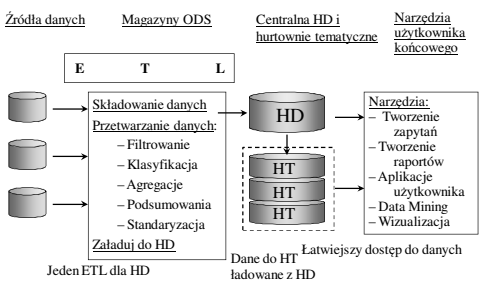
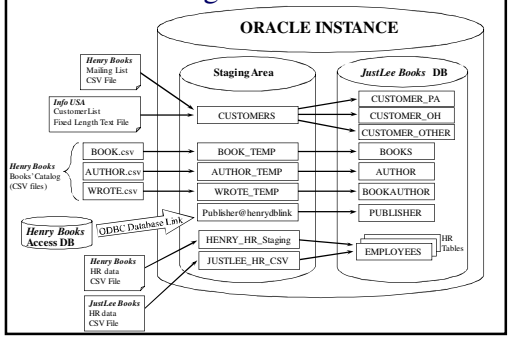
# Working with External/Internal Data

Materiały udostępnione przez prof. Janusza Szczupułę,
Carnegie Mellon University, Pittsburgh, USA

## Zależne hurtownie tematyczne i magazyny ODS




## Integration Plan




## Introduction

- Oracle provides tools for working with external data for:
  - Importing data from text files
  - Accessing external text files as if they were "normal" tables
  - Linking Oracle database with other datasets
- In principle this requires:
  - Extracting data from multiple external sources and moving the data into temporary tables (in a staging area)
  - Transforming the new data based on user requirements
  - Inserting "cleaned up" data into the main database tables



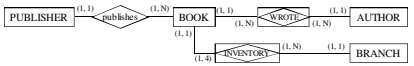
## A merger scenario

- *JustLee Books*, a book store chain purchased a small privately owned company, *Henry Books*
- Now they need to integrate the data from *Henry Books* into their existing database system
- The integration must consider:
  I. Customers data
  II. Books' Catalog (*Books and Authors*)
  III. Publisher data
- Extending current DB to include HR information



## Source of data

- *Henry Books* inventory database[1]:

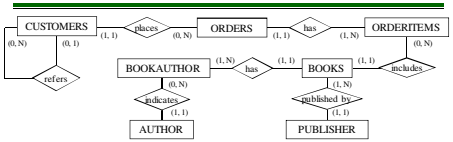



**PUBLISHER** (Publisher_Code, Publisher_Name, Publisher_City)
**BOOK** (Book_Code, Title, Price, Type, Paperback, Publisher_Code@)
**WROTE** (Author_Num@, Book_Code@, Sequence)
**AUTHOR** (Author_Num, Author_Last, Author_First)
**BRANCH** (Branch_num, Branch_name, Branch_location, Num_employees)
**INVENTORY** (Branch_num@, Book_code@, On_hand)

- Customer data:
  - *Henry Books* mailing list: maintained independently in a text file
  - Additional data: new customers' information
- HR data from both companies – available in text files

[1] Model adopted from "A Guide to SQL" by Philip Pratt, Course Technology, 2005

## Target – *JustLee Books* database




**CUSTOMERS** (Customer#, Lastname, Firstname, Address, City, State, Zip, Referred®, Region, Email)
**ORDERS** (Order#, Orderdate, Shipdate, Shipstreet, Shipcity, Shipstate, Shipzip, Shipcost, Customer#®)
**ORDERITEMS** (Order#®, Item#, ISBN®, Quantity, Paideach)
**BOOKS** (ISBN, Title, Pubdate, PubID®, Cost, Retail, Discount, Category)
**BOOKAUTHOR** (ISBN®, AuthorID® )
**AUTHOR** (AuthorID, Lname, Fname)
**PUBLISHER** (PubID, Name, Contact, Phone)

Model from "Oracle 11g: SQL" by Joan Casteel, Course Technology, 2010

## A merger scenario (cont.)

- Each source dataset is specific in terms of its:
  - Data structures
  - Level of integrity enforcement
  - Standards / Conventions
  - Data formats, etc.
- Nevertheless, the data must be integrated into the Oracle database maintained by *JustLee Books*
- At the same time, *JustLee Books* need to reevaluate their operation and to modify the existing database model in order to include new information as well as to implement new details.

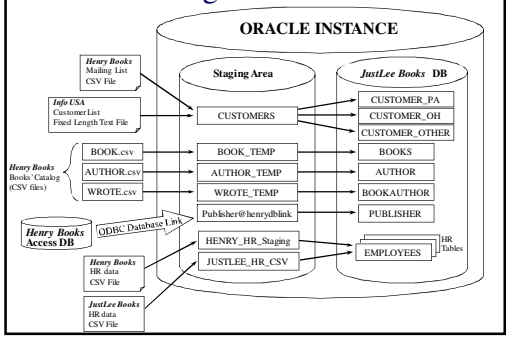


## A merger scenario (cont.)

- *JustLee Books* is interested in separating their customer data geographically:
  - This requires splitting customer data into separate customer tables based on their location: Pennsylvania (PA) customers, Ohio (OH) customers, "Other" customers
- The existing *CUSTOMERS* table with its current content is moved into staging area as an aggregation point for all customer data prior to creating the three customer tables.
  - With the combined data set and the new operational model, *JustLee* will no longer be tracking customer referrals



## Integration Plan

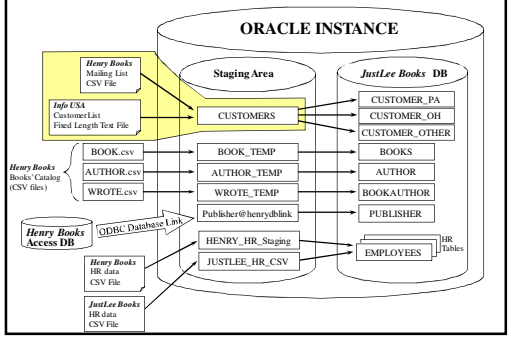



## Integration – Part I

Focus on Customer information

- Source – two lists:
  - *Henry Books* customer mailing list, and
  - New customer data from *InfoUSA.com*
- Format – comma separated values (CSV) and fixed length format
- Target – three Oracle tables containing three groups of customers based on the customer's state: PA, OH, and "Other"
- Method – we append new customer data into a table in the ***Oracle staging area***. Then we move "cleaned up" data to appropriate table(s) using DML statements.

## Part I: Customers' Data

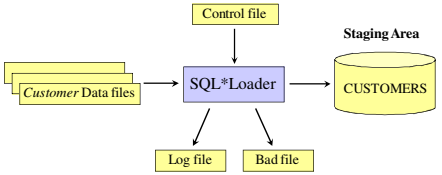


## Extracting Data with SQL*Loader

- Oracle's SQL*Loader:
  - Reads from multiple input files
  - Loads data into one or more database tables
  - Handles files with fixed-length as well as variable-length records
  - Supports a number of different data types
  - Uses Oracle's SQL functions to manipulate data
  - Includes functionality for dealing with "whitespace", delimiters, and Null values

Source: http://oreilly.com/catalog/orsqlloader/chapter/ch01.html

## SQL*Loader (cont.)






## SQL*Loader (cont.)

- Data files – include records to be inserted or appended into existing table(s)
  - Each record must match the structure of the table where it is to be inserted
  - Able to load multiple data files into multiple tables
- Control file – includes commands to be executed and appropriate parameters that define the process
- Log file – captures the server feedback
- Bad file – stores rejected records



## *Mailing List* data file – CSV

Content of the 'HenryCustList.csv' file was extracted from *Henry Books*' proprietary contact management system:

```
Henry Books Mailing List
Exported from Henry Books' System in October 2011

Customer Number, Address1, Type, City, State, Zip, Name
"500","123 Main St., Apt 2","Corporate","Pittsburgh","PA","15222",Johnson~Fred
"505","7477 Elm St","Individual","Monroeville","PA","15146",Bellows~Jim
"510","23 Oak Ln","Corporate","Pittsburgh","PA","15210",Hawk~Anne
"515","767 View Dr","Individual","Rockville","MD","20847",Ellis~Jen
"520","3766 Blue Way","Individual","Arlington","VA","22201",Bruny~Phil
"525","11 Redbird Rd","Corporate","Alexandria","VA","22301",Downing~Juanita
"530","1 Fawn Circle","Individual","Wheeling","WV","26003",Lowe~Prada
"535","45 Green Way","Corporate","Oakland","PA","15213",Wald~Beatrice
"540","767 Fly Way","Individual","Youngstown","OH","44501",Gruno~Pina
"545","987 Count Dr","Individual","Cleveland","OH","44101",Fisk~Ren
"550","7676 Egan Point","Pittsburgh","PA","11122"
```

Missing values:

## *Mailing List* data file – CSV

- Formatting
  - Some fields surrounded by double-quotes, others are not
- Mismatched Data
  - Third item in each row is *Customer Type* and is not needed for the Customers table
  - Customer name field includes first and last name separated by tilde "~" character
- Exceptions
  - Address in first data row is too long for the column in the database table (*customer number = 500*)
  - Last record is incomplete (*customer number = 550*)

## *Mailing List* control file – CSV

- Control file (*henry.ctl*) defines the loading process:

```
LOAD DATA
  INFILE  'c:\DataLoad\HenryCustList.csv'
  BADFILE 'c:\DataLoad\BadFile1.txt'
  APPEND
  INTO TABLE customers
  FIELDS TERMINATED BY ',' OPTIONALLY enclosed by '"'
    (Customer#,
     Address,
     CustType FILLER,
     City,
     State,
     Zip,
     LastName TERMINATED BY '~',
     FirstName )
```

## *Mailing List* control file – CSV

1. Set filename of source file and location/name of file to write rejected records to
```
INFILE  'c:\DataLoad\HenryCustList.csv'
BADFILE 'c:\DataLoad\BadFile1.txt'
```
2. Define delimiter information
```
FIELDS TERMINATED BY ',' OPTIONALLY enclosed by '"'
```
3. Indicate field that is to be skipped during import
```
CustType FILLER,
```
4. Define different delimiter for a particular field
```
LastName TERMINATED BY '~',
```



## SQL*Loader – Control file

- Possible Commands:

Insert – inserts records into a table only if the target table is empty

Append – appends record to a table regardless of whether the target table is empty or not

Replace – empties the table and then adds records

Truncate – truncates table before starting the load



## SQL*Loader command – CSV

- The loader command is executed from the "Command Line" (i.e., **cmd.exe**):

```
sqlldr control = c:\dataload\Henry.ctl
       log = c:\dataload\log1.txt
       userid = casteel/tiger
       skip = 4
```

**Skip 4** option – skips the first 4 lines in the data file (which includes comments and names of the columns)

**Log** – designates the location of the log output file from the data load



## *Mailing List* log file – CSV

- Header of the log file
  - Number of rows to load
  - Number or rows skipped at top of file
  - Errors allowed for successful run
  - Table into which the records will be loaded and method used (*Truncate, Replace, Append, or Insert*)
- Imported Columns
  - Lists columns' names, columns' positions, terminating character, and the data type
- Errors
  - Lists rejected records and reason for rejection – these rows are added to the "BAD" file
- Summary
  - Lists number of rows loaded
  - Number of rows with errors

## *Mailing List* log file – CSV

- Partial output of the log file, *log1.txt*:

```
Number to load: ALL
Number to skip: 4
Errors allowed: 50

Table CUSTOMERS, loaded from every logical record.
Insert option in effect for this table: APPEND

Column Name          Position   Len   Term Encl Datatype
-------------------- ---------- ----- ---- ---- ---------------------
CUSTOMER#                 FIRST     *   ,  O(") CHARACTER
ADDRESS                    NEXT     *   ,  O(") CHARACTER
CUSTTYPE                   NEXT     *   ,  O(") CHARACTER
  (FILLER FIELD)
CITY                       NEXT     *   ,  O(") CHARACTER
STATE                      NEXT     *   ,  O(") CHARACTER
ZIP                        NEXT     *   ,  O(") CHARACTER
LASTNAME                   NEXT     *   ~  O(") CHARACTER
FIRSTNAME                  NEXT     *   ,  O(") CHARACTER
```

## *Mailing List* log file (cont.)

```
Record 11: Rejected - Error on table CUSTOMERS, column ZIP.
Column not found before end of logical record (use TRAILING
   NULLCOLS)
Record 1: Rejected - Error on table CUSTOMERS, column ADDRESS.
ORA-12899: value too large for column
   "JUSTLEE"."CUSTOMERS"."ADDRESS" (actual: 22, maximum: 20)

Table CUSTOMERS:
  9 Rows successfully loaded.
  2 Rows not loaded due to data errors.
  0 Rows not loaded because all WHEN clauses were failed.
  0 Rows not loaded because all fields were null.

Total logical records skipped:          4
Total logical records read:            11
Total logical records rejected:         2
```

## Result

◇The *CUSTOMERS* table after the loading information from *Henry Books*:

```
SQL> SELECT customer#, lastname, firstname,
  2         firstname, address, city, state, zip
  3  FROM   Customers;

CUSTOMER# LASTNAME  FIRSTNAME ADDRESS          CITY          ST ZIP
--------- --------- --------- ---------------- ------------- -- -----
     1001 MORALES   BONITA    P.O. BOX 651     EASTPOINT     FL 32328
     1002 THOMPSON  RYAN      P.O. BOX 9835    SANTA MONICA  CA 90404
     1003 SMITH     LEILA     P.O. BOX 66      TALLAHASSEE   FL 32306
     1004 PIERSON   THOMAS    69821 SOUTH AVE  BOISE         ID 83707
     1005 GIRARD    CINDY     P.O. BOX 851     SEATTLE       WA 98115
     1006 CRUZ      MESHIA    82 DIRT ROAD     ALBANY        NY 12211
     . . .
      505 Ballows   Jim       7477 Elm St      Monroeville   PA 15146
      510 Hawk      Anne      23 Oak Ln        Pittsburgh    PA 15210
      515 Ellis     Jen       767 View Dr      Rockville     MD 20847
      520 Bruny     Phil      3766 Blue Way    Arlington     VA 22201
      525 Downing   Juanita   11 Redbird Rd    Alexandria    VA 22301
      530 Lowe      Prada     1 Fawn Circle    Wheeling      WV 26003
      535 Wald      Beatrice  45 Green Way     Oakland       PA 15213
      540 Gruno     Pina      767 Fly Way      Youngstown    OH 44501
      545 Fisk      Ren       987 Count Dr     Cleveland     OH 44101

29 rows selected
```

*JustLee Books*

*Henry Books*

## *Mailing List* bad file – CSV

◇"Bad File" from SQL*Loader:

- Includes records from source file that were rejected
- Allows these records to be corrected and later imported using SQL*Loader
- Indicates corrections need to made these records suitable for successful import

Missing *Customer Type & Name:*

```
"550","7676 Egan Point","Pittsburgh","PA","11122"
"500","123 Main Street, Apt 2","Corporate","Pittsburgh","PA","15222",Johnson-Fred
```

*Address* too long

## *InfoUSA* data file – fixed width

◇Content of the '*info_usa.txt*' file:

```
ID   Region LastName  FirstName  MI Address          City         St Zip
800  East   Flemming  Rena       J  13 46th St.      Millvale     pa 15209
801  North  Wilkenson Thad       l  11822 Rte. 19    Pittsburgh   pa 15221
802  West   Saunders  Olga          441 Reynolds St  Pittsburgh   pa 15221
803  West   Rialto    Pierce     Q  67 West 17th Ave Columbus     oh 43085
804  North  O'Connell Sommers       88 Circle Dr     Columbus     oh 43085
805  South  Powell    Drayton    p  601 North Ave    Wilkins      pa 15146
806  East   Inud      Floyd         25 Summer Rd     Cranberry    oh 16066
807  East   Tylon     Gretchen   N  146 18th St.     Cleveland    oh 44102
808  East   Eldridge  Florence   w  8585 Eigth Ave   Detroit      mi 48202
809  West   Gundle    Wendy      C  90 Jump St       Indianapolis in 46217
810  South  Drake     William       45 Degree Rd     Pittsburgh   pa 15217
```

Columns not needed

UPPER case required



## *InfoUSA* control file – fixed width

◇We need to append new customer records to the *CUSTOMERS* table in the staging area without losing any information already existing in the table

◇The *info_usa.txt* file includes columns not needed for target data model: MI (middle initial) & Region

- In delimited data: FILLER FIELDS are used
- For fixed width data we ignore those positions

◇The "State" column values must be stored in uppercase characters

- We can introduce UPPER function in control file

## *InfoUSA* control file – fixed width

```
LOAD DATA
  INFILE  'c:\DataLoad\info_usa.txt'
  BADFILE 'c:\DataLoad\BadFile2.txt'
  APPEND
  INTO TABLE Customers
    (Customer# position(01:03),
     Lastname  position(13:25),
     Firstname position(26:36),
     Address   position(40:57),
     City      position(58:70),
     State     position(71:73) "UPPER(:State)",
     Zip       position(74:79) )
```

## SQL*Loader command – fixed width

◇The loader command executed from the "Command Line":

```
sqlldr control = c:\DataLoad\infoUSA.ctl
       log = c:\DataLoad\log2.txt
       userid = casteel/tiger
       skip = 1
```

## *InfoUSA* log file – fixed width

◈Partial output of the log file, *log2.txt*:

```
Control File:   c:\DataLoad\infoUSA.ctl
Data File:      c:\DataLoad\info_usa.txt

Number to load: ALL
Number to skip: 1
Errors allowed: 50

Table CUSTOMERS, loaded from every logical record.
Insert option in effect for this table: APPEND

Column Name             Position   Len  Term Encl Datatype
----------------------- ---------- ----- ---- ---- ---------------------
CUSTOMER#                    1:3     3           CHARACTER
LASTNAME                   13:25    13           CHARACTER
FIRSTNAME                  26:36    11           CHARACTER
ADDRESS                    40:57    18           CHARACTER
CITY                       58:70    13           CHARACTER
STATE                      71:73     3           CHARACTER
    SQL string for column : "UPPER(:State)"
ZIP                        74:79     6           CHARACTER

Table CUSTOMERS:
  11 Rows successfully loaded.
  0 Rows not loaded due to data errors.
```

## Result

◈ The *CUSTOMERS* table after the appending additional customers' information from *Info USA*:

```
SQL> SELECT customer#, lastname, firstname,
  2         firstname, address, city, state, zip
  3  FROM   Customers;

CUSTOMER# LASTNAME   FIRSTNAME  ADDRESS            CITY         ST ZIP
--------- ---------- ---------- ------------------ ------------ -- -----
      1001 MORALES    BONITA     P.O. BOX 651       EASTPOINT    FL 32328
      1002 THOMPSON   RYAN       P.O. BOX 9835      SANTA MONICA CA 90404
      1003 SMITH      LEILA      P.O. BOX 66        TALLAHASSEE  FL 32306
      1004 PIERSON    THOMAS     69821 SOUTH AVE    BOISE        ID 83707
      1005 GIRARD     CINDY      P.O. BOX 851       SEATTLE      WA 98115
      1006 CRUZ       MESHIA     82 DIRT ROAD       ALBANY       NY 12211
        . . .
       505 Bellows    Jim        7477 Elm St        Monroeville  PA 15146
       510 Hawk       Anne       23 Oak Ln          Pittsburgh   PA 15210
       545 Fisk       Ren        987 Count Dr       Cleveland    OH 44101
        . . .
       800 Flemming   Rena       13 46th St.        Millvale     PA 15209
       801 Wilkenson  Thad       11822 Rte. 19      Pittsburgh   PA 15221
       802 Saunders   Olga       441 Reynolds St    Pittsburgh   PA 15221
       803 Rialto     Pierce     67 West 17th Ave   Columbus     OH 43085
        . . .
40 rows selected.
```

*JustLee Books*

*Henry Books*

*InfoUSA*

## Additional SQL*Loader Capabilities

◈Accessing multiple input files in one process

◈Loading data into multiple table using WHEN

◈Using functions within the control file

– Any SQL expression that is valid in the VALUES clause of an INSERT statement can be used within SQL*Loader

e.g.:

◈UPPER (used in our example above)

◈TO_DATE

◈DECODE

◈etc.

Source: http://psoug.org/reference/sqlloader.html

## Loading from Multiple Files

```
LOAD DATA
INFILE  '[file1]'
INFILE  '[file2]'
APPEND
INTO TABLE [table_name]
FIELDS TERMINATED BY ","
       (FIELD1, FIELD2, FIELD3, . . .)
```



## Loading into Multiple Table

SQL*Loader can load data into multiple tables using simple criteria based on values. *e.g., to load customer data into two Customers' tables we could use the following:*

```
LOAD DATA
INFILE    'c:\Load\CustList2.txt'
BADFILE   'c:\Load\BadFile2.txt'
APPEND
INTO TABLE customer_pa
WHEN state = 'pa'
  (Customer# position(01:03), LastName position(13:25),
   FirstName position(26:36), Address  position(40:57),
   City      position(58:70), State    position(71:73) "UPPER(:STATE)",
   Zip       position(74:79) )
INTO TABLE customer_OH
WHEN state = 'oh'
  (Customer# position(01:03), LastName position(13:25),
   FirstName position(26:36), Address  position(40:57),
   City      position(58:70), State    position(71:73) "UPPER(:STATE)",
   Zip       position(74:79) )
```

## Integration – Part II

Focus on Publisher information

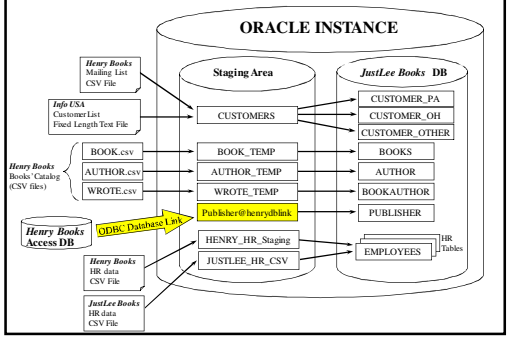
◈ Source – publisher information from *Henry Books*

◈ Format – a table in an MS Access database

◈ Target – PUBLISHER table in *JustLee Books*

◈ Method – we use Open Database Connectivity (ODBC) to access MS Access table with Publisher data. Then, we merge the MS Access table with PUBLISHER table in *JustLee Books* database.

## Part II: Publishers' Data

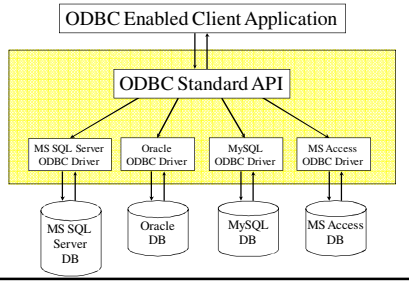



## Linking databases – ODBC

- For data stored in other DBMSs, MS Excel or any other ODBC accessible data source, we can use ODBC connectivity to access the data
- ODBC is a layer that exposes database functionality
- ODBC is a specification for a database API, it is a standard, it is not a tool in and of itself



## Where does ODBC reside?




## Linking databases – ODBC (cont.)

- With ODBC drivers and ODBC enabled applications, we can connect to and interact with multiple, disparate data sources
- Example of ODBC enabled data sources:
  - Oracle Database
  - MS SQL Server
  - MS Excel
  - MS Access
  - MySQL
  - etc.



## Accessing external data

- Oracle "Database Link" object
  - A database link is a pointer for a one-way communication path from an Oracle DB to an external DB
  - This allows external tables to appear as Oracle tables
  - The path is one way in the sense that it does not allow the external DB access to access the Oracle tables
  - Oracle is the **client** application accessing tables in an external MS Access DB
- *JustLee Books* will use an Oracle *Database Link* object to connect to *PUBLISHER* table in MS Access
  - The PUBLISHER table will appear as a read-only table in Oracle Instance

Source: http://download.oracle.com/docs/cd/B28359_01/server.111/b28310/ds_concepts002.htm

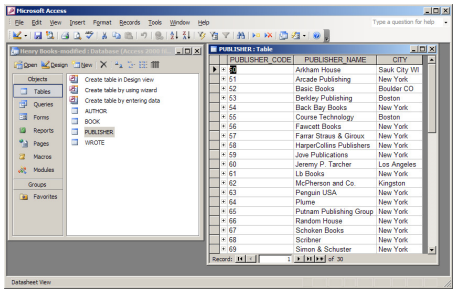
## Linking Oracle DB to Access DB

- Steps:
  1. Create ODBC DSN
  2. Edit *inithsodbc.ora*
  3. Edit *listener.ora*
  4. Edit *tnsnames.ora*
  5. Test new TNS entry
  6. Create database link in DB



Source: http://www.orafaq.com/node/60

## *Henry Books* Access DB




## Linking Oracle DB to Access DB (cont.)

- Create and test new Database Link
- Results are the same 30 rows, directly from the MS Access DB

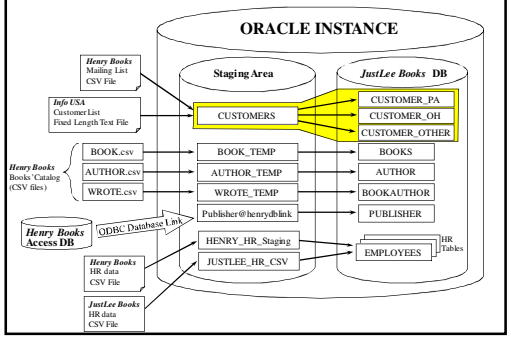
```
SQL> CREATE DATABASE LINK henryDBlink USING 'henryDB';
Database link created.

SQL> SELECT * FROM   publisher@henrydblink;
PUB PUBLISHER_NAME              CITY
--- --------------------------- --------------------
50  Arkham House                Sauk City WI
51  Arcade Publishing           New York
52  Basic Books                 Boulder CO
53  Berkley Publishing          Boston
54  Back Bay Books              New York
55  Course Technology           Boston
56  Fawcett Books               New York
57  Farrar Straus & Giroux      New York
58  HarperCollins Publishers    New York
59  Jove Publications           New York
60  Jeremy P. Tarcher           Los Angeles
61  Lb Books                    New York
62  McPherson and Co.           Kingston
63  Penguin USA                 New York
64  Plume                       New York
65  Putnam Publishing Group     New York
66  Random House                New York
67  Schoken Books               New York
68  Scribner                    New York
69  Simon & Schuster            New York
70  Scholastic Trade            New York
71  Taunton Press               Newtown CT
72  Tor Books                   New York
73  Thames and Hudson           New York
74  Touchstone Books            Westport CT
75  Vintage Books               New York
76  W.W. Norton                 New York
77  Westview Press              Boulder CO
78  Printing Is Us              Baltimore
79  American Publishing         Los Angeles
```

## Conclusion

- All steps to this point involved accessing data from unrelated independent sources and integrating that data with *JustLee Books* data – in staging area
- After importing data into the staging area, further data cleansing is needed
  - Validating domain values
  - Enforcing data integrity – PK/FK values
  - Avoiding duplicating data or losing existing data
  - Ensuring consistent data format
    - Rd vs. Road, Ave vs. Avenue, etc.
  - Ensuring that data is valid
    - Verifying that Zip Codes and City/States are valid
- Best to clean data while in the staging area rather than to load inconsistent data into final tables

## Working with Internal Data

## Part I: New Customers' Tables




## CUSTOMERS table

```
SQL> SELECT customer#, lastname, firstname,
  2         firstname, address, city, state, zip
  3  FROM   Customers;

CUSTOMER# LASTNAME   FIRSTNAME  ADDRESS              CITY          ST ZIP
--------- ---------- ---------- -------------------- ------------- -- -----
     1001 MORALES    BONITA     P.O. BOX 651         EASTPOINT     FL 32328
     1002 THOMPSON   RYAN       P.O. BOX 9835        SANTA MONICA  CA 90404
     1003 SMITH      LEILA      P.O. BOX 66          TALLAHASSEE   FL 32306
     1004 PIERSON    THOMAS     69821 SOUTH AVE      BOISE         ID 83707
     1005 GIRARD     CINDY      P.O. BOX 851         SEATTLE       WA 98115
     1006 CRUZ       MESHIA     82 DIRT ROAD         ALBANY        NY 12211
      . . .
      505 Bellows    Jim        7477 Elm St          Monroeville   PA 15146
      510 Hawk       Anna       23 Oak Ln            Pittsburgh    PA 15210
      545 Fisk       Ren        987 Count Dr         Cleveland     OH 44101
      . . .
      800 Rana       Flemming   13 46th St.          Millvale      PA 15209
      801 Thad       Wilkanson  11822 Rte. 19        Pittsburgh    PA 15221
      802 Olga       Saunders   441 Reynolds St      Pittsburgh    PA 15221
      803 Pierce     Rialto     67 West 17th Ave     Columbus      OH 43085
      . . .

40 rows selected.
```

*JustLee Books*

*Henry Books*

*InfoUSA*

## CUSTOMERS table – sorted

```
SQL> SELECT customer#, lastname, firstname,
  2         firstname, address, city, state, zip
  3  FROM   Customers;
  4  ORDER BY state;

CUSTOMER# LASTNAME   FIRSTNAME  ADDRESS            CITY          ST ZIP
--------- ---------- ---------- ------------------ ------------- -- -----
     1002 THOMPSON   RYAN       P.O. BOX 9835      SANTA MONICA  CA 90404
     1009 PEREZ      JORGE      P.O. BOX 8564      BURBANK       CA 91510
     1016 DAUM       MICHELL    9851231 LONG ROAD  BURBANK       CA 91508
     1001 MORALES    BONITA     P.O. BOX 651       EASTPOINT     FL 32328
     1003 SMITH      LEILA      P.O. BOX 66        TALLAHASSEE   FL 32306
     ...
      540 Gruno      Pina       767 Fly Way        Youngstown    OH 44501
      804 Sommers    O'Connell  88 Circle Dr       Columbus      OH 43085
      806 Floyd      Inud       25 Summer Rd       Cranberry     OH 16066
      807 Gretchen   Tylon      146 18th St.       Cleveland     OH 44102
      803 Pierce     Rialto     67 West 17th Ave   Columbus      OH 43085
      545 Fisk       Ren        987 Count Dr       Cleveland     OH 44101
      505 Bellows    Jim        7477 Elm St        Monroeville   PA 15146
      810 William    Drake      45 Degree Rd       Pittsburgh    PA 15217
      805 Drayton    Powell     601 North Ave      Wilkins       PA 15146
      802 Olga       Saunders   441 Reynolds St    Pittsburgh    PA 15221
      801 Thad       Wilkenson  11822 Rte. 19      Pittsburgh    PA 15221
     ...
40 rows selected.
```

"Other" (CA, FL) · OH · PA



## Multitable **INSERT** overview

- Multitable **INSERT** statements can be used to transfer data to one or more target tables
- In this case, we are moving records from the CUSTOMERS table in the staging area into 3 new geographically-based tables based on the customer's state value:
  - CUSTOMER_PA
  - CUSTOMER_OH
  - CUSTOMER_OTHER
- We can do this with a single SQL Insert statement



## Multitable **INSERT** (cont.)

- The generic syntax of a multitable **INSERT** statement:

```
INSERT [ALL] | [FIRST]
[conditional_insert_clause]
[into_clause  values_clause]
subquery

conditional_insert_clause:

[WHEN condition1 THEN] [into_clause values_clause]
[WHEN condition2 THEN] [into_clause values_clause]
[ELSE] [into_clause values_clause]
```



## Multitable **INSERT** (cont.)

- Unconditional mutitable **INSERT**:
  - Multiple **into_clause values_clause** followed by a subquery that defines the source of data
  - The Oracle server executes each **into_clause values_clause** once for each row retrieved by the subquery
- Conditional mutitable **INSERT**:
  - Includes **conditional_insert_clause(s)**
  - The Oracle server evaluates each **WHEN** clause to determine whether or not to execute each **into_clause values_clause**



## Conditional **INSERT ALL**

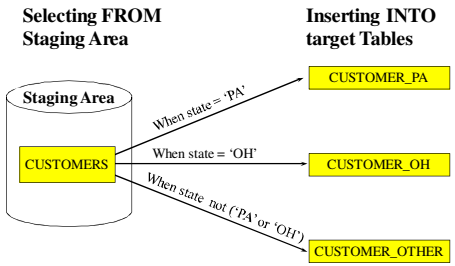
- If we specify **ALL**, the Oracle server evaluates each **WHEN** clause independently of the other **WHEN** clauses
- For each **WHEN** condition, if it evaluates to true, the Oracle server executes the corresponding **INTO** clause then continues to the following condition



## Conditional **INSERT FIRST**

- If we specify **FIRST**, the Oracle server evaluates each **WHEN** clause in the order in which it appears in the statement
  - For the first **WHEN** clause that evaluates to true, Oracle executes the corresponding **INTO** clause
  - Skips all subsequent **WHEN** clauses for the given row
  - if no **WHEN** clause evaluates to true then
    - If we specified an **ELSE** clause, Oracle executes the **INTO** clause list associated with the **ELSE** clause
    - If we did not specify an **ELSE** clause, the Oracle server takes no action for that row

## Customers data transfer




## Customers data transfer (cont.)

- We copy customer records from the temporary table into 3 base tables according to the customers' state:

```
SQL> INSERT FIRST
  2    WHEN state = 'PA'
  3    THEN INTO customer_pa
  4           (customer#, lastname, firstname, address, city, state, zip)
  5      VALUES (customer#, lastname, firstname, address, city, state, zip)
  6    --<>--
  7    WHEN state = 'OH'
  8    THEN INTO customer_oh
  9           (customer#, lastname, firstname, address, city, state, zip)
 10      VALUES (customer#, lastname, firstname, address, city, state, zip)
 11    --<>--
 12    ELSE INTO customer_other
 13           (customer#, lastname, firstname, address, city, state, zip)
 14      VALUES (customer#, lastname, firstname, address, city, state, zip)
 15    --<>--
 16  SELECT *
 17  FROM customers;

20 rows created.
```

## Result of the INSERT

- Rows in CUSTOMER_PA:

```
SQL> SELECT *
  2  FROM   Customer_PA;

CUSTOMER# LASTNAME   FIRSTNAME ADDRESS              CITY         ST ZIP
--------- ---------- --------- -------------------- ------------ -- -----
      505 Bellows    Jim       7477 Elm St          Monroeville  PA 15146
      510 Hawk       Anne      23 Oak Ln            Pittsburgh   PA 15210
      535 Wald       Beatrice  45 Green Way         Oakland      PA 15213
      800 Rena       Flemming  13 46th St.          Millvale     PA 15209
      801 Thad       Wilkenson 11822 Rte. 19        Pittsburgh   PA 15221
      802 Olga       Saunders  441 Reynolds St      Pittsburgh   PA 15221
      805 Drayton    Powell    601 North Ave        Wilkins      PA 15146
      810 William    Drake     45 Degree Rd         Pittsburgh   PA 15217

8 rows selected.
```



## Result of the INSERT (cont.)

- Rows in CUSTOMER_OH:

```
SQL> SELECT *
  2  FROM   Customer_OH;

CUSTOMER# LASTNAME   FIRSTNAME ADDRESS              CITY         ST ZIP
--------- ---------- --------- -------------------- ------------ -- -----
      540 Gruno      Pina      767 Fly Way          Youngstown   OH 44501
      545 Fisk       Ren       987 Count Dr         Cleveland    OH 44101
      803 Pierce     Rialto    67 West 17th Ave     Columbus     OH 43085
      804 Sommers    O'Connell 88 Circle Dr         Columbus     OH 43085
      806 Floyd      Inud      25 Summer Rd         Cranberry    OH 16066
      807 Gretchen   Tylon     146 18th St.         Cleveland    OH 44102

6 rows selected.
```



## Result of the INSERT (cont.)

- Rows in CUSTOMER_OTHER:

```
SQL> SELECT *
  2  FROM   Customer_OTHER;

CUSTOMER# LASTNAME   FIRSTNAME ADDRESS              CITY         ST ZIP
--------- ---------- --------- -------------------- ------------ -- -----
      515 Ellis      Jen       767 View Dr          Rockville    MD 20847
      520 Bruny      Phil      3766 Blue Way        Arlington    VA 22201
      525 Downing    Juanita   11 Redbird Rd        Alexandria   VA 22301
      530 Lowe       Prada     1 Fawn Cricle        Wheeling     WV 26003
      808 Florence   Eldridge  8585 Eigth Ave       Detroit      MI 48202
      809 Wendy      Gundle    90 Jump St           Indianapolis IN 46217

6 rows selected.
```

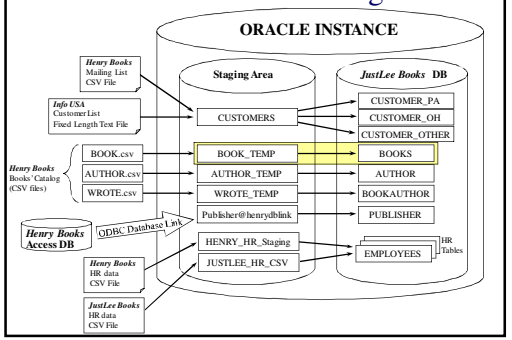


## Data Transfer – Part II

Focus on Book Catalog

- Source – "External Table" with Book inventory information from *Henry Books*
- Target – BOOKS table in *JustLee Books*
- Method:
  - Access the BOOK_TEMP external table to get additional book data
  - Use conditional insert to append new BOOK records into *JustLeeBooks'* BOOKS table
  - Additional processing of records based on *Henry Books* TYPE attribute: for HOR books, set *retail price = price + 10%;* for FIC books, set *retail price = price + 15%;* and for all other books set *retail price = price + 7%*

# Part II: Book Catalog




# BOOKS Table

```
SQL> SELECT isbn, title, pubdate, pubid, cost, retail, category
  2  FROM   books;

ISBN       TITLE                                    PUBDATE     PUBID    COST   RETAIL CATEGORY
---------- ---------------------------------------- --------- ------ ------- -------- ---------------
1059831198 BODYBUILD IN 10 MINUTES A DAY            21-JAN-01      4   18.75    30.95 FITNESS
0401140733 REVENGE OF MICKEY                        14-DEC-01      1    14.2       22 FAMILY LIFE
4981341710 BUILDING A CAR WITH TOOTHPICKS           18-MAR-02      2    37.8    59.95 CHILDREN
8843172113 DATABASE IMPLEMENTATION                  04-JUN-99      3    31.4    55.95 COMPUTER
3437212490 COOKING WITH MUSHROOMS                   28-FEB-00      4    12.5    19.95 COOKING
3957136468 HOLY GRAIL OF ORACLE                     31-DEC-01      3   47.25    75.95 COMPUTER
1915762492 HANDCRANKED COMPUTERS                    21-JAN-01      3    21.8       25 COMPUTER
9959789321 E-BUSINESS THE EASY WAY                  01-MAR-02      2    37.9     54.5 COMPUTER
2491748320 PAINLESS CHILD-REARING                   17-JUL-00      5      48    89.95 FAMILY LIFE
0299282519 THE WOK WAY TO COOK                      11-SEP-00      4      19    28.75 COOKING
8117949391 BIG BEAR AND LITTLE DOVE                 08-NOV-01      5    5.32     8.95 CHILDREN
0132149871 HOW TO GET FASTER PIZZA                  11-NOV-02      4   17.85    29.95 SELF HELP
9247381001 HOW TO MANAGE THE MANAGER                09-MAY-99      1    15.4    31.95 BUSINESS
2147428890 SHORTEST POEMS                           01-MAY-01      5   21.85    39.95 LITERATURE

14 rows selected.
```

# Conditional INSERT overview

- The Conditional **INSERT** statements provide advantages over separate **INSERT** statements:
  - Multiple independent single table **INSERT ... SELECT** statements process the same source data multiple times and increase the processing workload.
  - Using the **FIRST** clause ensures that each record is only inserted through one **INSERT** statement
  - Using **ALL** clause allows one row to be inserted multiple times in different ways



# Conditional INSERT overview (cont.)

- Inserts with different actions dependent on certain defined criteria
- Allows for distinct INSERT clauses for each criteria match
- Possible to insert same source row multiple times based with different INSERT clauses
- Similar to multi-table insert, but can be used for complex inserts into a single table as well

# Conditional INSERT

```
INSERT FIRST
WHEN type = 'HOR' THEN
  INTO books (ISBN, title, pubid, cost, retail, category)
  values
  (book_code, title, publisher_code, price, price * 1.10, type)
WHEN type = 'FIC' THEN
  INTO books (ISBN, title, pubid, cost, retail, category)
  values
  (book_code, title, publisher_code, price, price * 1.15, type)
ELSE
  INTO books (ISBN, title, pubid, cost, retail, category)
  values
  (book_code, title, publisher_code, price, price * 1.07, type)
SELECT * from BOOK_TEMP;
```

# BOOKS Table

```
SQL> SELECT isbn, title, pubdate, pubid, cost, retail, category
  2  FROM   books;

ISBN       TITLE                                    PUBDATE     PUBID    COST   RETAIL CATEGORY
---------- ---------------------------------------- --------- ------ ------- -------- ---------------
1059831198 BODYBUILD IN 10 MINUTES A DAY            21-JAN-01      4   18.75    30.95 FITNESS
0401140733 REVENGE OF MICKEY                        14-DEC-01      1    14.2       22 FAMILY LIFE
4981341710 BUILDING A CAR WITH TOOTHPICKS           18-MAR-02      2    37.8    59.95 CHILDREN
8843172113 DATABASE IMPLEMENTATION                  04-JUN-99      3    31.4    55.95 COMPUTER
3437212490 COOKING WITH MUSHROOMS                   28-FEB-00      4    12.5    19.95 COOKING
3957136468 HOLY GRAIL OF ORACLE                     31-DEC-01      3   47.25    75.95 COMPUTER
1915762492 HANDCRANKED COMPUTERS                    21-JAN-01      3    21.8       25 COMPUTER
9959789321 E-BUSINESS THE EASY WAY                  01-MAR-02      2    37.9     54.5 COMPUTER
2491748320 PAINLESS CHILD-REARING                   17-JUL-00      5      48    89.95 FAMILY LIFE
0299282519 THE WOK WAY TO COOK                      11-SEP-00      4      19    28.75 COOKING
8117949391 BIG BEAR AND LITTLE DOVE                 08-NOV-01      5    5.32     8.95 CHILDREN
0132149871 HOW TO GET FASTER PIZZA                  11-NOV-02      4   17.85    29.95 SELF HELP
9247381001 HOW TO MANAGE THE MANAGER                09-MAY-99      1    15.4    31.95 BUSINESS
2147428890 SHORTEST POEMS                           01-MAY-01      5   21.85    39.95 LITERATURE
0189       Magic Terror                                        56    7.99     8.79 HOR
1351       Dreamcatcher: A Novel                               68    19.6    21.56 HOR
0180       A Deepness in the Sky                               72    7.19     7.69 SFI
0200       The Stranger                                        75       8     8.56 FIC
0378       Venice                                              69    24.5    26.22 ART
079X       Second Wind                                         65   24.95     26.7 MYS
0808       The Edge                                            59    6.99     7.48 MYS
1382       Treasure Chests                                     71   24.46    26.17 ART
138X       Beloved                                             64   12.95    13.86 FIC

23 rows selected.
```

## Restrictions on Conditional INSERT

- Can be performed only on tables – not on views, materialized views, or remote tables
- The SELECT subquery at the end of the insert statement cannot use a sequence
- All **`into_clauses`** combined cannot specify more than 999 target columns



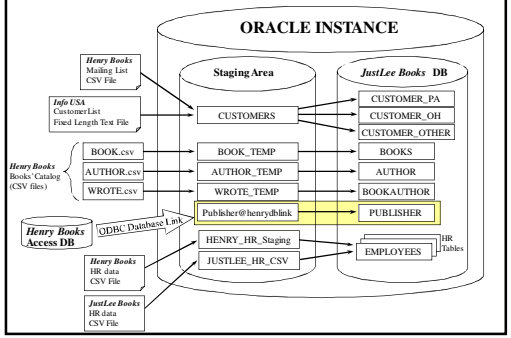
## Data Transfer – Part III

Focus on Publisher data from *Henry Books*

- Source – ODBC Link to *Henry Books'* Publisher data
- Target – *JustLee Book's* Publisher table
- Method:
  - Accessing data stored in MS Access database
  - Resolving overlapping data already in *JustLee Books's* Publisher table



## Part III: *Publisher* Data




## Data Transfer Issues/Process

- Some data from *Henry Books* corresponds to existing publishers' information in *JustLee Books* database
  - These records need to be updated with new attributes
- Remaining publisher records from *Henry Books* must be appended into *JustLee Books* Publisher table
- The Publisher data are combined using a single Merge statement



## Merging data in Oracle

- Existing overlap:

```
SQL> SELECT *
  2  FROM   publisher@henrydblink
  3  WHERE  LOWER(publisher_name) IN
  4         (SELECT LOWER(name) FROM publisher);

PUB PUBLISHER_NAME           CITY
--- ------------------------ --------------------
78  Printing Is Us           Baltimore
79  American Publishing      Los Angeles
```

- *Henry Book's* DB has an additional column (CITY) that we want to capture for the existing records
- We start by adding a new column into the *JustLee Books* table:

```
ALTER TABLE publisher ADD city VARCHAR2(50);
```

- Then we merge the publisher records from *Henry Books*

## Merging information

Merging ODBC linked *Henry Books* Publisher's table with *JustLee Books'* Publisher table in a single step:

```
MERGE INTO  publisher j
USING publisher@henrydblink h
  ON (LOWER(j.name) = LOWER(h.publisher_name))
  WHEN MATCHED THEN
     UPDATE
     SET j.city = LOWER(h.city)
  WHEN NOT MATCHED THEN
     INSERT (pubid, name, city)
     VALUES (h.publisher_code, h.publisher_name, h.city);
```

## Merging information

**`MERGE INTO publisher j`**: the **`publisher`** table (in the *JustLee Books* DB) is to be the result table (alias **`j`**)

**`USING publisher@henryDBlink h`**: the ODBC linked table **`publisher`** in the *Henry Books* Access DB provides the data to update and/or to insert into the *JustLee Books* **`publisher table`** (alias **`h`**)

**`ON(Lower(j.name) = Lower(h.publisher_name))`**: the rows of the two tables will be matched based on the publisher name column in each DB. The LOWER function is used to avoid case-sensitivity issues

**`WHEN MATCHED THEN`**: if a match is found, execute the **`UPDATE`** action to modify the rows in *JustLee Books* with "new" CITY data from the rows in *Henry Books*.

**`WHEN NOT MATCHED THEN`**: if no match is found (i.e., a publisher exists in *Henry Books* that is not in *JustLee Books*), then perform the **`INSERT`** action

## Publisher table after Merge

```
SQL> SELECT * FROM publisher;

 PUBID NAME                      CONTACT          PHONE        CITY
------ ------------------------- ---------------- ------------ ----------
     1 PRINTING IS US            TOMMIE SEYMOUR   000-714-8321 Baltimore
     2 PUBLISH OUR WAY           JANE TOMLIN      010-410-0010
     3 AMERICAN PUBLISHING       DAVID DAVIDSON   800-555-1211 Los Angeles
     4 READING MATERIALS INC.    RENEE SMITH      800-555-9743
     5 REED-N-RITE               SEBASTIAN JONES  800-555-8284
    50 Arkham House                                           Sauk City WI
    51 Arcade Publishing                                      New York
    52 Basic Books                                            Boulder CO
       . . .                    . . .            . . .        . . .
    73 Thames and Hudson                                      New York
    74 Touchstone Books                                       Westport CT
    75 Vintage Books                                          New York
    76 W.W. Norton                                            New York
    77 Westview Press                                         Boulder CO

33 rows selected.
```

**Note:** As the *Henry Books'* book catalog is merged with *JustlLee Books* database, we must be aware of the new publisher IDs for publishers that already existed in *JustlLee Books*

## Data transfer without Merge

- We can do this in multiple steps:

1. Extend the *JustLee's* PUBLISHER table by adding CITY column
2. Update the matching records with the new CITY information
3. Add the remaining new records into the Publisher table in *JustLee* database



## Data transfer without Merge (cont.)

1. Add a CITY column into Publisher table

```
ALTER TABLE publisher ADD city VARCHAR2(50);
```

2. Update existing records that exist in both data sets

```
UPDATE publisher j
SET city = (SELECT h.city
            FROM   publisher@henrydblink h
            WHERE  Lower(h.publisher_name) = Lower(j.name) )
WHERE  Lower(name) IN (SELECT Lower(publisher_name)
                       FROM   publisher@henrydblink)
```

3. Add new records from *Henry Books*

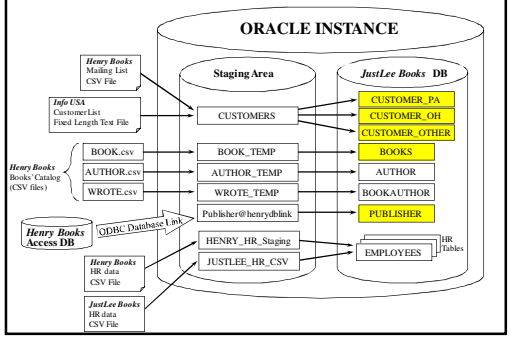
```
INSERT INTO publisher (pubid, name, city)
SELECT publisher_code, publisher_name, city
FROM   publisher@henrydblink
WHERE  Lower(publisher_name) NOT IN (SELECT Lower(name)
                                     FROM   publisher);
```

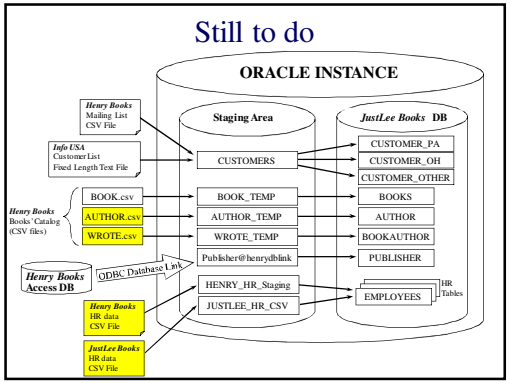
## Merge Delete

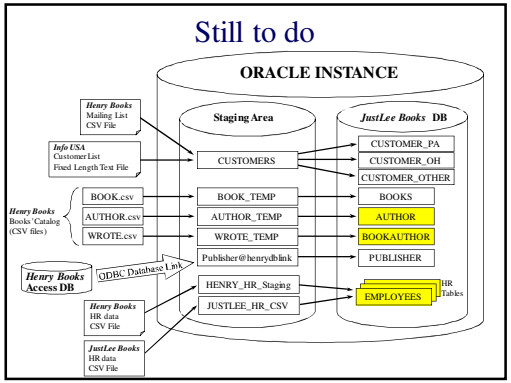
- Additional capability of the MERGE statement allows for deletion of records in target table
- Allows for completing multiple actions within a single command

```
MERGE INTO  target_table t
      USING source_table s
         ON (t.primary_key = s.primary_key)
WHEN MATCHED THEN
     UPDATE SET target.field1 = source.field1,
                target.field2 = source.field2
   DELETE  WHERE target.field3 = 'value';
```

## Done so far




Still to do
ORACLE INSTANCE
Staging Area
JustLee Books DB
Henry Books Mailing List CSV File
Info USA Customer List Fixed Length Text File
Henry Books Books 'Catalog (CSV files)
BOOK.csv
AUTHOR.csv
WROTE.csv
CUSTOMERS
BOOK_TEMP
AUTHOR_TEMP
WROTE_TEMP
Publisher@henrydblink
HENRY_HR_Staging
JUSTLEE_HR_CSV
CUSTOMER_PA
CUSTOMER_OH
CUSTOMER_OTHER
BOOKS
AUTHOR
BOOKAUTHOR
PUBLISHER
EMPLOYEES
HR Tables
Henry Books Access DB
ODBC Database Link
Henry Books HR data CSV File
JustLee Books HR data CSV File


Still to do
ORACLE INSTANCE
Staging Area
JustLee Books DB
Henry Books Mailing List CSV File
Info USA Customer List Fixed Length Text File
Henry Books Books 'Catalog (CSV files)
BOOK.csv
AUTHOR.csv
WROTE.csv
CUSTOMERS
BOOK_TEMP
AUTHOR_TEMP
WROTE_TEMP
Publisher@henrydblink
HENRY_HR_Staging
JUSTLEE_HR_CSV
CUSTOMER_PA
CUSTOMER_OH
CUSTOMER_OTHER
BOOKS
AUTHOR
BOOKAUTHOR
PUBLISHER
EMPLOYEES
HR Tables
Henry Books Access DB
ODBC Database Link
Henry Books HR data CSV File
JustLee Books HR data CSV File