

ETL Processing within Oracle9i

An Oracle White Paper

June 2001

ETL Processing within Oracle9i

EXECUTIVE OVERVIEW	3
ORACLE9I - THE BUSINESS INTELLIGENCE PLATFORM	3
THE COMMON ETL CHALLENGE.....	4
HOW ORACLE CHANGES THE ETL PROCESS.....	5
ETL processing mostly outside the database	6
Loading into a database staging area for ETL processing.....	7
The new paradigm.....	8
ETL CAPABILITIES OF ORACLE9I.....	8
Change Data Capture.....	8
External Tables	9
Example:.....	10
Multi-Table Insert	11
Example:.....	11
Upsert Functionality	12
Example:.....	12
Table Functions.....	13
Example:.....	14
Transportable Tablespaces.....	15
Resumable Statements.....	16
Parallel DML Operations	16
Partitioning and Partition Maintenance Operations	16
CONCLUSION.....	17

EXECUTIVE OVERVIEW

Oracle8i is the leading relational database for data warehousing. Oracle has achieved this success by focusing on basic, core requirements for data warehousing: performance, scalability, and manageability. Data warehouses will store larger volumes of data, support more users, and require faster performance, so these key requirements remain key factors in the successful implementation of data warehouses.

However, Oracle9i goes far beyond the core requirements of performance, scalability, and manageability and represents the first true 'business intelligence platform'. Many data warehouses today use the relational database primarily for managing data and executing basic queries. While these operations are the foundation of any data warehouse, Oracle9i broadens the footprint of the relational database in a data warehouse so that Oracle9i is the scaleable data engine for all operations on data warehousing data, not just loading and basic query operations. Oracle9i provides new server functionality in three areas: analytic capabilities, ETL (Extraction, Transformation, Loading), and data-mining.

This paper provides an overview about the ETL capabilities of Oracle9i, and describes how this new database functionality redefines the way to implement ETL data flows and transformations, by utilizing Oracle9i as an ETL transformation engine. Other tasks that are also essential for a successful ETL implementation; such as scheduling, monitoring and maintenance addressed by the Oracle solution with Oracle Warehouse Builder, but are beyond the scope of this paper.

ORACLE9i - THE BUSINESS INTELLIGENCE PLATFORM

Oracle9i continues to lead the industry in providing enterprise-level performance, scalability and manageability, and provides many new features in these areas. However, Oracle9i also introduces broadens the capabilities of the relational database for data warehousing.

Oracle9i is the industry's first 'business intelligence platform'. The benefit of a data warehouse platform is integration, and specifically the integration of the

data so that the same server infrastructure can be leveraged for all tasks which involve processing of large amounts of data.

Oracle has extended the database's capabilities and the database's language (SQL) in order to be able to meet the requirements of a business intelligence platform. The relational features of Oracle9i do not imply that the relational database is a complete solution for OLAP, data-mining or ETL; however, these features do provide a complete server infrastructure for operating on large volumes of data in each of these scenarios.

THE COMMON ETL CHALLENGE

The methodology and tasks associated with ETL have been well known for many years, and are hardly unique to data warehouse environments: a wide variety of proprietary applications and database systems form the IT backbone of any enterprise. Data has to be shared between applications or systems in order to integrate them, so that these applications have (at least partially) the same picture of the world. This data sharing was often addressed by mechanisms similar to what we nowadays call ETL.

Data warehouse environments face the same challenge with the additional burden that they not only have to exchange but to integrate, rearrange and consolidate data over many systems, thereby providing a new unified information base for Business Intelligence. Additionally, the data volume in data warehouse environments tends to be very large.

What happens during the ETL process? During *Extraction*, the desired data has to be identified and extracted from many different sources, including database systems and applications. Very often, it is not possible to identify the specific subset of interest; therefore more data than necessary has to be extracted, since the identification of the relevant data will be done at a later point in time. Depending on the source system's capabilities (e.g., OS resources), some transformations may take place during this extraction process. The size of the extracted data varies from hundreds of kilobytes to hundreds of gigabytes, depending on the source system and the business situation. Just as the size of the data extraction may vary widely, the frequency at which the data is extracted may also vary widely: the time span may vary between days/hours and minutes to near real-time. Web server log files for example can easily become hundreds of megabytes in a very short period of time, thus necessitating frequent extractions.

After extracting data, this data has to be physically transported to the target system or an intermediate system for further processing. Depending on the chosen mechanism of transportation, some transformations can be done during this process too. For example, a SQL statement which directly accesses a remote target through a gateway can do transformations using standard SQL

processing (a simple example would be concatenating two columns as part of the SELECT).

After extracting (and transporting) the data, the most challenging and time consuming parts of ETL follow: *Transformation* and *Loading* into the target system. This may include

- Applying complex filters.
- Validating the incoming data against information which already existing in target database tables.
- Comparing new data to existing data in the data warehouse, to determine whether the new data needs to be inserted or updated.
- Computing aggregations and other derived data based on the new data.

These operations should be done as quickly as possible in a scaleable manner and must not affect the existing target with respect to concurrent access for information retrieval.

In multilevel data warehouse architectures, some of the above described process steps might take place several times between an ODS (operational data store) and a data warehouse/data mart or between a data warehouse and data marts, ETL can become a very complex process within a data warehouse architecture.

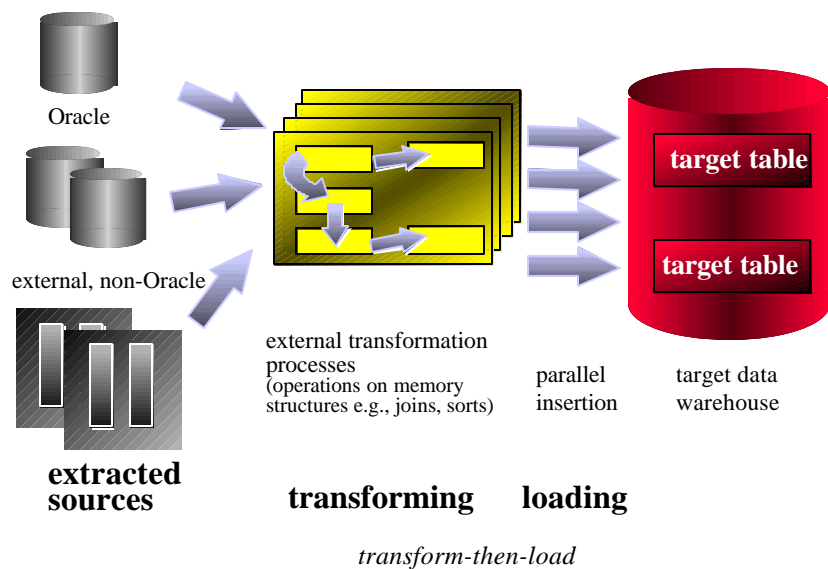
HOW ORACLE CHANGES THE ETL PROCESS

The common ETL process flow prior to Oracle9i could be classified into two main categories:

- ETL processing outside the database
- Loading into a database staging area for ETL processing

ETL processing mostly outside the database

Most of the transformation and cleansing is done outside the database, in separate standalone ETL engines/processes. These engines work with various data sources, trying to integrate them for the necessary ETL steps. If existing data in the target database is required, e.g., for data cleansing or ID lookup, the target database is treated like every other external data source involved in the ETL process. Many of the required exercises could be addressed with basic SQL capabilities: joins, sorts, string manipulations, and validation of referential integrity.



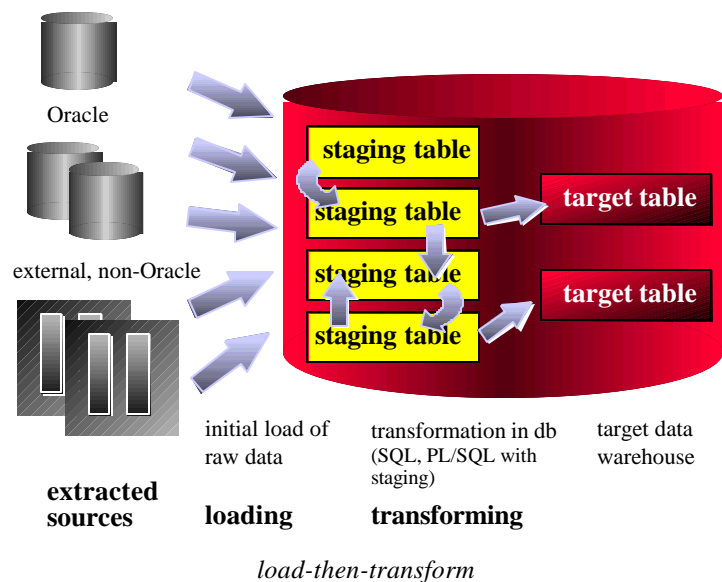
After successful cleansing and transformation, the new data is loaded into the database or existing data is updated via an external process. The primary database tasks are loading data, maintaining indexes and validating constraints.

The main risks and disadvantages in this architecture are:

- Scalability for most of the ETL steps has to be provided by an external mechanism outside the database. If the external mechanism fails to scale, it will become a bottleneck.
- Depending on the architecture, the external mechanism has to control the progress of the ETL process and provide recovery and restartability functionality for the ETL process.
- Many of these solutions are home-grown, which can be difficult to maintain and extend.
- The database capabilities are not fully utilized.

Loading into a database staging area for ETL processing

Different sources in various formats reside outside the database. Rather than using an external engine as the single point of control, the database is used. All raw data is loaded mostly unchanged in neutral staging structures. If the source systems are relational databases, the staging tables will be typical relational tables. If the source systems are non-relational, the data may be staged in tables with columns like VARCHAR2(4000) for further processing inside the database. After successfully loading the external data unmodified into the database, the transformation steps take place inside the database. This is the serial approach *load-then-transform*.

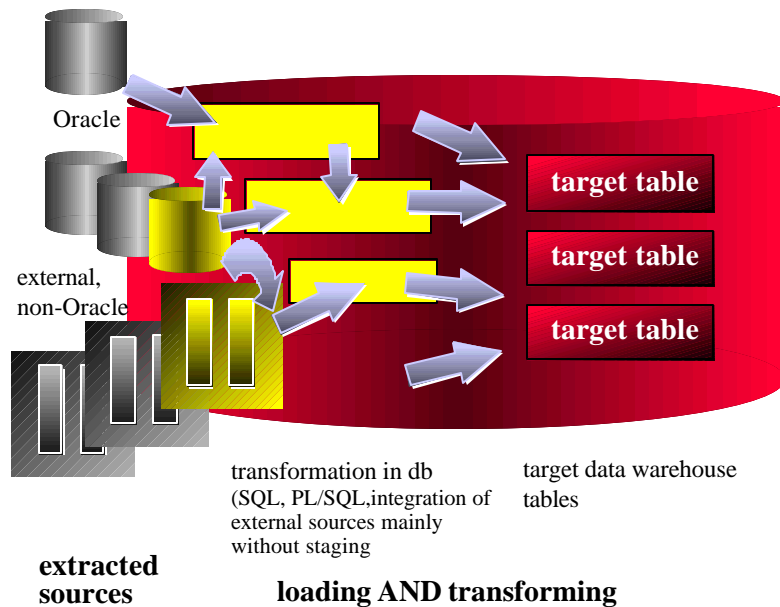


The main risks and disadvantages in this architecture are:

- The transformation process is interrupted by storing not only intermediate results but also the original raw data from the source systems in the database.
- Though most of the transformation tasks can be addressed with database functionality like SQL and PL/SQL, these languages might not be optimized to handle specifically all ETL issues.

The new paradigm

Beginning with Oracle9i, Oracle's database capabilities are significantly enhanced to address specifically some of the tasks in ETL environments. The ETL process flow can be changed dramatically and the database becomes the integrated data transformation engine.



transform-while-loading

By taking advantage of new, Oracle9i functionality, many process steps are no longer necessary, while others can be implemented much more efficiently and scalably. Oracle9i is not designed for *transform-then-load* or *load-then-transform* paradigms. Instead, Oracle9i provides a *transform-while-loading* paradigm.

ETL CAPABILITIES OF ORACLE9I

Oracle9i offers a wide variety of new capabilities to address all the issues and tasks relevant in an ETL scenario. It is important to understand that the database offers an infrastructure, allowing customers and tools to choose the necessary ingredients for their ETL solution. Oracle9i provides a set of features which enable more efficient ETL processing. The following will discuss Oracle9i's new ETL functionality and enhancements to existing functionality in more detail.

Change Data Capture

Change Data Capture is an important element to optimize the extraction from a source system. Instead of re-extracting the complete data from the source, only

the changed and newly inserted data between two extractions is of interest. Unfortunately, identifying this data is often either difficult or simply impossible. In many packaged-applications environments, the DBA cannot modify the underlying schema to track data-changes. Furthermore, after identifying the changed data, it must be made accessible for the targets which need to get this information. You must ensure that a target gets the same changed data only once and that the changed data is accessible until all targets have been able to get it. Oracle Change Data Capture addresses both of these issues.

1. Oracle Change Data Capture offers the capability to capture changed data from Oracle data sources; this can be done either synchronously through Oracle's Replication framework, or asynchronously by 'sniffing' the archived redo logs with Oracle's LogMiner technology. The asynchronous change data capture works without any intervention on the source site and will be included in Oracle9i, Release 2.
2. Oracle Change Data Capture provides the capability not only for capturing change data but for publishing it and allowing applications to subscribe to the change data in a controlled fashion.

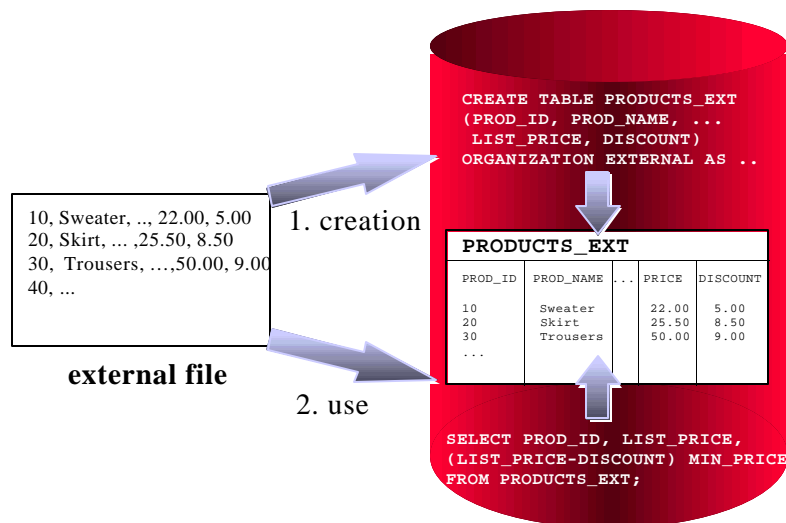
Additionally, Change Data Capture offers an extensible documented API for 3rd party vendors to integrate external data drivers for non-Oracle sources into the Oracle Change Data Capture framework.

Oracle's Change Data Capture framework can be used to optimize the extraction portion of the ETL process and build the basic maintenance framework for the repetitive scheduled execution of the complete ETL process.

External Tables

Very often there is a need for interaction between data already residing in the target database and data in external data sources like flat files. Oracle9i's External Table feature allows external data to be exposed in the database like any other data residing in a regular table. As a result, the external table acts as a "virtual table" and can be queried and joined directly and in parallel without requiring the external data to be first loaded in the database, using the full power of SQL, PL/SQL, and Java. Any other new SQL and PL/SQL functionality discussed in this paper can be combined with the external table feature.

External Tables enable the pipelining of the transformation phase with the loading phase. The transformation process can be merged with the loading process without any interruption of the data streaming. It's no longer necessary to stage the data inside the database for comparison or transformation.



From a user's point of view, the main difference between External Tables and regular tables is that externally organized tables are read-only. No DML operations (update/insert/delete) are possible and no indexes can be created on them.

Example:

```
CREATE TABLE products_ext
(prod_id NUMBER, prod_name VARCHAR2(50), ...,
price NUMBER(6,2), discount NUMBER(6,2))
ORGANIZATION EXTERNAL
(
TYPE oracle_loader
DEFAULT DIRECTORY stage_dir
ACCESS PARAMETERS
( RECORDS DELIMITED BY NEWLINE
LOGFILE log_dir:'log_products_ext'
FIELDS TERMINATED BY ','
MISSING FIELDS ARE NULL
)
LOCATION ('new_prod1.txt','new_prod2.txt')
)
PARALLEL 5
REJECT LIMIT UNLIMITED;

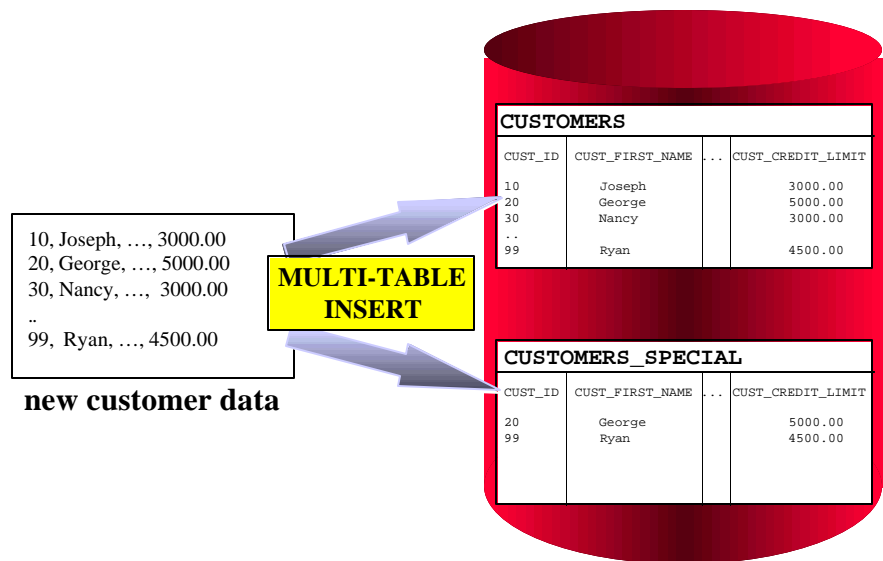
CREATE VIEW products_delta AS
SELECT prod_id, prod_name, ..., price as list_price,
(price-discount) AS min_price FROM products_ext;
```

Oracle9i's external tables are enhancing existing SQL*Loader functionality, and are especially useful for environments where the complete external source has to be joined with existing database objects and transformed in a complex manner.

Multi-Table Insert

Many times, external data sources have to be segregated based on logical attributes for insertion into different target tables. It is also frequent in data warehouse environments to fan out the same source data into several target objects. Multi-Table Inserts provide a new SQL command as solution for these kind of transformations, where data can be inserted into multiple target tables, or data can be inserted into exactly one of a number of possible target tables, depending on the business transformation rules.

Multi-table inserts offers the benefits of the INSERT . . . SELECT statement when multiple tables are involved as targets. Prior to Oracle9i, this functionality could be implemented using n independent INSERT . . . SELECT statements, thus processing the same source data n times and increasing the transformation workload up to n times. Alternatively, this functionality could be implemented using procedural logic, in which each row is individually examined to determine how to handle the insertion; this procedural solution lacked the access to high-speed access paths directly available in SQL.

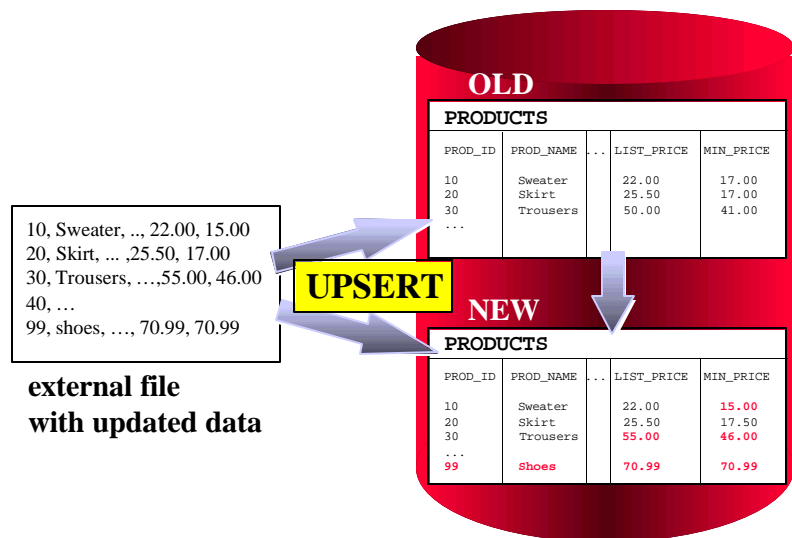


Example:

```
INSERT FIRST
WHEN cust_credit_limit >=4500 THEN
INTO customers_special VALUES(cust_id,cust_credit_limit)
INTO customers
ELSE
INTO customers
SELECT * FROM customers_new;
```

Upsert Functionality

Many times, the source system cannot distinguish between newly inserted or changed information during the extraction; for very complex transformations, it's sometimes nearly impossible to know the effect of changed source data. These scenarios means that the determination of whether a given row is entirely new, or simply a modification of a previously existing row, must be done in the data-warehouse rather than during the extraction from the source system. Oracle9i extends SQL in order to provide the ability to update or insert a row conditionally into a table, introducing the new SQL keyword MERGE.



Prior to Oracle9i, this functionality was expressed either as a sequence of DMLs or as PL/SQL loops operating on each row. Both of these approaches suffer from deficiencies in performance and usability. This new functionality overcomes these deficiencies and offers a new SQL statement. The new MERGE syntax has been proposed as part of the upcoming SQL standard.

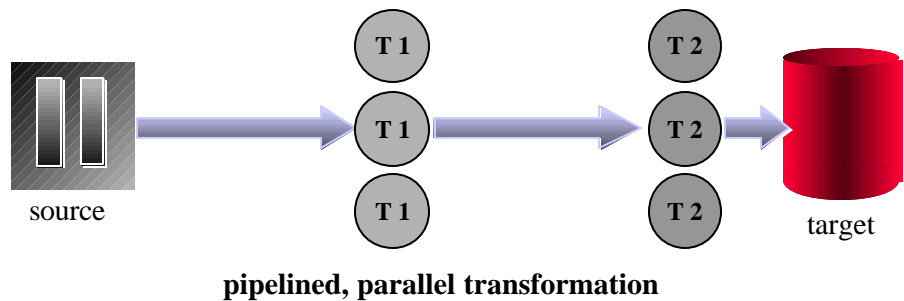
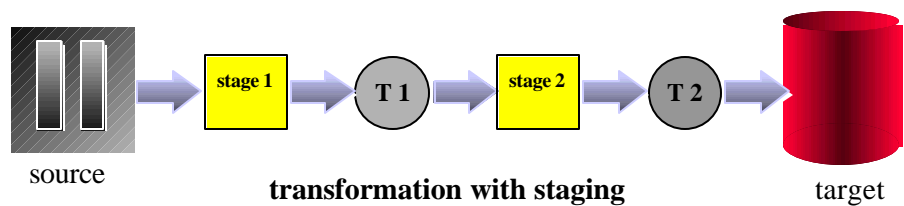
Example:

```
MERGE INTO products t
USING products_delta s
ON t.prod_id=s.prod_id
WHEN MATCHED THEN
UPDATE SET t.prod_list_price=s.list_price,
           t.prod_min_price=s.min_price
WHEN NOT MATCHED THEN
INSERT (prod_id,prod_name,...,prod_list_price,
       prod_min_price)
VALUES (s.prod_id, s.prod_name, ..., s.list_price,
       s.min_price);
```

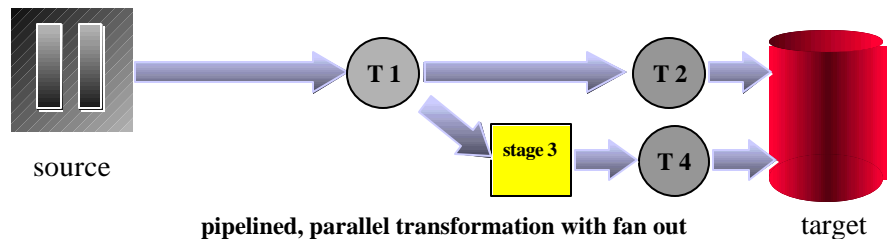
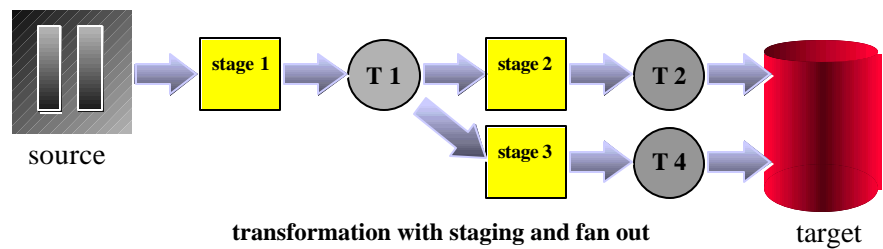
Table Functions

In the ETL process, the data extracted from a source system passes through a sequence of transformations before it is loaded into a data warehouse. Complex transformations are implemented in a procedural manner, either outside the database or inside the database using PL/SQL. When the results of a transformation become too large to fit into memory, they must be staged by materializing them either into database tables or flat files. This data is then read and processed as input to the next transformation in the sequence.

Oracle9i's table functions provide the support for pipelined and parallel execution of such transformations implemented in PL/SQL, C or Java. The scenarios mentioned above can be done without requiring the necessity the use of intermediate staging tables, which interrupt the data flow through various transformations steps.



Although a table function is part of a single atomic transaction, it provides the additional capability to *fan-out* data within the scope of an autonomous transaction. This can be used in a variety of ways, such as for exception or progress logging or to fan-out subsets of data to be used by other independent transformations.



A table function is defined as a function which can produce a set of rows as output; additionally, Oracle9i's table functions can take a set of rows as input. These sets of rows are processed iteratively in subsets, thus enabling a *pipelined* mechanism to stream these subset results from one transformation to the next before the first operation has finished. Furthermore, table functions can be processed transparently in parallel, which is equivalent to SQL operations like a table scan or a sort.

The following example shows the transparent usage of a table function within SQL. It will generate two new records based on one input record

Example:

```
CREATE FUNCTION StockPivot(p InputCursorType)
return ReturnCursorTypeSet
PIPELINED                                     -- pipelining
PARALLEL_ENABLE(PARTITION P BY ANY) is       -- enables
                                             parallel execution

    ret_rec ReturnCursorType;
begin
  FOR rec IN p LOOP
    ret_rec.Ticker := rec.Ticker;
    ret_rec.PriceType := "O";
    ret_rec.Price := rec.OpenPrice;
    PIPE ROW (ret_rec);                       -- first record for
                                             push back
    ret_rec.PriceType := "C";
    ret_rec.Price := rec.ClosePrice;
    PIPE ROW (ret_rec);                       -- second record for push
                                             back
  END LOOP;
  RETURN;
END;
```

The function 'StockPivot' can be used to generate another table from the 'Stocks' table in the following manner. When the StockTable is scanned in parallel, the table function will be parallelized, too.

```
INSERT INTO AlternateStockTable
SELECT * FROM
TABLE(StockPivot(CURSOR(SELECT * FROM StockTable)));
```

The ability to pipeline and parallelize table functions enables you in many cases to use PL/SQL, C or Java seamlessly together with SQL, like selecting from a table function or to pass results of SQL subqueries directly into a table function.

Transportable Tablespaces

Transportable Tablespaces are an elegant mechanism to share encapsulated data between several Oracle instances. Transportable Tablespaces give you the opportunity to move an encapsulated tablespace between databases physically rather than coping with it logically. Only the data dictionary (i.e. metadata) information of the tablespace is exported from the source and imported in the target system. Transportable Tablespaces are an extremely efficient mechanism for moving data from one Oracle database to another, because Transportable Tablespaces allow data to be moved without the data ever being unloaded or reloaded. Data transportation time becomes predictable and less error-prone.

The restrictions prior to Oracle9i were to guarantee the same operating system, the same database version and same blocksize. In current data warehouse environments, transportable tablespaces are mainly used to transport data from a central data warehouse to a dependent data mart or from an Operational Data Store (ODS) to a data warehouse. Leveraging transportable tablespaces as a transport mechanism between an OLTP source system and the data warehouse failed very often due to the restriction on database blocksize. Beginning with Oracle9i, the restriction of having the same blocksize for transportable tablespaces is removed.

The following new and enhanced capabilities of Oracle9i, which are targeted for general application usage, will be discussed here for their benefits in ETL environments.

Resumable Statements

If a long-running operation fails in the middle or at the very end of its processing, it can be a frustrating and expensive situation. Not only is all of the processing up to the time of failure lost, but the operation must entirely roll back before it can be restarted. With resumable statements, Oracle9i provides the ability to suspend and resume execution of long-running database operations in the event of correctable failures. Currently the types of errors from which a statement can resume are space limit and out-of-space errors.

When an operation suspends, the DBA has the opportunity to take the corrective steps to resolve the error condition. Alternatively, a procedure can be registered to automate the error correction. Once the error condition is solved, the suspended statement will automatically resume and continue operation. If the error is not corrected within an optionally-specified time limit, the statement will finally fail.

Parallel DML Operations

Parallel DML functionality has been provided since Oracle8.0 to speed up the processing of large data volumes. Prior to Oracle9i, a parallel INSERT into partitioned tables was constrained by the number of partitions. Oracle9i lifts this restriction, providing arbitrary parallelism for direct path INSERT into partitioned tables.

Partitioning and Partition Maintenance Operations

Oracle partitioning is one of the most important features for handling data warehouses in the terabyte range. A careful design of the data warehouse alleviates maintenance operations by taking advantage of the powerful and flexible partition maintenance operations. For example, a very common scenario, known as the 'rolling window' environment, is where data for a specific time window back in history is being stored in the data warehouse. When new data is added to the data warehouse, some of the existing data in the data warehouse will become aged out. Consider an environment where new data is added every month; because data of a new month is added, the data of the oldest month will become automatically obsolete and must be removed from the data warehouse. Using a range-partitioned table covering data on a monthly base enables you to perform the described scenario above with 2 partition maintenance commands: new data is added with an ALTER TABLE .. ADD PARTITION and the old data is removed with an ALTER TABLE .. DROP PARTITION.

Whereas local indexes are unaffected from the above mentioned operations, the entire global index is invalidated when a partition DDL is done. You have to rebuild all global indexes. This is no longer true for Oracle9i where you can choose whether a global index should be maintained as part of the DDL operation or to invalidate and rebuild a global index after the partition maintenance operations.

Beginning with Oracle9i, Oracle's partitioning methods have been expanded to including partitioning based on lists of distinct values. This can become very helpful in ETL environments where a source file includes data in different logical states, thus making different transformations necessary. Assuming this data is flagged, those data can be logically divided into separate partitions during the loading into a list-partitioned table.

CONCLUSION

Oracle9i introduces the first real business intelligence platform on the market. It is another milestone of the Oracle database as the most successful RDBMS in the data warehousing market. By introducing these new data warehousing features and enhancing existing functionality, Oracle increases the value and importance of the database as the Business Intelligence platform in every data warehouse environment.

Oracle9i provides a broad toolkit of powerful functionality specifically targeted towards the ETL process, enabling you to define your own specific ETL process and data flow, and taking advantage of the reliability, scalability and performance of Oracle's proven database technology.



ETL Processing within Oracle9i
June 2000

Author: Hermann Baer
Contributing Authors:

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
www.oracle.com

Oracle Corporation provides the software
that powers the internet.

Oracle is a registered trademark of Oracle Corporation. Various
product and service names referenced herein may be trademarks
of Oracle Corporation. All other product and service names
mentioned may be trademarks of their respective owners.

Copyright © 2000 Oracle Corporation
All rights reserved.