.

Анализ преимуществ и недостатков разработанных методов позволит в дальнейшем решить проблему инкрементальной загрузки данных ХД в банках Республики Таджикистан.

Литература:

1. Шин В.В., Ходжаев Р.Д. Анализ способов обновления хранилища данных. Материалы VI-й Международной научно-практической конференции «Перспективы развития науки и образования». Часть-2. Издание Таджикского Технического Университета имена академика М.С. Осими, Душанбе, ноябрь 2012.

2. Kimball, R., Caserta, J.: The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data. John Wiley & Sons, 2004

3. Спирли Э. Корпоративные хранилища данных: планирование, разработка, реализация, Том 1, М.– СПб-Киев, Вильямс, 2001. – 396С.

4. Барсегян А.А., Куприянов М.С., Степаненко В.В., Холод И.И. Методы и модели анализа данных: OLAP и Data Mining. – СПб.: БХВ-Петербург, 2004. – 336 с.: ил. ISBN 5-94157-522-X.

Рецензент: к.т.н., доцент Глазунов Д.В.