

# SQL SSIS Data Loading: Customizing and Getting Under the Hood

**Michael Miars**

**AAC Inc.**

**With Introduction by**

**Eric Hunter**

**Bradford & Barthel, LLP**

# Problem Overview

- **Load timekeeper budget hours**
  - Some reports may reference this data
- **Start with a spreadsheet of annual hours per timekeeper to be loaded**
  - Simple format, hours will need to be split into each month
- **Set this up once and be able to use it year after year with little change**
  - Need a reusable process that can use parameters
- **Handle errors**
  - Save problem rows so they can be reviewed and corrected

## Spreadsheet Source

- A list of employee codes and names, along with the budgeted billable hours for the current year

	A	B	C
1	EMPLOYEE_CODE	EMPLOYEE_NAME	Billable Hours 01
2	02551	BROWN, DEAN S.	1550
3	02752	EDMONDS, ARTHUR	1750
4	02403	KARATAS, AUSTIN N II	1400
5	02754	WENTWORTH, WILLIAM F	1750
6	02555	BREWSTER, STEPHANIE	1550
7	02506	ZAPPELLI, KEVIN C.	1500
8	02607	O'SHEA, ANDREW	1600
9	02808	SAMUELSON, PETER	1800
10	02609	BERGHOFF, MRS. LUCY	1600
11	02560	ZAMPELLI, AMY	1550
12	02636	BARROWS, POLLY	1625
13	02542	LAMOTHE, JULIE	1600

## Budget table

- Customized database table created for the purpose of using these numbers within reports
- Data is period-based, so for each row in the spreadsheet, twelve rows will need to be created in this table, each with a unique row\_id
- The empl\_id will need to be looked up

```
create table TIMEBUD_BL
(
    ROW_ID          int
    ,EMPL_ID        int
    ,PERIOD          int
    ,BL_HRS          money
)
```

## Sidetrack: SQL-based solution

- Load the spreadsheet data as is into the database
- Use purely SQL to break up the data into periods; assign row\_id; and insert
- Non-trivial SQL

```

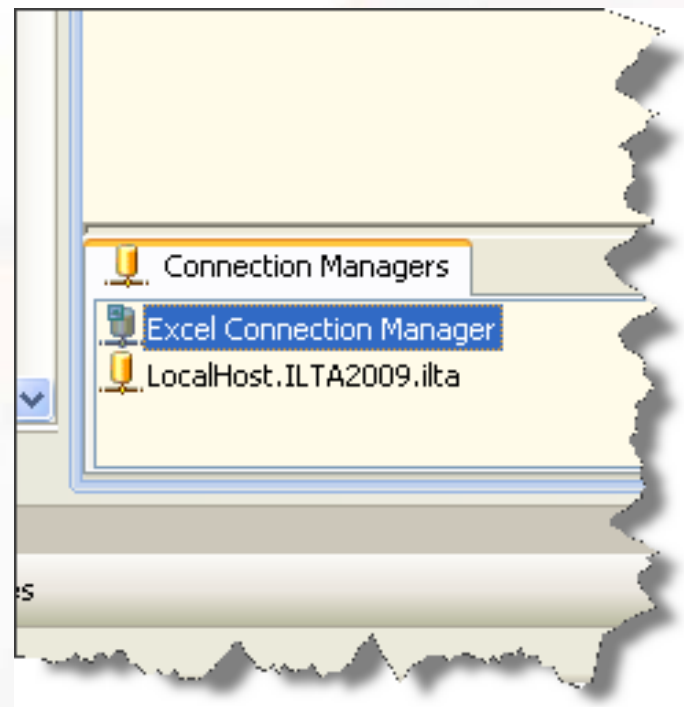
alter table budgets add
add _seq_num int identity(1,1)
    ,_seq_num_12 int

update budgets
set _seq_num_12 = _seq_num * 12

insert into timebud_b1
select _seq_num + isnull((select max(row_id) from timebud_b1), 0) + c.period % 100 - 1
    ,b.empl_id
    ,c.period
    ,round([Billable Hours] / 12, 1)
from budgets a
LEFT OUTER JOIN Employees b ON a.employee_code = b.employee_code
LEFT OUTER JOIN period_ref c ON c.year = 2009
  
```

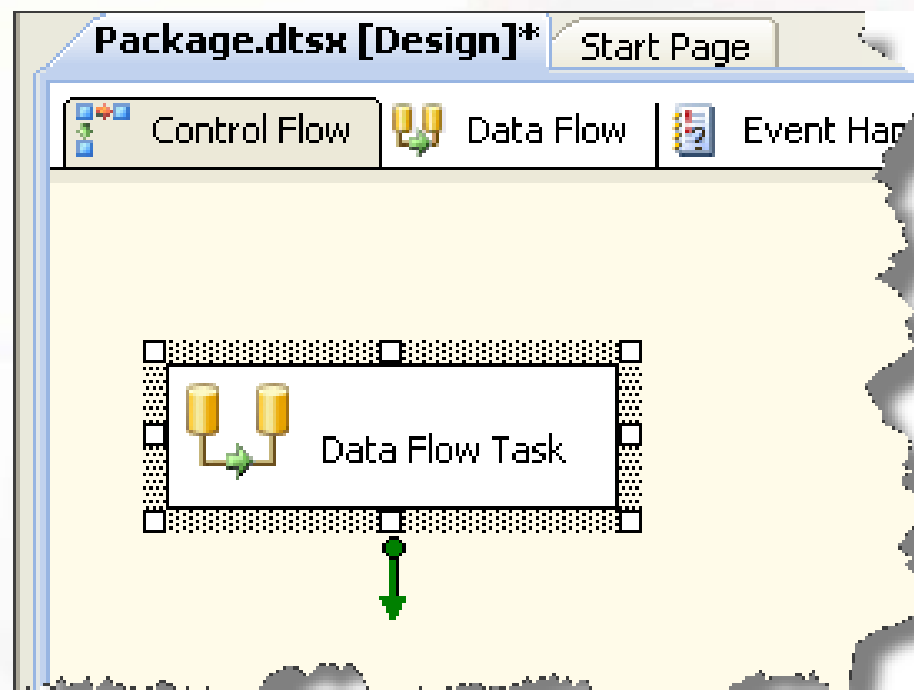
## Connection Managers

- Define these first
- Specifies the file locations or basic connection information
- Does not specify a particular table, worksheet, etc.



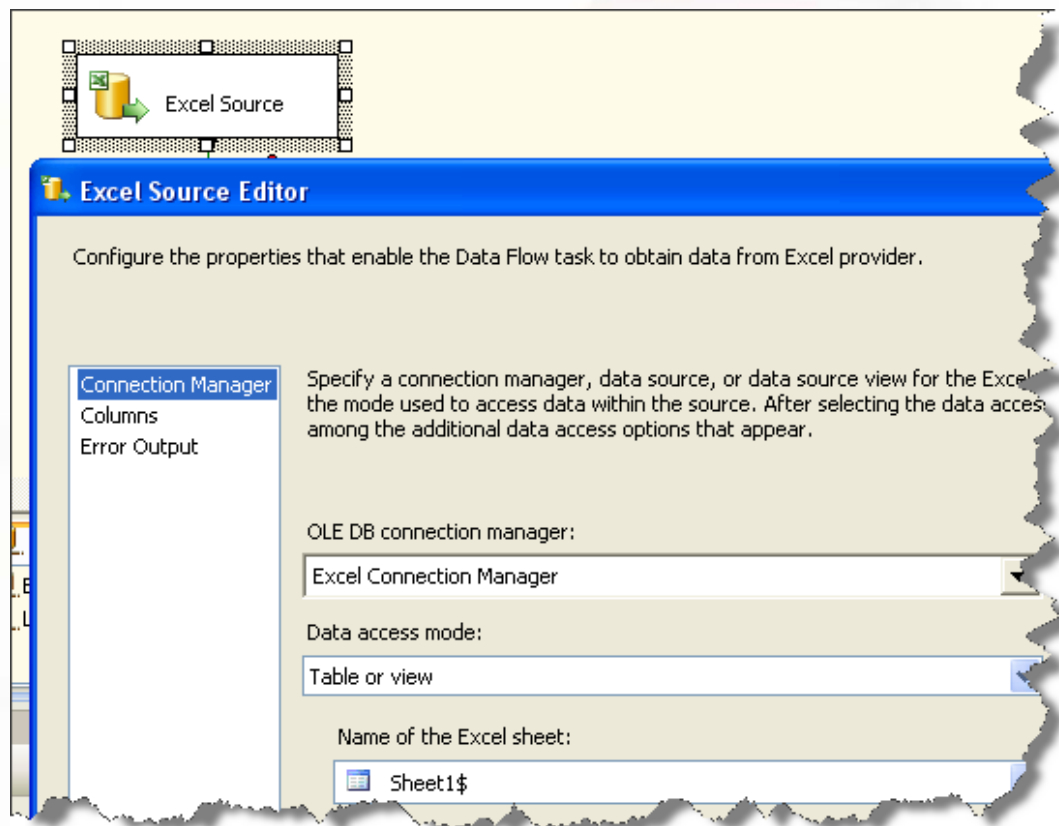
## Data Flow Task

- Everything for this example will reside within a single Data Flow Task
- A Data Flow Task is created by dragging it from the Toolbox into the Control Flow design area of the screen
- Double-clicking the Data Flow Task will open it and change the view to the Data Flow design area



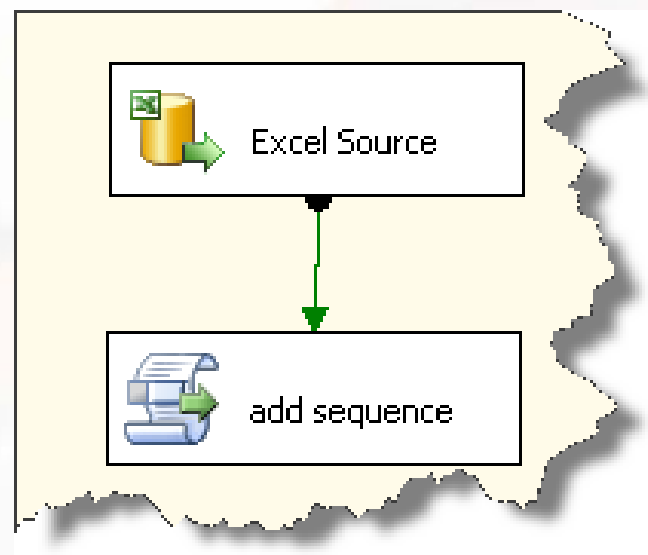
## Excel Source – Read the Rows

- Within the Data Flow Task, start by creating an Excel Source
- Use the Excel Connection Manager defined earlier
- Pick the worksheet to use
- Specify the columns to use from the worksheet



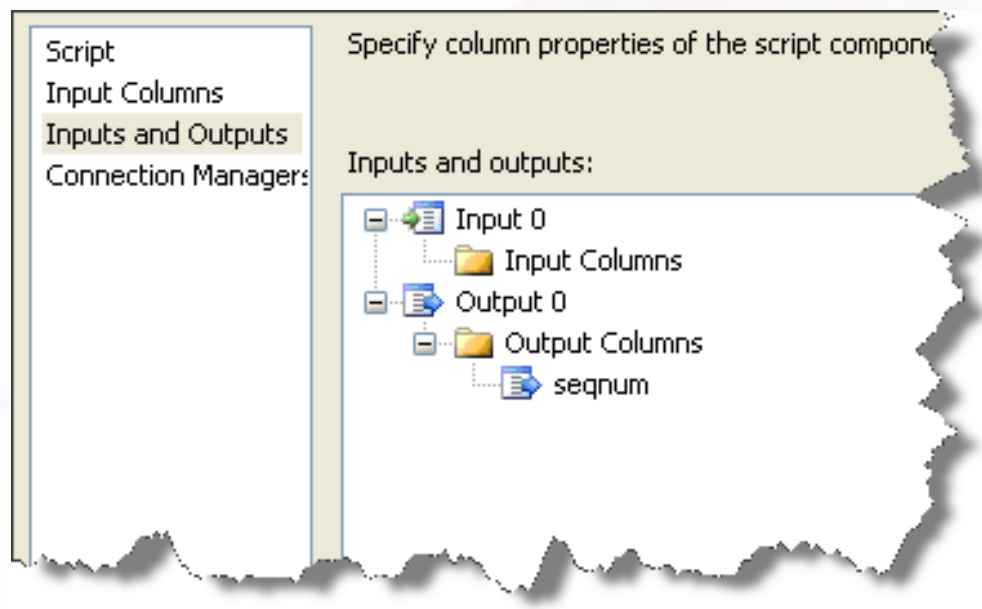
## Add Sequence Number

- SSIS does not have a built-in component for adding a unique number to each row in a data flow (analogous to the identity() function in T-SQL)
- Sequence numbers can be added by creating a Script Component



## Add Sequence Number (cont'd)

- Within the new script component, define the new output columns, as well as any variables from the package that will need to be visible from within the script (either Read-Only or Read-Write variables)
- In this case, there is one new output column called “seqnum”



## Add Sequence Number (cont'd)

- The script itself can be either Visual Basic or C# (new feature of SSIS 2008)
- Simple coding now, initialize a variable and increment it on each passing row

```
Dim CurrentKey As Integer
```

```
Public Overrides Sub PreExecute()  
    MyBase.PreExecute()
```

```
    CurrentKey = -1
```

```
End Sub
```

```
Public Overrides Sub PostExecute ...
```

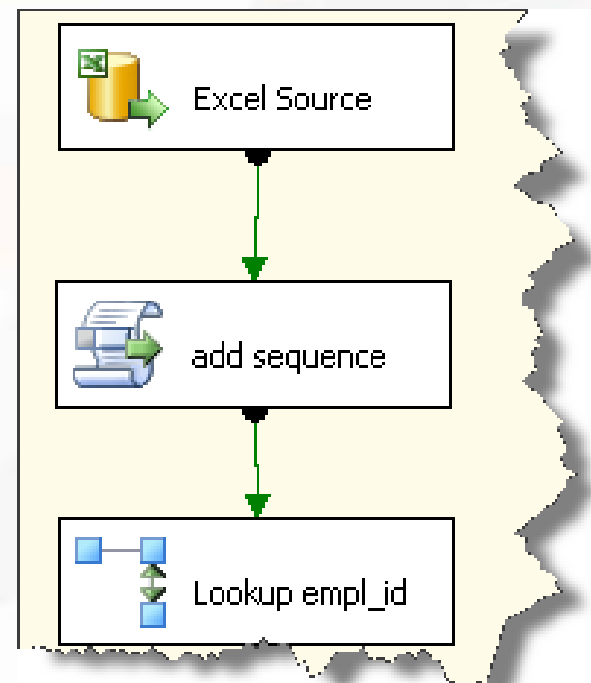
```
Public Overrides Sub Input0_ProcessInputRow(ByVal Row As Input0Buffer)  
    CurrentKey += 1
```

```
    Row.seqnum = CurrentKey
```

```
End Sub
```

## Lookup the empl\_id

- Drop in a lookup component to the Data Flow
- This will add a new column to the data flow stream



## Lookup the empl\_id (cont'd)

- Within the lookup component, the data source is specified
- The employee\_code field from the data stream is linked visually to the employee\_code field in the database's Employee table
- The resulting lookup field, [EMPL\_ID] is added to the data flow stream

The screenshot displays the configuration of a lookup component in Oracle Data Integrator. The 'Columns' tab is active, showing two panels: 'Available Input Columns' and 'Available Lookup Columns'. A line connects the 'EMPLOYEE\_CODE' field in the input columns to the 'EMPL\_ID' field in the lookup columns. Below these panels, a table summarizes the lookup operation.

Lookup Column	Lookup Operation	Output Alias
EMPL_ID	<add as new column>	EMPL_ID

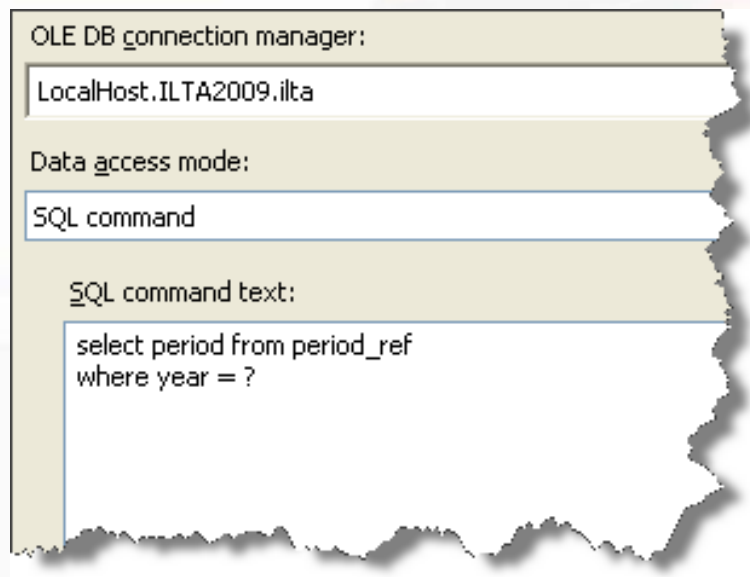
## Assign Twelve Unique Sequence Numbers

- Drop in a Derived Column component
- Use this to create a variable that increases in increments of 12 by multiplying the original seqnum variable by 12.
- The [cross] field will be explained later, note that it is hardcoded to 1

Derived Column Name	Derived Column	Expression	Data Type	Le
seqnum_12	<add as new column>	seqnum * 12	four-byte signed int...	
cross	<add as new column>	1	four-byte signed int...	

## Lookup Periods for the Year

- A table was created in SQL called `period_ref` that contains years and periods.
- A new OLE DB data source is created that selects all the periods from `period_ref` where the year is 2009; it will return 12 rows
- This will be used to cross-join with the original data flow to create 12 rows per single row from the spreadsheet



OLE DB connection manager:

LocalHost.ILTA2009.ilta

Data access mode:

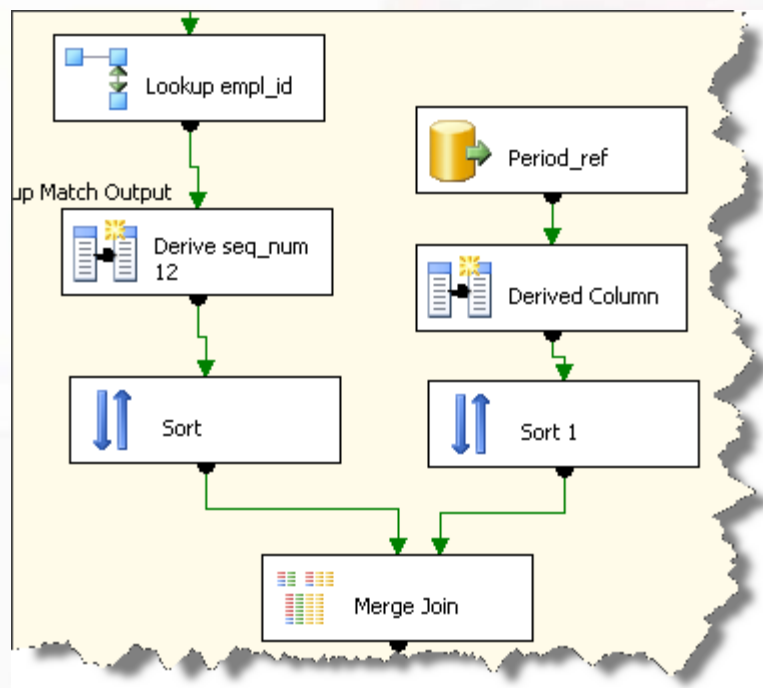
SQL command

SQL command text:

```
select period from period_ref  
where year = ?
```

## Perform a Cross Join

- SSIS does not natively support cross-join functionality
- It can be tricked into doing this by adding a constant-value field to two data streams, sorting on that field first, then joining on it; this was the [cross] field
- The Merge Join component is used to join the data streams, which must first be sorted. The sort was performed on the [cross] field that was added to each stream within the Derived Column components earlier



## Perform a Cross Join (cont'd)

- The [cross] fields are joined visually
- Any fields from the two inputs that will go to the output need to be selected here

**Merge Join Transformation Editor**

Configure the properties used to join two sources of sorted data. Select the join type and then specify the columns to be used as the join key. Join keys must be used in the order specified by the sort-key position of the column.

Join type:

Sort	Name	Order	
<input type="checkbox"/>	016 Practice Development	0	
<input type="checkbox"/>	018 Management	0	
<input type="checkbox"/>	seqnum	0	
<input checked="" type="checkbox"/>	EMPL_ID	0	
<input checked="" type="checkbox"/>	seqnum_12	0	
<input type="checkbox"/>	cross	1	

Sort 1	Name	Order	Join Key
<input checked="" type="checkbox"/>	period	0	<input type="checkbox"/>
<input type="checkbox"/>	cross	1	<input checked="" type="checkbox"/>

Input	Input Column	Output Alias
Sort	EMPL_ID	EMPL_ID
Sort	seqnum_12	seqnum_12
Sort 1	period	period
Sort	Billable Hours	Billable Hours

## Calculate remaining fields

- Add a Derived Column component to the output of the Merge Join
- Divide the billable\_hrs field by 12; rounding to the tenths place
- Assign a unique row\_id to each row by adding the modulo-base-100 of the period to the [seqnum\_12] field that was incrementing by 12

Derived Column Name	Derived Column	Expression	Data Type	Le
ROW_ID	<add as new column>	seqnum_12 + period % 100	Four-byte signed int...	
bl_hrs	<add as new column>	ROUND([Billable Hours] / 12,1)	double-precision flo...	

## OLD DB Destination – Write Out Rows

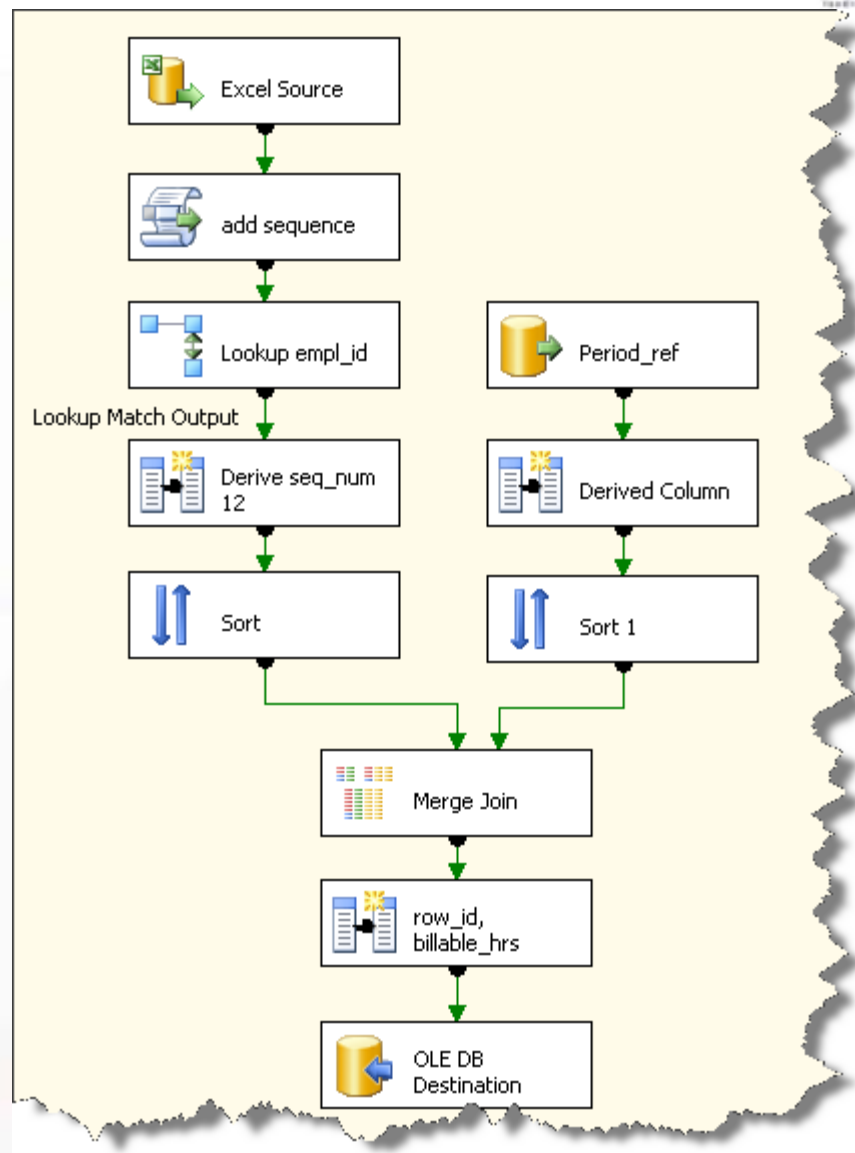
- Create an OLE DB Destination component and point it to the destination table, TIMEBUD\_BL
- Map the data flow fields to the table fields

The screenshot shows the 'Connection Manager' window with the 'Mappings' tab selected. It displays a mapping configuration for an OLE DB Destination. The 'Available Input Columns' list includes: EMPL\_ID, seqnum\_12, period, Billable Hours, ROW\_ID, and bl\_hrs. The 'Available Destination Columns' list includes: ROW\_ID, EMPL\_ID, PERIOD, and BL\_HRS. Lines connect the input columns to the destination columns: ROW\_ID to ROW\_ID, EMPL\_ID to EMPL\_ID, period to PERIOD, and bl\_hrs to BL\_HRS. The 'seqnum\_12' and 'Billable Hours' columns are not mapped.

Input Column	Destination Column
ROW_ID	ROW_ID
EMPL_ID	EMPL_ID
period	PERIOD
bl_hrs	BL_HRS

## Finished Product

- Each piece of the flow can be examined between components by adding a Data Viewer to capture live data
- Clear naming is helpful when trying to debug or when accessing particular components
- Additional streams can break off from this (e.g. the Lookup Unmatched Output) and perform cleanup, notification and logging of rows with an issue
- The Row Count component is a good “stub” to use while developing the data flow



# From here...

- Run the package from within the Business Intelligence Development Studio for each load, or...
- Save the package as a file and schedule it to run with a parameter for the year
- Add email notifications, error handling routines
- Use the same kind of logic for other data loading tasks