Oracle Data Warehousing – Pushing the Limits

Jason Laws

Principal Consultant

WhereScape Consulting

Introduction

Oracle is the leading database for data warehousing. This paper covers some of the reasons why. When faced with large data volumes, large user communities, short processing times, and limited budgets, you have to use your resources smartly. Using a case study, some techniques used to get the most from Oracle in extreme circumstances are discussed.

Case Study

Project Description

The project scope was to provide a mechanism to store industry wide market share data which included extensive competitor sales information from six external sources. All six external sources were provided in the same format.

The project team included two experienced data warehouse consultants (Kurt Gustafson and Jason Laws), a Project Manager and a DBA. Only four months was given to deliver the project.

It was planned that this project would be the first of many culminating in an enterprise wide data warehouse.

Data Description

The data came in a relatively simple format with three different record types for historical data and the same three record types for future projection data. Type 1 and type 2 records are summaries of type three records.

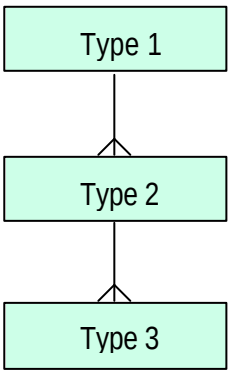

Figure 1 - Record Type Relationships

Data arrives each month on tapes containing 70 to 100 ASCII files, approximately 80Gb of raw data and 350 million rows.

It became evident early in the project that both data quantity and quality would be an issue.

Data Retention Period

There was an initial requirement to keep lowest level data online for five years. This equated to 21 billion rows of fact data.

Requirements changed one month after the initial system load was completed. The historic set of data is now kept for three years and the future projection data for one month. 35% of data is filtered out and not loaded.

Table	Data GB	Index GB	Num Rows
Fact Table: Set 1; Type 1	158.9	94.9	941,371,917
Fact Table: Set 1; Type 2	172.8	98.7	1,021,233,805
Fact Table: Set 1; Type 3	256.1	128.6	1,318,353,006
Fact Table: Set 2; Type 1	6.0	3.4	35,397,389
Fact Table: Set 2; Type 2	6.6	3.6	39,356,657
Fact Table: Set 2; Type 3	10.6	5.1	54,716,397
TOTAL	**611.1**	**334.4**	**3,410,429,171**

Hardware and Software

Initially, the production database was housed on a Sun E6500 running solaris 2.7. It had eight 400Mhz CPUs, 4GB of memory and used a hitachi 7700E disk array.

An Oracle Database was chosen for implementation, the project started on version 8.1.5 and ended on 8.1.6.3. Since go live, the database has been upgraded to 8.1.7. Currently the database has an eighty gigabyte temp tablespace and a twelve gigabytes of rollback segments.

Logical and physical database design was conceived without a front-end tool in mind. Late in project, Oracle Discoverer was chosen as the front-end.

Logical Design

There was no need or requirement to build a transactional layer or operational data store (ODS). The snapshot nature of the data lent itself to a Star Schema data model. As this was the first data warehouse project, all dimensions had to be built from scratch.

The design incorporated six fact tables and twelve dimension tables. Some dimensions were very complex with one dimension having 96 columns!

Data Processing

Initially it was intended that all data would be loaded. Requirements changed shortly after go live. Data was reprocessed to exclude 35% of unwanted rows.

Traditional data warehouse processing steps usually include:

- Loading data into staging tables.
- Validating and transforming staging data and adding surrogate keys. A cursor loop is used to process data one record at a time. Transformations and surrogate keys are updated onto the staging table.
- Publishing staging data into fact tables - usually a simple insert statement.

Due to the sheer volume being processed, this traditional approach was estimated to run for 85 days each month. This meant another approach was required.

The chosen approach takes three days each month and involved:

- Producing new files in unix containing transformed and filtered data.
- Loading the new files into staging tables.
- Validating the staging data.
- Publishing the staging data into fact tables, including adding surrogate keys.

It is important that all processes scale, as data volumes can vary significantly.

Data transformation processing in unix was built using awk. Parallel processing was used whenever possible.

Direct path SQL*Loader was used in the load step. After the upgrade from 8.1.5 to 8.1.6, the load rate doubled to an unbelievable 40,000 rows per second. Load processing was manually parallelised, and limited to ten concurrent streams.

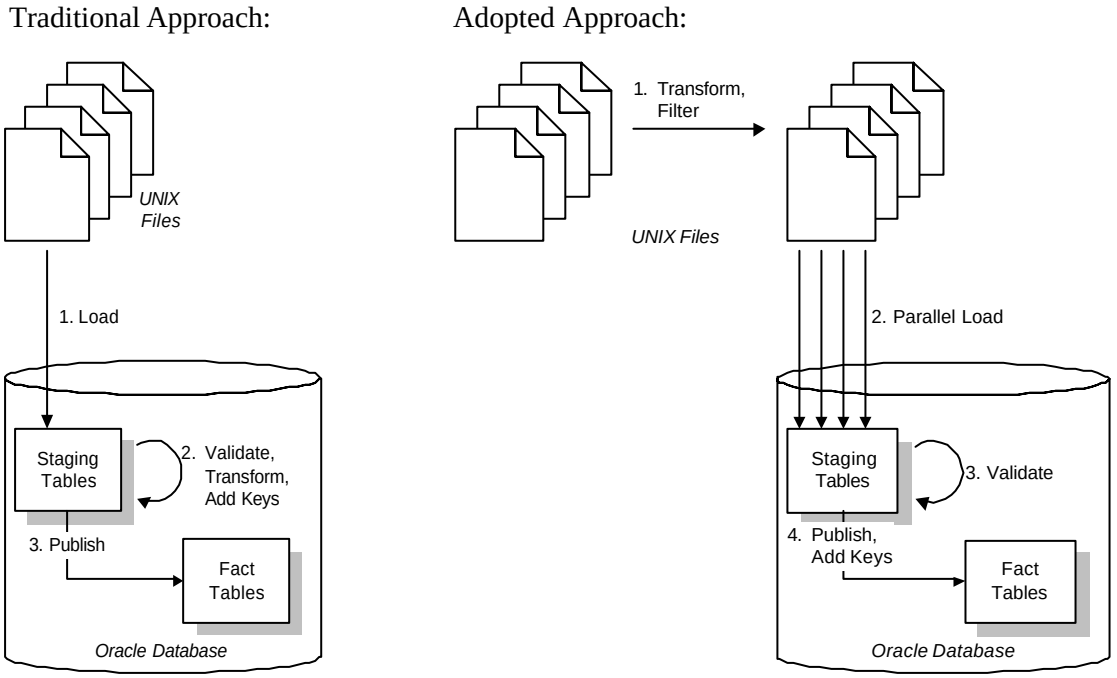

Figure 2 - Comparison of Processing Approaches

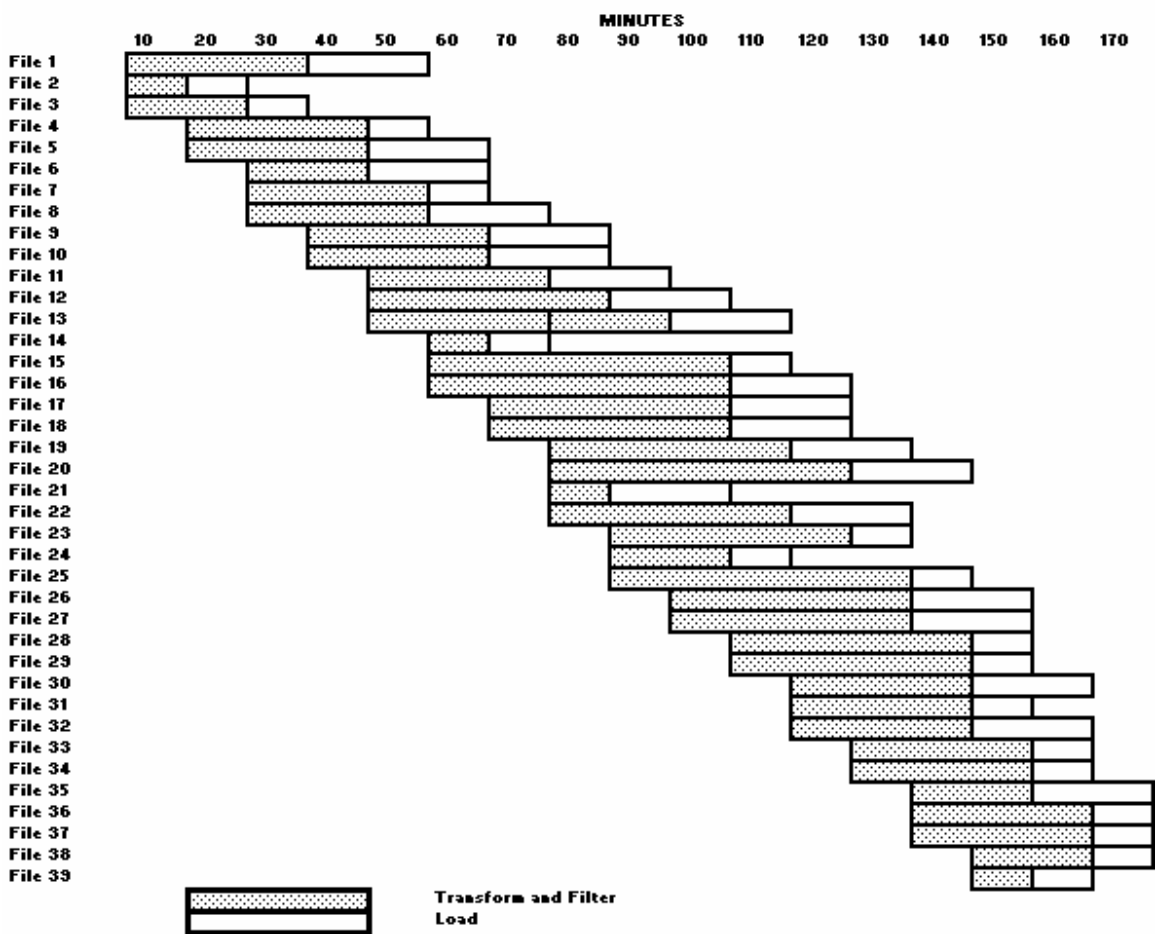

Figure 3 - Transform, Filter and Load Timeline

Staging Tables were partitioned by the six sources and then hash subpartitioned so each subpartition contained between 500,000 and 2,000,000 rows in an average month.

In the validate step only the lowest level record type (type 3) was processed. This record type contained all possible values of all summary records. The unique key of the highest level staging record is also on the two lower level staging records.

The validate process looped on approximately fifty sub partitions of the type 3 staging table using dynamic SQL to directly name sub partitions. This was necessary as Pl/SQL does not support subpartition naming in FROM clauses. Two lists of Oracle Internal row ids were built. One contained row ids of rows that passed validation and the other rows that failed validation. Then the row ids were converted to the highest level primary key values.

Fact Tables were partitioned (but not sub partitioned) by the six sources, by year and by month. Each fact table has more than two hundred partitions and each partition is in its own tablespace. The fact tables have approximately thirty bitmap indexes each.

Tablespaces containing older partitions are made read only. Read only tablespaces are not backed up in regularly, but are backed up once when they are made read only.

The publish process included the following steps:

- Loop on the partitions of the fact table.
- Exchange the partition with a dummy table.
- Drop all indexes off the dummy table.
- Then loop on the subpartitions of the equivalent staging table partition and insert "passed" records.
- Once all subpartitions of the partition have been processed, rebuild indexes.
- Exchange the dummy table back with the partitioned table.

User Queries

User's run queries directly against one billion row fact tables, as data is hard to summarise. Simple queries run under one minute and complex queries run under fifteen minutes. The database has been tuned in favour of Star Transformations.

At various times it was necessary to override the optimiser with an undocumented, unsupported database parameter: _ALWAYS_STAR_TRANSFORMATION.

Risks

Systems that push the limits carry with them risks such as:

- Transition to support being difficult and requiring expert planning.
- Constantly being at the mercy of change.
- Performance related incidents being difficult to resolve.

Performance related incidents include:

- Update Performance suddenly degraded. This was caused by dimensions growing and exceeding memory thresholds. The incident was resolved by creating smaller snapshots of dimensions for update processing.
- After transition to support, the database version was upgraded from 8.1.6 to 81.7 without proper planning. Update Performance suddenly degraded 300%. The problem was tracked to one complex SQL statement taking ten times longer than normal. It was fixed using a "modified" Oracle Plan stability approach. Plans were generated for all SQL statements in an 8.1.6 database and moved into 8.1.7. This solution protects against future upgrades. It includes functionality to test new versions of Oracle by turning off plan stability temporarily or permanently.
- ORA-00600 database bugs occurred intermittently. The bugs were caused by extensive bitmap index use and a Local Prefixed Composite Primary Key Index. A "try again" work around was added to the publish code. Several database patch sets were applied.

Key Findings

Critical Success Factors

- Use a good database.
- Make the most of what you've got.
- Think outside the square.
- Use horizontal project teams.
- Have at least one person who understands all the technologies being used from end to end.
- Focus on the back end, the front end will sort itself out.
- Control key processes at the lowest possible level.

Important Back End Findings

- Oracle 8i can do almost everything needed to build a large warehouse.
- What it doesn't do fast enough with large data volumes is ungrouping grouped data files.
- Have a good DBA dedicated to the project.
- Tune, tune and tune again.
- Use the best disk you can afford.
- Layout SAN disk at the LUN level.
- Be careful with NAS disk.

Important Front End Findings

- Build the back end to be tool independent.
- Any Front End will do.
- Forget MOLAP.

Summary

When faced with large data volumes, large user communities, short processing times, and limited budgets, you have to use your resources smartly.

You have to:

- Use a good database – Oracle 8i.
- Use the right hardware.
- Make the most of what you've got.
- Think outside the square.
- Use horizontal project teams.
- Have at least one person who understands all the technologies being used from end to end.
- Focus on the back end. The front end will sort itself out.
- Control key processes at the lowest possible level.
- Tuned the database carefully. Oracle can be tuned to allow user queries to run against billion row plus fact tables in under 10 seconds.