Technical overview: Ad-hoc Aleph Reporting System -Harvard University Library

System requirements

1. Report information must be extracted from the production database to another server in order to prevent contention between analysts and the public for computer resources.

2. A coherent data model must be built from the base Aleph tables. Table and column names are to be renamed and data types harmonized to clarify primary and foreign key relationships.

3. Creation of the data must not require any downtime on the production Aleph database.

4. The reporting data must be renewed in a timely manner: nightly (preferred) or weekly

System technical overview

1. Uses Oracle materialized views. A view is a SELECT statement that shows sub-sets of data contained in a table and is used to simplify data relationships or aggregate values. Whereas views are virtual tables managed in memory and consume no mass storage, the result set of a materialized view is permanently written to disk. The advantage of the materialized view (also known as a snapshot) is that the data can be extensively indexed. Oracle designed this mechanism to meet the requirements of large data warehouse applications.

2. Uses desktop query tool used in previous generation of Harvard library reporting system. As the data is transferred, the view creation statements convert the information into a rational data model that can be accessed using standard desktop query tools. The view definition statements also permit us to create more descriptive names for the native Aleph data objects (ORDERS instead of Z68, etc.)

3. Operates over a network link to move the data from the production server to a reporting environment. A *database link* is an Oracle interface that exchanges data between networked database servers. The link defines a TCP/IP conduit that is a combination of the instance identifier (SID), server computer name, IP port and log-on user account. Nearly all structured query language (SQL) and data definition language (DDL) statements work over links.

Software Code Design

PL/SQL is used to wrap all view creation statements inside anonymous code blocks - this allows the incorporation of more sophisticated error handling via exception catching. When errors occur, automated email is generated to the operations support group.

The system has been designed to be as portable as possible. There are Harvard specific-features such as the budget TUB, ORG and FUND columns, but on the whole other Aleph customers can customize the code to their institutional needs.

Utility Functions

Ex Libris date formats and dollar amounts are represented as text strings and not as native SQL

data types. A set of functions are included in-line in the view SELECT statement to convert specific column values to standard form.

MARC Data Parsing

Most of the MARC fields are parsed through the equivalent of regular expression matching using PL/SQL. (Since the data is stored in a RAW TEXT column, it cannot be queried using native SQL.) For HOL and BIB MARC data, we have limited reporting to approximately 50 fields. Several are copied from the Z13 table and the rest are individually parsed.

View Creation Process

The DML statement for the creation of the **INVOICE_LINES** view illustrates how the reporting data is generated. Please note the following:.

- The CREATE statement renames HVD50.Z75 to INVOICE_LINES
- The AS clause renames columns.
- The bibliographic primary key is extracted as a sub string from the Z75_REC_KEY and converted to a true NUMBER data type.
- Trailing spaces are removed. This is necessary because most Aleph columns are defined as CHAR.
- Aleph date strings in the format 'YYYYMMDD' are error checked and converted to DATE data type.
- Dollar amounts that are left zero padded and contain an implied decimal point are converted to FLOAT data types.

```
CREATE SNAPSHOT INVOICE_LINES REFRESH WITH ROWID AS
SELECT
   TO_NUMBER(SUBSTR(Z75_REC_KEY,   1, 9))         AS ADM_DOC_ID,
   TO_NUMBER(SUBSTR(Z75_REC_KEY,  10, 5))         AS ADM_SEQUENCE_ID
   UPPER(RTRIM(SUBSTR(Z75_REC_KEY_2, 1, 20)))     AS VENDOR_ID,
   LTRIM(RTRIM(SUBSTR(Z75_REC_KEY_2, 21, 15)))    AS INVOICE_NUMBER,
   TO_NUMBER(LTRIM(SUBSTR(Z75_REC_KEY_2, 36, 5))) AS INVOICE_LINE_NUMBER,
   LTRIM(RTRIM(Z75_I_OBJECT_CODE))                AS OBJECT_CODE,
   Z75_I_CREDIT_DEBIT                             AS CREDIT_OR_DEBIT,
   TO_AMOUNT_REPORTING(Z75_I_LISTED_PRICE)        AS PRICE_LISTED,
   TO_AMOUNT_REPORTING(Z75_I_NET_AMOUNT)          AS AMOUNT_NET,
   TO_AMOUNT_REPORTING(Z75_I_TOTAL_AMOUNT)        AS AMOUNT_TOTAL,
   Z75_I_NO_UNITS                                 AS NUMBER_OF_UNITS,
   Z75_I_NOTE                                     AS INVOICE_NOTE,
   TO_DATE_REPORTING(Z75_I_DATE_FROM)             AS DATE_FROM,
   TO_DATE_REPORTING(Z75_I_DATE_TO)               AS DATE_TO,
   Z75_I_DATE_RANGE                               AS DATE_RANGE
FROM HVD50.Z75@ALEPH0.LMS01.HARVARD.EDU;
```

System Administration

Hardware Requirements

The reporting machine is a 2 processor Sunfire 480. The system is completely contained on 2 internal 72gb disks. We do not expect the database to grow larger than 40 gigabytes (10% of the 424 gigabytes now used by the production instance). The network connection between it and the production system is gigabit Ethernet.

Operating Environment

The production Aleph environment is Aleph version 15.2.1 under Oracle 8.1.7. The reporting database is an Oracle 9.2.0 environment.

Security

Separate accounts are used to monitor and measure usage by library. These accounts have only select privilege on the objects in the reporting instance; this prevents user access to the database link to the production server.

Database maintenance

Jobs are run nightly to update the reporting database. Currently batch jobs take 10-12 hours to complete. An index analyze is performed following the load; this process completes in about 25 minutes.