Шин В.В., Ходжаев Р.Д.

ФОРМИРОВАНИЕ ETL ЗАДАЧ ДЛЯ ИНКРЕМЕНТАЛЬНОЙ ЗАГРУЗКИ ДАННЫХ

V.V. Shin, R.D. Khozhaev

FORMATION OF ETL TASKS FOR INCREMENTAL DATA LOAD

УДК:623:627.35/4

В статье рассматривается формирование ETL задач для инкрементальной загрузки информации из легируемых источников в хранилища данных банков.

The article describes the formation of ETL tasks for incremental load of information from logged sources to data warehouses of banks.

Типовые Автоматизированные Банковские Системы (АБС) ориентированы на транзакционную обработку данных. В связи с этим банки внедряют систему поддержки принятия решений (СППР) основанную на хранилище данных (ХД). Структура ХД отличается от структуры базы данных АБС (операционного источника). Данное отличие позволяет сотрудникам банка формировать отчеты из ХД для комплексного анализа данных, при этом, не загружая операционный источник. Для того чтобы в ХД были актуальные данные оно периодически обновляется, то есть данные загружаются из операционного источника в ХД. Для обновления данных ХД используются ETL задачи. ETL задача представляет собой совокупность правил, которые выполняют функцию сопоставления данных источника и ХД. ETL задачи применяются для валидации, очистки и дополнения загружаемых в ХД данных. Существуют две категории ETL задач, различающиеся по технике их формирования: полная и инкрементальная.

Для полной загрузки источниками данных являются операционный источник, внешние таблицы (текстовые файлы) и ETL область. Источниками данных для инкрементальной загрузки (ИЗ) являются представления stage области, внешние таблицы (текстовые файлы), снимки таблиц источника, таблицы ETL области. ETL область предназначена для установления связи между записями в таблицах ХД и операционного источника. Основная задача ИЗ - загрузка новых данных в соответствующие сущности ХД и снимки таблиц источника, размещенные в ETL области. Для формирования ETL задач предлагается использовать Oracle Warehouse Builder версии 11.2 (OWB). С помощью операторов OWB конфигурируются правила трансформации и загрузки данных в целевые таблицы. Далее рассматривается формирование типовых ETL задач и настройка целевых таблиц для ИЗ данных в ХД.

Редактор общих настроек ETL задачи можно вызвать в рабочей области «Дерево метаданных» OWB. В настройках необходимо установить следующие параметры (ключевые выделены жирным шрифтом) (Рис.1). Наиболее важными параметрами являются: Use Target Load Ordering, Generation Mode, Default Operating Mode. В зависимости от логики ETL задачи значения указанных параметров могут меняться, но, как правило, это:

Use Target Load Ordering - true - использование той очерёдности загрузки, которая указана в настройках ETL задачи;

Generation Mode = Row Based - генерация ETL задачи для построчного режима;

Default Operating Mode = Row Based - выполнение загрузки в построчном режиме.

Property	DEFAULT_CONFIGURATION
CRD_DEV_GRP_DIC_INCR	
Deployable	true
Generation Comments	
Language	PL/SQL
Referred Calendar	
⊞ Aggregation Operators	
⊞ Chunking options	
⊟ Code generation options	
Analyze table statements	true
ANSI SQL Syntax	true
AUTHID option	None
Bulk processing code	true
Commit Control	Automatic
Enable Parallel DML	false
Error Trigger	
Generation Mode	Row based
Optimized code	true
Use Target Load Ordering	true
⊟ Runtime parameters	
Analyze table sample percentage	40
Bulk size	1000
Commit frequency	1000
Default audit level	Error Details
Default Operating Mode	Row based
Default purge group	WB
Maximum number of errors	50

Рис. 1. Настройки ETL задачи

Рассматриваются следующие типовые инкрементальные загрузки:

- загрузка стандартной сущности;
- загрузка древовидной (иерархической) сущности;
- загрузка снимков.

Ниже описывается реализация каждой типовой ИЗ с использованием следующих обозначений:

A (Fsp, Fs) - таблица-источник, где Fsp - первичный ключ, Fs -поля, не входящие в первичный ключ.

Teti(Fdwhp) Fsp, Fa) - таблица ETL - области для полного набора связей ключей источника и ХД, где Fdwhp - первичный ключ таблицы ХД, Fsp – первичный ключ таблицы-источника, Fa - дополнительные поля.

$T_{et}i$(dmi) (Fdwhp, Fsp,F_t,F_a) - таблица ETL области, хранящая результат последней инкрементальной заг-

рузки, где F_{dwhp} – первичный ключ таблицы ХД, F_{sp} – первичный ключ таблицы-источника, F_t – технические поля, F_a – дополнительные поля.

T_{dwh} (F_{dwhp}, F_{dwh}) – таблица ХД, где F_{dwhp} – первичный ключ, F_{dwh} – поля, не входящие в первичный ключ, S_T_{dwh} – последовательность, реализующая суррогатный ключ F_{dwhp}.

V_A_s (F_s, F_t) – представление stage области, где: F_s – поля таблицы источника, F_t – технические поля, **fTrn** – функция для получения уникального номера транзакции.

Код создания функции **fTrn** следующий:

```
FUNCTION fTrn(
  CSCN_IN IN NUMBER,
  RSID_IN IN NUMBER) RETURN
  NUMBER AS
  BEGIN
  RETURN (CSCN_IN*10E10 + RSID_IN);
  END fTrn;
```

При инкрементальной загрузке стандартной сущности необходимо использовать в качестве источника V_A_{stg}. Так как у стандартной сущности первичный ключ не изменяется, то допускается консолидация DM операций. Суть консолидации операций следующая: необходимо определить последнюю DML операцию в V_A_{stg} относительно первичного ключа F_{sp}. Так как транзакцию определяет уникальным образом поле CSCN$ принято использовать его при определении консолидирующей операции:

```
SELECT V_Astg.Fsp,
MAX ( fTrn(V_Astg.CSCN$,V_Astg.RSID$)
FROM V_A
GROUP BY V_A.Fsp
```

Из стандартной сущности состоит из трех шагов.

Первый шаг – это консолидация операций. Результат консолидации обозначается V_A_{cons} – это не таблица, а набор данных, получаемый запросом из V_A.

Второй шаг – разделение потоков данных в зависимости от типа операции и загрузка соответствующих целевых таблиц:

первый поток – удалённые данные. Тип операции DELETE. Условие ветвления: **V_A.OPERATION$ = 'D'** (обозначим набор данных $V\_A_{cons(del)}$). В редакторе OWB с помощью операторов «Панели инструментов» необходимо реализовать следующие DML конструкции:

а) фиксируется удаление данных в $T_{etl(del)}$

```
INSERT INTO Tetl(del)
(Fdwhp, Fsp, CSCN$, RSID$, DML_TIMESTAMP)
SELECT Tetl.Fdwhp, V_Acons_del.Fsp, V_Acons_del.CSCN$,
V_Acons(del).RSID$, V_Acons(del).DML_TIMESTAMP
FROM V_Acons(del), Tetl
WHERE V_Acons(del).Fsp = Tetl.Fsp
```

б) удаляются записи из T_{dwh}:

```
DELETE FROM Tdwh
WHERE Fdwhp IN (SELECT Fdwhp FROM Tetl(del))
```

в) T_{etl} синхронизируется с таблицами источника с помощью stage области:

```
DELETE FROM Tetl
WHERE Fdwhp IN (SELECT Fdwhp FROM Tetl(del))
```

второй поток – добавленные и изменённые данные. Тип операции INSERT или UPDATE. Условие ветвления: **V_A.OPERATION$IN ('I','UN')**. Для определения типа изменяемой DML операции, необходимо объединить внешним соединением V_A_{cons} и T_{etl}:

```
SELECT Tetl.Fdwhp, V_Acons.*
FROM V_Acons, Tetl
WHERE V_Acons.OPERATION$ IN ('I', 'UN')
AND V_Acons.Fsp = Tetl.Fsp(+)
```

Записи, у которых $T_{etl}.F_{dwhp}$ не определён (IS NULL) – новые записи (набор данных обозначается $V\_A_{(cons(ins))}$), а у которых $T_{etl}.F_{dwhp}$ определён (IS NOT NULL) – изменённые (набор данных обозначается $V\_A_{cons(upd)}$). Таким образом, второй поток вновь разделяется на два потока: для новых и для изменённых записей и необходимо реализовать следующие DML конструкции:

а) изменение существующих данных:

a_1) фиксируется изменение данных в $T_{etl(upd)}$:

```
INSERT INTO Tetl(upd)
(Fdwhp, Fsp, CSCN$, RSID$, DML_TIMESTAMP)
SELECT Tetl.Fdwhp, V_Acons(upd).Fsp, V_Acons(upd).CSCN$,
V_Acons(upd).RSID$, V_Acons(upd).DML_TIMESTAMP
FROM V_Acons(upd), Tetl
WHERE V_Acons(upd).Fsp = Tetl.Fsp
```

a_2) изменяется данные в T_{dwh}:

```
UPDATE Tdwh T SET (T.Fdwh) =
(SELECT V_Acons(upd).Fs FROM V_Acons(upd), Tetl(upd)
WHERE V_Acons(upd).Fsp = Tetl(upd).Fsp
AND Tetl(upd).Fdwhp = T.Fdwhp)
```

б) добавление новых данных:

$б_1$) фиксируется добавление данных в $T_{etl(ins)}$:

```
INSERT INTO Tetl(ins)
(Fdwhp, Fsp, CSCN$, RSID$, DML_TIMESTAMP)
SELECT S_Tdwh.NEXTVAL, V_Acons(ins).Fsp, V_Acons(ins).
CSCN$, V_Acons(ins).RSID$, V_Acons(ins).DML_TIMESTAMP FROM
V_Acons(ins)
```

$б_2$) добавляется данные в T_{dwh}:

```
INSERT INTO Tdwh (Fdwhp, Fdwh)
SELECT Tetl(ins).Fdwhp, V_Acons(ins).Fs
FROM V_Acons(ins), Tetl(ins)
WHERE V_Acons(ins).Fsp = Tetl(ins).Fsp
```

$б_3$) T_{etl} синхронизируется с таблицами источника с помощью stage области:

```
INSERT INTO Tetl (Fdwhp, Fsp)
SELECT Fdwhp, Fsp
FROM Tetl(ins)
```

Третий шаг – определение очерёдности загрузки. Это очень важный шаг, так как ETL задача состоит из нескольких потоков загрузки и эти потоки взаимосвязаны. Очерёдность загрузки будет следующей:

1. фиксируется удаление данных в $T_{etl(del)}$
2. удаляются записи из T_{dwh}
3. удаляются записи из T_{etl}
4. фиксируется изменение данных в $T_{etl(upd)}$
5. изменяются данные в T_{dwh}
6. фиксируется добавление данных в $T_{etl(ins)}$
7. добавляются данные в T_{dwh}
8. добавляются данные в T_{etl}

Принципы ИЗ для древовидной сущности схожи с ИЗ для стандартной сущности. Отличия заключаются в следующем:

- при добавлении новых данных либо изменении существующих, поле T_{dWh}ID – HI игнорируется, т.е. не заполняется (см. загрузку стандартной сущности пункты a_2, $б_2$).

- реализуется отдельный дополнительный поток по загрузке T_{dwh}.ID – HI, основанный на следующей DML конструкции:

UPDATE T_{dwh} T_1 SET (T_1.ID_HI) = (SELECT T_2.ID_HI FROM

(SELECT T_2.Ftohp, TREE.Fdwhp as ID HI FROM Tai, Tai TREE WHERE TM,. SRC_ID_HI = TREE.Fsp (+) MINUS

SELECT Tduh.Fdwhp, Tdwh D_HI FROM Td,,h) T2 WHERE T2 Ftohp — Ti.Fdwhp)

очерёдность загрузки целевых таблиц повторяет очерёдность загрузки стандартной сущности. Поток по заполнению Tdwh.ID_HI должен выполняться последним.

Инкрементальная загрузка снимков состоит из трех шагов:

Первый шаг - консолидация операций **(V–Acons).**

Второй шаг - разделение потоков: первый поток - удалённые данные:

а) удаляются данные из Tet1

DELETE FROM T et1

WHERE F IN (SELECT V_Acons(det)Fsp

FROM V_Acons<del), Tet

WHERE V_Acons(del).Fsp = Tetl.Fsp

второй поток - добавленные и изменённые данные. Тип операции INSERT или UPDATE. Условие ветвления: V_A.OPERATIONS IN ('I', 'UN'). В данном случае не будет определяться тип операции, а устанавливается для TeH тип загрузки INSERT\ UPDATE, т.е. MERGE:

а) обновляются данные в Tetl: MERGE INTO Ta1 T USING (SELECT Fsp, Fs FROM V_Acons MERGEJ5UBQUERY ON (T.Fsp = MERGE_SUBQUERY.Fsp) WHEN NOT MATCHED THEN INSERT (Fsp,Fs) VALUES (MERGE_SUBQUERY. Fsp, MERGE_SUBQUERY.Fs)

WHEN MATCHED THEN

UPDATE SET T.Fs = MERGE_SUBQUERY.Fs Третий шаг - определение очерёдности загрузки данных в таблицы (не существенно, так как потоки не взаимосвязаны):

- удаляются записи из Te,,
- обновляются записи в Te,|

С помощью OWB Designer настраиваются режимы генерации и выполнения ETL задач. Существует построчный (Row Based) и пакетный (Set Based) режим генерации ETL задач. Генерацией называется формирование PLASQL пакета (package) на базе существующей ETL задачи с учётом установленных параметров.

При пакетном режиме достигается высокий уровень производительности, но при этом формируется минимум логирующей информации. Такой режим подходит для штатной работы загрузки данных.

Построчный режим, сопровождается низкой производительностью загрузки, при этом лог выполнения загрузки отражает более полную информацию о происходящих событиях во время работы ETL задачи. Данный режим применяется при отладке (Debug) процесса загрузки.

В пакетном режиме реализованные DML конструкции загружают данные одним блоком, например:

INSERT INTO...

SELECT... FROM

MERGE INTO ... USING (SELECT...

DELETEFROM ... WHERE

Тем самым достигается высокая производительность. При пакетном режиме недопустимо использование конструкций изображенной на рис. 2.

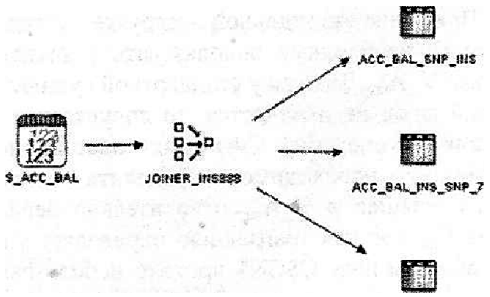

Рис.2

Так как при вставке записей в ACC_BAL_ SNP_ INS, ACC_BAL_INS_SNP_7, ACCJBALJNSJ1 значение S ACC BAL.NEXTVAL будет разным, хотя предполагается, что оно должно быть одинаковым.

Во втором режиме конструкции DML выполняют загрузку построчно. Пакет выполняется по следующей логике: набор данных заполняет коллекцию (FETCH <CURSOR> BULK COLLECT INTO), а затем для каждой записи коллекции выполнится DML операция в определённые целевые таблицы (FORALL i IN ... INSERT INTO), при этом логирующая информация более детальна.

Предлагаемый метод формирования ETL задач для инкрементальной загрузки позволит в дальнейшем разработать систему инкрементального обновления ХД банков из дотируемых источников.

Литература:

1. Фейерштейн С., Прибыл Б. Oracle PL\SQL для профессионалов 3-е изд. СПб.: Питер, 2003. - 941 е.: ил. ISBN 5-318-00528-4.
2. Griesemer B. Oracle Warehouse Builder llg:Getting Started. Packt Publishing Ltd, 2009.-368p. ISBN 1847195741
3. МакДональд К., Кац Х., Кристофер Б., Кальман Дж, Нокс Д. Oracle PL/SQL для профессионалов: практические решения. СПб.: ДиаСофтЮП, 2005. - 560 е.: ISBN 5-93772-160-8.

Рецензент: к.т.н., доцент Советбеков Б.С.