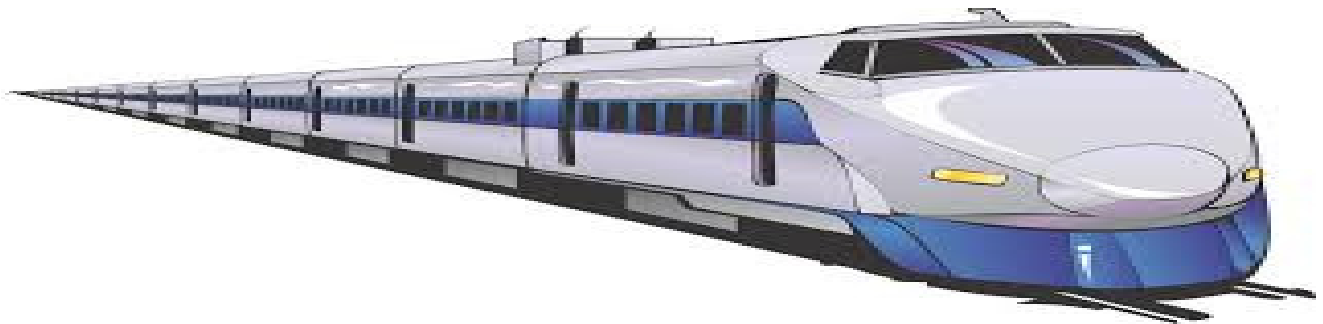


Climb Aboard The ProDataSet Train



Paul Guggenheim & Associates, Inc.

1788 2nd St., Ste 200
Highland Park, IL 60035

(847) 926-9800

www.pgasmarts.com

Climb Aboard The ProDataSet Train



Copyright © 2016

Paul Guggenheim & Associates, Inc.

The Climb Aboard The ProDataSet Train is copyrighted and all rights are reserved by Paul Guggenheim & Associates, Inc. This manual is copyrighted and all rights are reserved. This document may not, in whole or in part, be copied, photocopied, translated, or reduced to any electronic medium or machine-readable form without prior consent, in writing, from Paul Guggenheim & Associates, Inc.

Printed in U.S.A.

First Printing – June, 2016

All company and product names are the trademarks or registered trademarks of their respective companies. OpenEdge is a registered trademark of Progress Software Corporation.

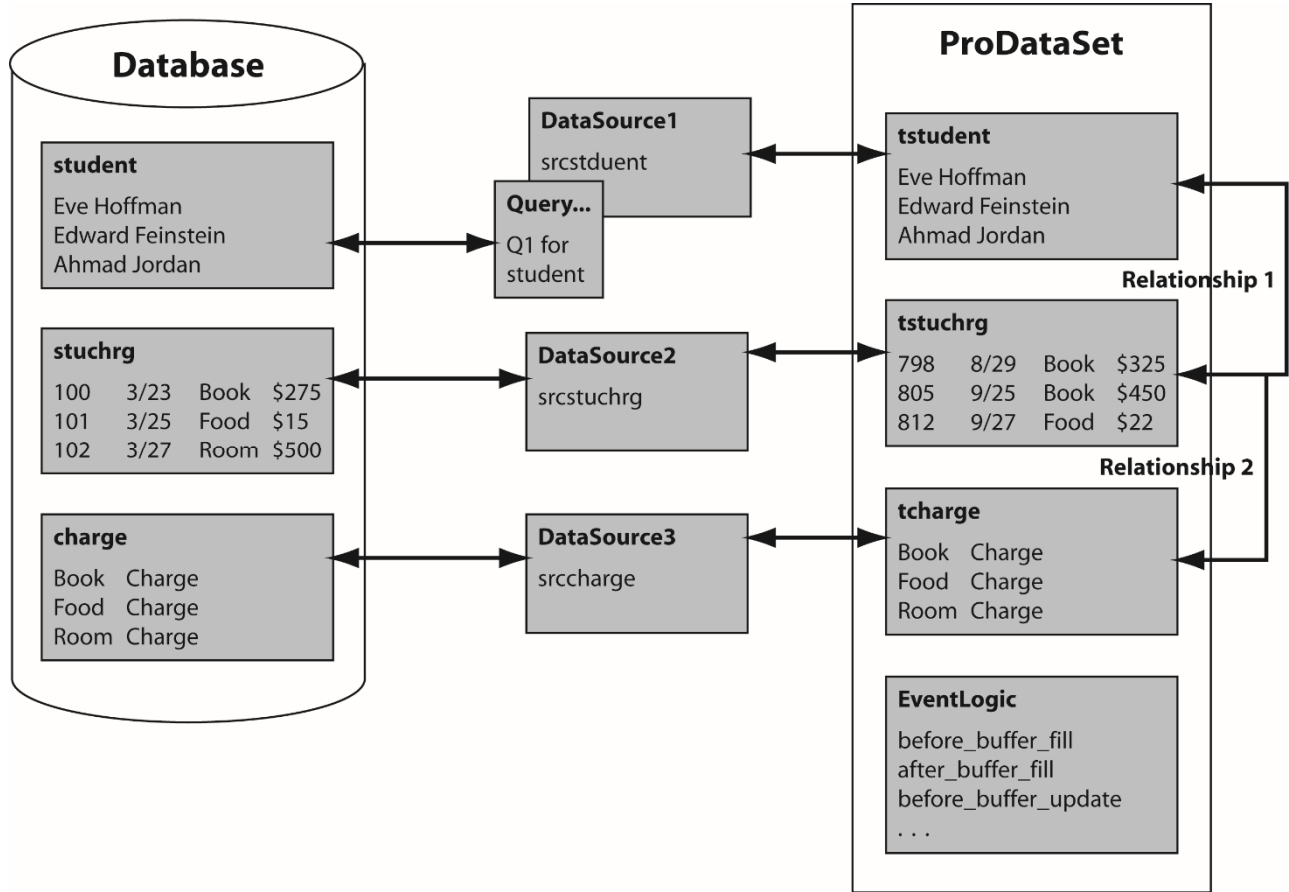


Notes





Introduction to ProDataSets





Introduction to ProDataSets

What is a ProDataSet?

- A collection of one or more temp-tables that optionally contain a collection of data-relations among the member temp-tables.
- Each member temp-table can attach to a data-source object in order to receive data from the database or writing data to the database.
- The term for a ProDataSet receiving data from the database is called filling.
- In addition, 4GL procedures may be assigned to the many ProDataSet named events in order to customize the behavior, perform validation, and execute different kinds of business logic.



Introduction to ProDataSets

What can ProDataSets do?

- Define complex business logic between many levels of related data
- Define a data mapping between the ProDataSet and the database
- Associate hooks to custom event procedures
- Records changes to ProDataSet temp-table records for updating the database
- Can pass the DataSet as a single parameter with a single handle from one procedure to another, within a single Progress session or between sessions.
- Can be either a static or dynamic object, consistent with other objects such as temp-tables

What are ProDataSet good for?

- Isolating business logic into one object
- Minimizing network traffic in distributed applications
- Use the ABL to develop applications using the .NET user interface
- Provide a convenient, powerful and consistent way to separate application data access from how the underlying data is stored in the database
- Capture creates, changes and deletions on either client or server and allow maximum flexible transaction control
- Provide a migration strategy for ADM2 SDOs and SBOs (SMARTDataObjects and SMARTBusinessObjects)



Introduction to ProDataSets

Typical uses for ProDataSets

1. A database disconnect user interface application such as WebClient or .NET.
 - For example, a client enters a purchase order which requests a ProDataSet from a server. The client receives the PO information from the server, using the data in the ProDataSet to display the data in browses or viewers. Data is updated on the client and a reduced ProDataSet containing only the changes is then passed back to the AppServer to minimize network traffic. On the AppServer the data is validated and updated against the database.
2. Processing server-side business logic.
 - A business logic procedure uses one or more ProDataSets to access the pertinent application data. ProDataSets encapsulate the data, insulating the logic from the actual structure of the data sources. This is an essential aspect of a future-proof application architecture.
3. Using ProDataSets to perform complex calculated data such as a price sheet.
4. Pass a ProDataSet containing a single temp-table **BY-REFERENCE**, since this option is not available for passing temp-tables by themselves.
5. Use a ProDataSet as a mechanism to pass a set of possibly unrelated lookup tables to be used by the client session.

Climb Aboard The ProDataSet Train



Notes





ProDataSet Syntax

- ProDataSets are defined using the DEFINE DATASET statement.
- There are 3 components to a ProDataSet:
 - Temp-Tables
 - Data-Relations
 - Data-Sources
- Temp-tables must be defined before the static ProDataSet is defined. The ProDataSet references the temp-table in the ProDataSet definition by referring to a buffer name of the temp-table. By default, the buffer name is the temp-table name.
- Data-Relations are optional, dependent components defined in the ProDataSet. Typically, Data-Relations are commonly used to define how a parent and child buffer are linked together in a ProDataSet. If only one buffer is defined for a ProDataSet, then a Data-Relation doesn't need to be defined.

Syntax

Define DATASET statement

```
DEFINE { [ [ NEW ] SHARED ] | [ PRIVATE | PROTECTED ] [ STATIC ] }  
  
DATASET dataset-name  
  
[ NAMESPACE-URI namespace ] [ NAMESPACE-PREFIX prefix ]  
  
[ XML-NODE-NAME node-name ] [ SERIALIZE-NAME serialize-name ]  
  
[ XML-NODE-TYPE node-type ] [ SERIALIZE-HIDDEN ]  
  
[ REFERENCE-ONLY ] FOR buffer-name [ , buffer-name ] ...  
  
[ DATA-RELATION [ data-rel-name ] FOR data-rel-spec ] ...  
  
[ PARENT-ID-RELATION [ data-rel-name ] FOR parent-id-rel-spec ] ...
```



ProDataSet Syntax

- A Data-Source is an independent object that is used to transfer data between the database and a ProDataSet.
- Typically, there is one Data-Source defined for each defined temp-table buffer in a ProDataSet. However, the Data-Source may represent more than one database buffer (table) if there are fields in the temp-table buffer that come from different database tables.
- The Data-Source definition identifies the unique key in the database table for the Data-Source. How it maps to the temp-table buffer of the ProDataSet is defined in the attach-data-source method (more on that later).
- The Data-Source definition may specify either a query phrase or a source-buffer-phrase or both.
 - Typically, a query is defined for the top-level ProDataSet buffer. The purpose is to only read a subset of the parent records when the ProDataSet is filled.
 - Typically, a source-buffer-phrase is used for a child ProDataSet buffer. This populates the child buffer with related records from the parent when the ProDataSet is filled.
- The reason the data-source is a separate object gives the developer flexibility in passing the ProDataSet to another session. Data-sources are connected with database buffers which can't be passed across a session boundary. Another advantage is it allows a ProDataSet to attach and detach data-sources during program execution which makes it easy to load data into a ProDataSet from different databases.
- Even though you can't pass data-sources across the session boundary, and cannot pass a data-source as a parameter, the data-source handle may be accessed through its DataSet if the DataSet is passed locally. Or another option is to simply include the same data-source definition in multiple procedures.



ProDataSet Syntax

- In general, a query is not appropriate or necessary to be defined for a data-source when:
 - The data-source is a child table in a data-relation of a ProDataSet. The data-source will retrieve the records based on the relationship with the parent record during a FILL method.
 - There are exceptions where a query is desired on a child data source. This is covered in the Advanced Reading Operations section.
 - The data-source is a top level table (one with no parent) and retrieving all the records from the data-source is desirable.
 - The data-source is used for update purposes only. Remember, a defined query for a data-source is only used on a FILL.
 - A query will be attached to the data-source dynamically at run-time.
- In general, a data-source may not be needed at all when an event procedure is used to fill or update the table. For example, a flat file of data may be used to load data into a temp-table during the execution of an event procedure.



Populating a ProDataSet

pds1.p

```
/* pds1.p dataset example one - one table read-only dataset */  
  
.  
.  
.  
  
define temp-table tstudent no-undo like student.  
  
define dataset dsstudent for tstudent.  
  
define query qstudent for student.  
  
define data-source srcstudent  
  for query qstudent  
  student keys (studentid).  
  
.  
.  
.  
  
buffer tstudent:attach-data-source(data-source srcstudent:handle).  
  
query qstudent:query-prepare("for each student NO-LOCK  
where stcode = 'IL' and syear = 2007").  
  
dataset dsstudent:fill().  
  
buffer tstudent:detach-data-source().
```

StudentID	First Name	Last Name	GPA	Phone
000206	Derwood	Glass	2.75	(312) 804-7418
001956	Diane	Huber	3.75	(312) 519-8147
002037	Gladys	Larson	2.83	(312) 534-0669
002819	Quincy	Jacobson	2.80	(312) 765-0009



StudentID: 000206
Name: Derwood Glass
City,St,Zip: Chicago IL 60639
Phone: (312) 804-7418

First Next Prev Last Full Done



Populating a ProDataSet

- The first example, **pds1.p**, the ProDataSet consists of only one table, and therefore there are no data-relations.
 - This example shows how to populate a ProDataSet using the fill method.
 - It defines a browse built on the query associated with the tstudent temp-table that is defined on the ProDataSet dsstudent.
 - In general, it is recommended that the DataSet temp-tables should be as normalized and concise as possible irrespective of how the data is stored in the database tables.
 - This allows the business logic and data display logic to reference properly organized data in the form best suited for the application.
 - Use the ProDataSet mapping features and custom logic to map the temp-table representation to the existing database design.
 - The advantage to this approach is that if the database changes later, only the mapping needs to change; the internal ProDataSet logic may remain the same.
 - Another recommendation is to use include files to define the temp-tables used in a ProDataSet. The same temp-tables may be used in many DataSet definitions and/or many procedures. Therefore the include files may be reused many times.



Populating a ProDataSet

There are 8 steps to building a ProDataSet.

1. Define the temp-table used in a ProDataSet.
2. Define the ProDataSet.
3. Define a database query used in the top-level data-source.
4. Define the data-sources used with the ProDataSet.
5. Execute the **attach-data-source** method for the data-sources.
6. Execute a **query-prepare** method on the query associated with the top-level data-source.
7. Execute the **fill** method on the ProDataSet.
8. Execute the **detach-data-source** method for the data-sources.



ATTACH-DATA-SOURCE Method

- The buffer-handle is the handle to the temp-table buffer defined in the ProDataSet.
- The datasource-hdl is the handle to the data-source object.
- The optional pairs-list is a comma delimited list of field pairs between the data-source field and then the ProDataSet field.
- Each field pair equates the data-source field with the corresponding DataSet field in the pair during the attachment. This is useful if the temp-table fields are named differently from the database fields.
- If there are any fields in the database buffer that don't match a temp-table field and are not specified in the pairs-list, then they are skipped.
- Make sure there are no embedded spaces between fields.
- The optional except-fields list excludes fields in the list from being loaded on the fill method. This is useful to limit how many fields are passed to reduce network traffic.
- The optional include-fields list specifies only those fields that may be loaded on the fill method. Like the except-fields list, this is also useful to limit how many fields are passed to reduce network traffic.
- Either the except-fields list or the include-fields list may be used but not both.
- If the include-fields list is used, a question mark "?" must be used with a comma "," as a place holder for the exclude-fields list.
- It is possible to use the rowid(db-buffer-name),trowid-field in the pairs-list. This allows Progress to use the ROWID of the database record as the key to uniquely identify the record. In this case, the phrase KEYS(ROWID) should be specified in the data-source definition.
- In this example, pairs-list option for ATTACH-DATA-SOURCE wasn't used. This is because the temp-table tstudent was defined exactly like the database table student so there was no need to use the pairs list.
- However, the except-fields and include-fields options could have been used.

Syntax

ATTACH-DATA-SOURCE method

```
ATTACH-DATA-SOURCE ( datasource-hdl
[ [ [ , pairs-list ] , except-fields ] , include-fields ]
```



Populating a ProDataSet

- Progress provides the fill method for populating the entire DataSet or for populating data for a particular member buffer of a DataSet.
- ProDataSet-object-handle:FILL() or buffer-object-handle:FILL()
- The fill method provides a consistent and uniform way for populating data in the ProDataSet.
- As noted in the steps to building a ProDataSet, the query-prepare method **MUST** be used for each query that is to be filled. This is true even if the query is static. Progress cannot associate a static OPEN QUERY statement with a DataSet.

- In this simple example, the fill is executed in this way:

```
dataset dsstudent:fill().
```

- However, since the ProDataSet contains only one temp-table the fill command could have been executed this way with the same results:

```
buffer tstudent:fill().
```

- Like other Progress methods, handle variables could have been used. For example:

```
define var hds as handle no-undo.
```

```
hds = dataset dsstudent:handle.  
hds:fill().
```

OR

```
define var hbuff as handle no-undo.  
hbuff = buffer tstudent:handle.  
hbuff:fill().
```

- There will be more on the fill method in the next example.

Syntax

Fill Method

```
FILL()
```




Populating a ProDataSet

pds2.p

```
/* pds2.p dataset example two - added fill button to load additional data */

define temp-table tstudent no-undo like student
index studentid is unique studentid .

define dataset dsstudent for tstudent.
.
.
define query qstudent for student.

define data-source srcstudent
  for query qstudent
  student keys (studentid).
.
.
.
procedure fillds:
  buffer tstudent:attach-data-source(data-source srcstudent:handle,"").

  query qstudent:query-prepare("for each student no-lock
                                where stcode = "
                                + "'"
                                + stcodevar
                                + "'"
                                + " and syear = "
                                + string(syearvar)
                                ).

/* or use quoter function */
  query qstudent:query-prepare("for each student no-lock
                                where stcode = "
                                + quoter(stcodevar)
                                + " and syear = "
                                + string(syearvar)
                                ).

  if fillmodevar
  then buffer tstudent:fill-mode = "empty".
  else buffer tstudent:fill-mode = "merge".

  dataset dsstudent:fill().

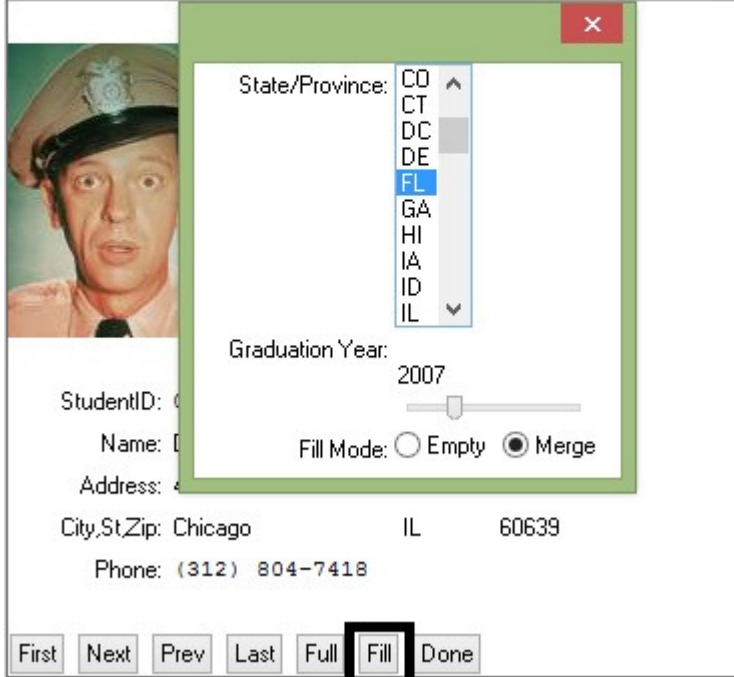
  buffer tstudent:detach-data-source().
end.
```



Populating a ProDataSet

Fill Method

StudentID	First Name	Last Name	GPA	Phone	State	Grd Yr
000206	Derwood	Glass	2.75	(312) 804-7418	IL	2007
001956	Diane	Huber	3.75	(312) 519-8147	IL	2007
002819	Quincy	Jacobson	2.80	(312) 765-0009	IL	2007
002037	Gladys	Larson	2.83	(312) 534-0669	IL	2007



State/Province: CO
CT
DC
DE
FL
GA
HI
IA
ID
IL

Graduation Year: 2007

Fill Mode: Empty Merge

StudentID: 0
Name: D
Address: 4
City,St,Zip: Chicago IL 60639
Phone: (312) 804-7418

First Next Prev Last Full **Fill** Done



Populating a ProDataSet

- In **pds2.p**, the **fill** method is executed as many times as the user desires to populate the DataSet with a particular group of records.
- Each buffer defined in a DataSet has an attribute called the **FILL-MODE**.
 - When the **FILL** method executes, Progress checks the **FILL-MODE** attribute for each buffer and acts according to the **FILL-MODE** specified.
 - The four **FILL-MODE** values are:
 - **EMPTY** - Any data that is in the temp-table for this particular buffer is emptied before the **FILL** method executes.
 - **NO-FILL** - No data is to be loaded into the temp-table. Possible reasons are that the temp-table has already been loaded and it is from a static database table such as state, or the temp-table will be filled in a separate operation.
 - **APPEND** - Records will be added to the temp-table in addition to the records already there. If the records added are duplicates based on the unique key for the table, an error condition will be raised. This is useful if a developer wants to be notified of duplicate records.
 - **MERGE** - This is the default **FILL-MODE**. Similar to **APPEND**, the **MERGE** mode adds records and automatically checks for duplicates. Duplicate records are not added and no error condition is raised. The **MERGE** mode requires that the temp-table have a unique index and the same field(s) are defined as keys for the data-source.
- In **pds2.p**, the user may press the Fill button and select query conditions stcode (State) and syear (graduation year) for the **FILL** method.
- In addition, the user may select either **EMPTY** or **MERGE** for the **FILL-MODE**. If **EMPTY** is selected, the current data in the tstudent temp-table is removed and replaced with the new records based upon the current query conditions. If **MERGE** mode is selected, new records are added to the existing set of records.
- The **QUOTER** function is useful for quoting string values in the **QUERY-PREPARE** method.



Populating a ProDataSet

pds3.p

```
/* pds3.p dataset example three - multi-table read-only dataset */

define temp-table tstudent no-undo like student
field chargetot as decimal label "Total Charges"
index studentid is unique primary studentid .

define temp-table tstuchrg no-undo like stuchrg.

define temp-table tcharge no-undo like charge.

define dataset dsstuchrg for tstudent, tstuchrg, tcharge
data-relation stuchrg for tstudent, tstuchrg
relation-fields (studentid, studentid)
data-relation charge for tstuchrg, tcharge
relation-fields (chargecode, chargecode).

define query qstudent for student.

define data-source srcstudent
  for query qstudent
  student keys (studentid).

define data-source srcstuchrg for stuchrg.

define data-source srccharge for charge.

.
.
.

buffer tstudent:attach-data-source(data-source srcstudent:handle,"").
buffer tstuchrg:attach-data-source(data-source srcstuchrg:handle,"").
buffer tcharge:attach-data-source(data-source srccharge:handle,"").

query qstudent:query-prepare("for each student
                             where stcode = 'IL'
                             and syear = 2007").

dataset dsstuchrg:fill().

buffer tstudent:detach-data-source().
buffer tstuchrg:detach-data-source().
buffer tcharge:detach-data-source().
```



Populating a ProDataSet

Student#	First Name	Last Name	GPA	Phone	Total Charges
000206	Derwood	Glass	2.75	(312) 804-7418	0.00
001956	Diane	Huber	3.75	(312) 519-8147	0.00
002037	Gladys	Larson	2.83	(312) 534-0669	0.00
002819	Quincy	Jacobson	2.80	(312) 765-0009	0.00

Charge No.	chargeDate	chargeCode	Amount	chargeCode	Description
014964	01/18/08	Other	\$50.00	book	Book Charge
014965	01/06/08	Other	\$50.00	food	Food Charge
014966	06/26/08	Other	\$12.00	other	Other Charge
014967	11/24/07	Other	\$50.00	room	Room Charge
014909	08/02/06	Room	\$400.00	tuition	Tuition Charge
014919	12/02/06	Room	\$550.00		

Done



- In **pds3.p**, we introduce a multiple table ProDataSet. The top-level temp-table is `tstudent` based on the student database table. The child table to `tstudent` is `tstuchrg` based on the `stuchrg` (Student Charge) database table. The third temp-table is `tcharge`, the parent of the `tstuchrg` temp-table and is based on the database table `charge`.
- The three temp-tables must be defined before the ProDataSet definition.
- Notice that there is a calculated field called `chargetot` defined on the `tstudent` temp-table. This field is displayed as 0 for each record in **pds3.p** and **pds4.p**. We will show you how to populate this field later using event procedures in **pds5.p**.
- With multiple temp-tables defined, it is desirable to define data-relations between the temp-tables.
- The **DATA-RELATION** name is optional. For example, the ProDataSet definition could have been written like this:

```
define dataset dsstuchrg for tstudent, tstuchrg, tcharge
data-relation          for tstudent, tstuchrg
relation-fields (studentid, studentid)
data-relation          for tstuchrg, tcharge
relation-fields (chargecode, chargecode).
```

- If the name isn't specified, then the first **DATA-RELATION** name is `relation1` and the second one is `relation2` by default.



Populating a ProDataSet

- The **RELATION-FIELDS** phrase specifies a comma-separated pair of fields in the order the buffers were specified in the **DATA-RELATION**.
- In the stuchrg data-relation, the first relation-field is from the tstudent temp-table and the second is from the tstuchrg temp-table.
- In the charge data-relation, the first relation-field is from the tstuchrg temp-table and the second is from the tcharge temp-table.
- Remember that Progress uses this data-relation information when ProDataSet is filled.
- Before the DataSet can be filled all defined data-sources for the ProDataSet must be attached, otherwise a run-time error occurs.
- The **FILL** method on the ProDataSet retrieves records from all data-sources into all temp-tables.
 - The **FILL** method starts by reading the first record in the top-level query, then proceeds down through the data-relation relationships recursively.
 - In pds3.p, the **FILL** method reads the first student record, then it reads the first stuchrg child record of the student record, followed by the parent charge record of the student charge record.

It then continues through all the child stuchrg records of the first student and their corresponding charge record before reading the second student record and repeating the process.
- It is possible to load the temp-tables in a non-nested manner during a **FILL** method if the **RELATIONS-ACTIVE** attribute on the ProDataSet object is set to false. Filling the DataSet this way also requires a query to be defined on each child temp-table.
- One issue on how we structured the DataSet in pds3.p is that the tcharge temp-table will only contain records that are related to the tstuchrg temp-table. However, there may be more records in the tcharge temp-table but do not relate to any children in tstuchrg table. Therefore, those tcharge records will not be included in the tcharge browse.
 - For example, there is a tax tcharge record, but in pds3.p, it will not be added from the **FILL** method unless the tstuchrg record is a tax charge.
 - If you want all tcharge records to be included in the DataSet, yet still be directly related to the tstuchrg temp-table, then the **REPOSITION** keyword should be added to the charge data-relation.



Populating a ProDataSet

pds4.p

```
/* pds4.p dataset example four - added reposition to tcharge data-relation */.
.
define dataset dsstuchrg for tstudent, tstuchrg, tcharge
data-relation stuchrg for tstudent, tstuchrg
relation-fields (studentid, studentid)
data-relation charge for tstuchrg, tcharge
relation-fields (chargecode, chargecode) REPOSITION.
.
.
.
```

Student#	First Name	Last Name	GPA	Phone	Total Charges
000206	Derwood	Glass	2.75	(312) 804-7418	0.00
001956	Diane	Huber	3.75	(312) 519-8147	0.00
002037	Gladys	Larson	2.83	(312) 534-0669	0.00
002819	Quincy	Jacobson	2.80	(312) 765-0009	0.00

Charge No.	chargeDate	chargeCode	Amount	chargeCode	Description
011599	05/06/08	Other	\$18.00	book	Book Charge
011600	11/28/07	Other	\$40.00	finance	Finance Charge
011601	03/02/08	Other	\$50.00	food	Food Charge
011555	08/02/06	Room	\$400.00	other	Other Charge
011563	12/02/06	Room	\$550.00	room	Room Charge
011571	03/02/07	Room	\$700.00	tax	Tax Charge

Done

- In pds4.p, the REPOSITION keyword was added, which allows loading of the entire charge table into the tcharge temp-table.
 - This makes it easier to add new stuchrg records and allows the user to select the correct charge from the browse.
 - Think of REPOSITION as sort of an "outer-join", where the parent table displays all records whether or not the child records relate to all the parent records or not.



Populating a ProDataSet

pds5.p

```

/* pds5.p dataset example five - added after-fill callback procedure */
.
.
.
default-window:width = 100.

buffer tstuchrg:set-callback-procedure
    ("after-fill", "poststuchrgFill", THIS-PROCEDURE).

query qstudent:query-prepare("for each student
                               where stcode = 'IL'
                               and syear = 2007").
.
.
.

wait-for close of this-procedure.

procedure poststuchrgfill:
    define input parameter dataset for dsstuchrg.
    for each tstuchrg of tstudent:
        accumulate chargeamt (total).
    end.
    assign chargetot = accum total chargeamt.
end.
    
```

Student#	First Name	Last Name	GPA	Phone	Total Charges ^
000206	Derwood	Glass	2.75	(312) 804-7418	27,646.00
001956	Diane	Huber	3.75	(312) 519-8147	25,371.00
002037	Gladys	Larson	2.83	(312) 534-0669	25,362.00
002819	Quincy	Jacobson	2.80	(312) 765-0009	25,020.00 v

Charge No.	chargeDate	chargeCode	Amount ^	chargeCode	Description ^
013057	01/03/08	Other	\$60.00	finance	Finance Charge
013058	11/18/07	Other	\$30.00	food	Food Charge
013059	04/08/08	Other	\$12.00	other	Other Charge
013001	08/02/06	Room	\$400.00	room	Room Charge
013011	12/02/06	Room	\$550.00	tax	Tax Charge
013021	03/02/07	Room	\$700.00 v	tuition	Tuition Charge v

Done



Populating a ProDataSet

- In pds5.p, we introduce event procedures or callback procedures to help populate the total charges per student.
- The **SET-CALLBACK-PROCEDURE** method associates a named event with an internal procedure to run when the event occurs, and the persistent procedure handle indicating where to run the internal procedure.
 - callback-name is the event of when the internal procedure executes internal-procedure is the name of the internal procedure procedure-context is the persistent procedure handle where the internal procedure will be executed.
 - Internal-procedure is the name of the internal procedure
 - Procedure-context is the persistent procedure handle where the internal procedure will be executed.
- The following are the Progress supplied DataSet events:

BEFORE-FILL
AFTER-FILL
BEFORE-ROW-FILL
AFTER-ROW-FILL

- The top two of these events may be used on both the DataSet and the DataSet temp-table buffer handle. The latter two may only be used for the DataSet temp-table buffer handle.
- In pds5.p, the **AFTER-FILL** callback procedure is used on the tstuchrg temp-table buffer. This means after the tstuchrg temp-table records are filled for a particular tstudent, the poststuchrgfill procedure is executed. It reads all the tstuchrg records of the tstudent record and calculates the total for the chargetot field in the tstudent table.
- The **SET-CALLBACK** method is used in OO classes to associate a named event with a *routine-name* in a *routine-context*. A *routine-name* can refer to either an OO method or an internal procedure. A *routine-context* refers to an object reference for a class instance or a handle to a persistent procedure that contains the method or internal procedure specified by *routine-name*.

Syntax

SET-CALLBACK-PROCEDURE Method

```
SET-CALLBACK-PROCEDURE (callback-name, internal-procedure
                        [ , procedure-context ])
```

SET-CALLBACK Method

```
SET-CALLBACK ( callback-name , routine-name [ , routine-context ] )
```



Lab 1 – Reading Records into a ProDataSet

1. Create Your First ProDataSet
 - a. Copy teacher1.p in the labs folder to teacher1lab.p.
 - b. Do the 8 steps needed to use a ProDataSet in the procedure for the teacher table.
 - c. Put the 8 steps in the procedure where indicated by the comments.

2. Fill the ProDataSet
 - a. Copy teacher2.p in the labs folder to teacher2lab.p.
 - b. Setup the query-prepare method based upon the selection of the lastnamevar and gendervar.
 - c. Assign the corresponding fill-mode attribute based upon the fillmodevar variable.

3. Create a Multi-Table ProDataSet
 - a. Copy offering1.p in the labs folder to offering1lab.p.
 - b. Define temp-tables like the database tables:
 - i. teacher
 - ii. offering
 - iii. course
 - iv. season
 - c. Define the dataset ds offering referencing the 4 temp-tables in the following order:
 - i. tteacher
 - ii. toffering
 - iii. tcourse
 - iv. tseason

4. Allow all courses to be loaded into the ProDataSet, even ones not linked to the Teachers, Offerings and Courses
 - a. Copy offering2.p in the labs folder to offering2lab.p

5. Calculate the total number of offerings per teacher and store it in the offeringcount temp-table field.
 - a. Copy offering2.p in the labs folder or offering2sol.p in the solutions folder to offering3lab.p
 - b. Use the set-callback-procedure to calculate the number of offerings for each teacher



Advanced Reading Operations

In this section, the following topics will be covered:

- Batching data with ProDataSets
- Using Multiple Database Buffers on a Data-Source
- Query on a child temp-table

Batching Data with ProDataSets

Up to this point, the examples shown have used a filter on the top level data-source to limit the number of records read.

What if a filter is not desired, how do we limit the number of records being read on a large table?

Also, what if not all the fields in the record need to be read?

OpenEdge provides ways to batch records into a ProDataSet.

The following language components will be used to control the data read into ProDataSets:

Component	Type
OFF-END	ProDataSet Event
BATCH-SIZE	Temp-Table Buffer Attribute
LAST-BATCH	Temp-Table Buffer Attribute
NEXT-ROWID	Data-Source Attribute
RESTART-ROWID	Data-Source Attribute



Advanced Reading Operations

Batching Data with ProDataSets

pds23.p

```
/* pds23.p dataset twenty-three - one table read-only dataset with batching all in one procedure
no appserver */
.
.
.
def var vFieldList as char init
"StudentID,sfirstname,slastname,phone,city,stcode,postalcode".
.
.
.
query qtstudent:set-callback-procedure("off-end", "offend").

run fillds (input 2975, output dataset dsstudent).

message "Show batching message when browse goes off the end?"
  view-as alert-box question buttons yes-no update showmsg.
.
.
.
wait-for close of this-procedure.

procedure fillds:
  define input parameter ipstudentid as int.
  define output parameter dataset for dsstudent.

  buffer tstudent:attach-data-source(data-source srcstudent:handle,"", ?, vFieldList).

  buffer tstudent:batch-size = 10.

  query qstudent:query-prepare("for each student no-lock where student.studentid > " +
                                string(ipstudentid)).

  dataset dsstudent:fill().

  buffer tstudent:detach-data-source().

end.

procedure offend:
  define input parameter dataset for dsstudent.

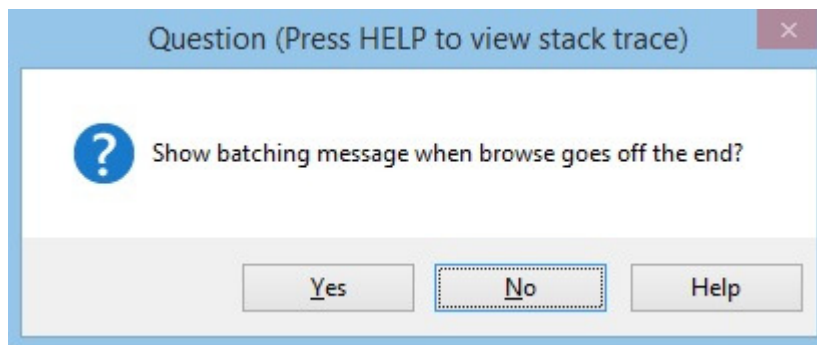
  if showmsg then
  message "inside offend" skip
    "last batch?" buffer tstudent:last-batch skip
  view-as alert-box.
  if not buffer tstudent:last-batch then do:
    find last tstudent.
    run fillds (tstudent.studentid, output dataset dsstudent ).
    return no-apply.
  end.
end.
```



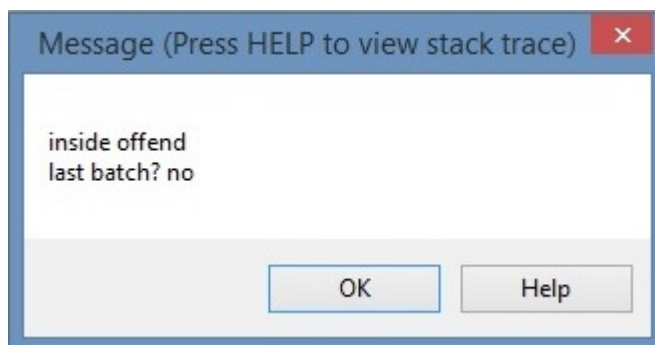
Advanced Reading Operations

Batching Data with ProDataSets

- In pds23.p, the student table is read in studentid order.
- The user is prompted when the program executes asking to show the batching message when the browse goes off the end.



- To demonstrate the last-batch attribute efficiently, the data-source starts reading student records at 2,975 (out of approximately 3,000). This is shown by the statement:
- run fillds (input 2975, output dataset dsstudent).
- Inside procedure fillds, the batch-size is set to 10 for the temp-table buffer tstudent.
- When the fill method executes, only ten records are read into the temp-table.
- When the user scrolls down the browse after ten records and more records remain, the following message is displayed:



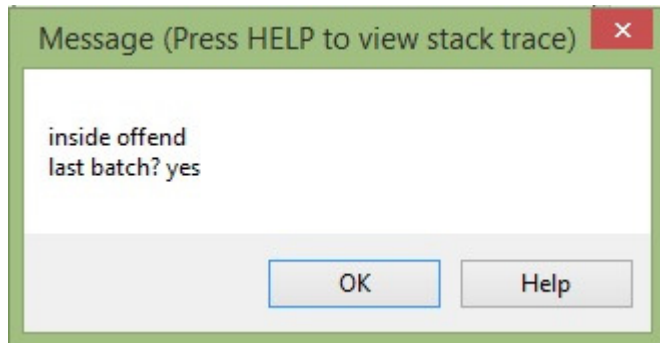
- Since more records remain, then the last batch attribute is set to no.



Advanced Reading Operations

Batching Data with ProDataSets

- Once the last record is read into the temp-table buffer and the user attempts to scroll forward in the browse, then the last batch attribute is set to yes.



- The RETURN NO-APPLY is very important to put into this event procedure. Without it, the program will go into an infinite loop.
- The vfieldlist variable consists of fields used in the browse and frame from the student table. When this variable is specified in the include field list parameter on the attach-data-source method then only those fields are populated from the database.
- This is used to limit network traffic.
- Notice when the full button is pressed, the fields not included in the list are shown as blank.

StudentID: 002976	First Name: Ed
Last Name: Kay	
Address:	
address2:	
address3:	
city: Charleston	State/Province: SC
Zip Code: 29415	Country: USA
addressAgg:	<input type="text"/>
Phone: (843) 436-9717	
email:	
Ethnic ID: 000000	Sex: Male
Birthday:	Year: 1997
Season#: 0	Graduated?: no
GPA: ?	Balance: \$0.00



Advanced Reading Operations

Batching Data with ProDataSets

pds24.p

```

/* pds24.p dataset twenty-four - one table read-only dataset with batching all in one procedure
with appserver */
.
.
.
create server hapsrv.

ok = hapsrv:connect("-AppService aspdsdbaschool -H localhost -sessionModel session-free").
if not ok then do:
  message "Failed to connect to Appserver" view-as alert-box.
  return no-apply.
end.
else message "Connection successful" view-as alert-box.

query qtstudent:set-callback-procedure("off-end", "offend").
run pds24callee.p on server hapsrv (input 2975, output dataset dsstudent append).
.
.
message "Show batching message when browse goes off the end?"
  view-as alert-box question buttons yes-no update showmsg.

wait-for close of this-procedure.

procedure offend:
  define input parameter dataset for dsstudent.

  if showmsg then
    message "inside offend" skip
      "last batch?" buffer tstudent:last-batch skip
  view-as alert-box.
  if not buffer tstudent:last-batch then do:
    find last tstudent.
    run pds24callee.p on server hapsrv (tstudent.studentid, output dataset dsstudent append).
    return no-apply.
  end.
end.
.
.

```

pds24callee.p

```

/* pds24callee.p - Fill Dataset for calling program */

def var vFieldList as char init
"StudentID,sfirstname,slastname,phone,city,stcode,postalcode".
define temp-table tstudent no-undo like student
index studentid is unique studentid .

define dataset dsstudent for tstudent.

define query qstudent for student.

define data-source srcstudent
  for query qstudent

```



```
student keys (studentid).

define input parameter ipstudentid as int.
define output parameter dataset for dsstudent.
buffer tstudent:batch-size = 10.

buffer tstudent:attach-data-source(data-source srcstudent:handle,"", ?, vFieldList).

query qstudent:query-prepare("for each student no-lock where student.studentid > " +
                             string(ipstudentid)).

dataset dsstudent:fill().
buffer tstudent:detach-data-source().
```




Advanced Reading Operations

Batching Data with ProDataSets

In pds24.p, batching is performed on the AppServer by calling fillds in pds24callee.p.

student			
StudentID	First Name	Last Name	Phone
002976	Ed	Kay	(843) 436-9717
002977	Debbie	Reilly	(615) 557-922E
002978	Enid	Anderson	(701) 325-020C
002979	Eugene	Dechter	(214) 857-828E
002980	Judith	Mages	(907) 686-5521
002981	Laney	Peters	(804) 754-038C

StudentID: 002976
Name: Ed Kay
City,St,Zip: Charleston SC 29415
Phone: (843) 436-9717

- In using the **OFF-END** callback event remember:
 - You can attach these events only to a query on a single ProDataSet temp-table buffer. You cannot attach these events to a query on a database buffer, or a query that involves a join.
 - The query must be a scrolling query.
 - If you never **RETURN NO-APPLY**, from the **OFF-END** event handler, the query will infinitely loop!
 - Call the **SET-CALLBACK-PROCEDURE()** method before the query is opened.
 - If you use the **GET LAST** statement or **GET-LAST()** method to get the last record associated with the query, the event handler is called repeatedly until it does not **RETURN NO-APPLY** (indicating that all records have been retrieved). For this reason, use caution when offering users the **GET LAST** action or avoid this.
 - The **INDEXED-REPOSITION** option is ignored for the query.
 - The **APPEND** keyword is needed going to the AppServer which appends the new batching data with the existing set of records. **APPEND** is not needed if it is a local call.



Advanced Reading Operations

Batching Data with ProDataSets

pds25.p

```
/* pds25.p dataset twenty-five - one table read-only dataset with batching all in one procedure
no appserver sorting on student name */
.
def var restartrowid as rowid.
.
run fillds (input-output restartrowid, output dataset dsstudent).
.
.
procedure fillds:
  define input-output parameter iopnextrowid as rowid.
  define output parameter dataset for dsstudent.

  buffer tstudent:attach-data-source(data-source srcstudent:handle,"", ?, vFieldList).

  query qstudent:query-prepare("for each student no-lock
                                where student.studentid > 2975
                                by student.slastname by student.sfirstname" ).

  buffer tstudent:batch-size = 10.

  data-source srcstudent:restart-rowid = iopnextrowid.

  dataset dsstudent:fill().

  iopnextrowid = data-source srcstudent:next-rowid.

  buffer tstudent:detach-data-source().

end.

procedure offend:
  define input parameter dataset for dsstudent.

  if showmsg then
  message "inside offend" skip
         "last batch?" buffer tstudent:last-batch skip
  view-as alert-box.
  if not buffer tstudent:last-batch then do:
    find last tstudent use-index name.
    run fillds (input-output restartrowid, output dataset dsstudent ).
    return no-apply.
  end.
end.
```



Advanced Reading Operations

Batching Data with ProDataSets

student			
StudentID	First Name	Last Name	Phone
002995	Jason	Albert	(314) 725-3301
002987	Kathy	Ali	(843) 592-6891
002978	Enid	Anderson	(701) 325-0200
002989	Lucyna	Berman	(907) 336-4877
002982	Jack	Bishop	(302) 927-5495
002984	Ahmad	Chavez	(502) 506-2905

StudentID: 002995
Name: Jason Albert
City,St,Zip: St. Louis MO 63102
Phone: (314) 725-3301

First	Next	Prev	Last	Full	Done
-------	------	------	------	------	------

- What if batching is to be performed where the records are sorted in an order with a non-unique index?
- In pds25.p, the student table is sorted by last name, first name. Again, for efficiency, we are only looking at the records where the studentid > 2975. The **QUERY-PREPARE** statement looks like this:

```
query qstudent:query-prepare("for each student no-lock
                               where student.studentid > 2975
                               by student.slastname
                               by student.sfirstname" ).
```

- Each time the **FILL** method executes, the fillds procedure retrieves the **NEXT-ROWID** attribute and assigns it to the restartrowid variable.
- In order for the data-source to pick up where it left off, it assigns this saved rowid into the **RESTART-ROWID** attribute for the data-source before performing the **FILL** method.



Advanced Reading Operations

Batching Data with ProDataSets

pds26.p

```

/* pds26.p dataset twenty-six - one table read-only dataset with batching with appserver sorting
on student name */
.
.
create server hapsrv.

ok = hapsrv:connect("-AppService aspdsdbaschool -H localhost -sessionModel session-free", "",
""").
if not ok then do:
  message "Failed to connect to Appserver" view-as alert-box.
  return no-apply.
end.
else message "Connection successful" view-as alert-box.

query qtstudent:set-callback-procedure("off-end", "offend").

run pds26callee.p on server hapsrv
      (input-output restartrowid, output dataset dsstudent append).
.
.
wait-for close of this-procedure.

procedure offend:
  define input parameter dataset for dsstudent.

  if showmsg then
    message "inside offend" skip
      "last batch?" buffer tstudent:last-batch skip
  view-as alert-box.
  if not buffer tstudent:last-batch then do:
    run pds26callee.p on server hapsrv
      (input-output restartrowid, output dataset dsstudent append).
    return no-apply.
  end.
end.
.
.

```

pds26callee.p

```

/* pds26callee.p - Fill Dataset for calling program sorting by student name */

def var vFieldList as char init
"StudentID,sfirstname,slastname,phone,city,stcode,postalcode".

define temp-table tstudent no-undo like student
index studentid is unique studentid .

define dataset dsstudent for tstudent.

define query qtstudent for student scrolling.

```



```
define data-source srcstudent
  for query qstudent
    student keys (studentid).

define input-output parameter iopnextrowid as rowid.
define      output parameter dataset for dsstudent.

buffer tstudent:attach-data-source(data-source srcstudent:handle,"", ?, vFieldList).

query qstudent:query-prepare("for each student no-lock
                             where student.studentid > 2975
                             by student.slastname by student.sfirstname" ).

buffer tstudent:batch-size = 10.

data-source srcstudent:restart-rowid = iopnextrowid.

dataset dsstudent:fill().

iopnextrowid = data-source srcstudent:next-rowid.

buffer tstudent:detach-data-source().
```



Advanced Reading Operations

Batching Data with ProDataSets

In pds26.p, batching on a non-unique sorting of temp-table records is performed using an AppServer.

Passing the dataset as an output parameter with **APPEND** tells Progress to *append* all the data passed back to the data that is *already* in the client's ProDataSet.

We will cover parameter passing in more detail later.



Advanced Reading Operations

Multiple Database Buffers on a Data-Source

pds30.p

```
/* pds30.p dataset example thirty - multiple database buffers on a data-source */
.
.
define query qstudent for activity, stuact, student.

define data-source srcstudent
  for query qstudent
  activity, stuact, student keys (studentid).
.
.
.
procedure fillds:

  buffer tstudent:attach-data-source(data-source srcstudent:handle,"").
  buffer tstuchrg:attach-data-source(data-source srcstuchrg:handle,"").
  buffer tcharge:attach-data-source(data-source srccharge:handle,"").

  query qstudent:query-prepare("for each activity
                                where activityid = "
                                + string(activityidvar)
                                + ", each stuact of activity,
                                each student of stuact
                                where syear = " + string(syearvar)
                                + " and gpa ge " + string(gpavar) ).

  if fillmodevar
  then buffer tstudent:fill-mode = "empty".
  else buffer tstudent:fill-mode = "merge".

  dataset dsstuchrg:fill().

  buffer tstudent:detach-data-source().
  buffer tstuchrg:detach-data-source().
  buffer tcharge:detach-data-source().

  run openq.
  apply "value-changed" to b1 in frame f1.
end.
```



Advanced Reading Operations

Multiple Database Buffers on a Data-Source

- Sometimes, the relationship between the number of database buffers in the data-source and receiving temp-table is not one to one.
- To get the desired subset of records for a particular temp-table it might be necessary to use several database buffers to filter the records.
- In pds30.p, the query qstudent is based upon the activity, stuact and student database buffers.
- The initially loaded subset is based upon students that have chess as an activity, are the class of 2008 and have a GPA greater than equal to 3.0.
- The fillds procedure can be used to put in different conditions like it did in pds2.p.

Student#	First Name	Last Name	GPA	Phone	Total Charges
000122	Jill	French	3.00	(843) 574-1047	36,842.00
001257	Diane	Blyth	3.20	(207) 389-6278	37,112.00
001307	Arnold	Moss	3.14	(208) 689-6595	37,645.00
002692	Lucy	Hudson	3.33	(406) 758-0141	37,744.00



Charge No.	chargeDate	chargeCode	Amount	chargeCode	Description
028476	08/28/06	Book	\$325.00	book	Book Charge
028482	12/27/06	Book	\$450.00	finance	Finance Charge
028488	04/02/07	Book	\$575.00	food	Food Charge
028494	08/28/07	Book	\$325.00	other	Other Charge
028500	12/27/07	Book	\$450.00	room	Room Charge
028506	04/02/08	Book	\$575.00	tax	Tax Charge

Fill Done

Activity: Cycling

Graduation Year: 2008

GPA Threshold: 2 - 5p

Fill Mode: Empty Merge



Advanced Reading Operations

Multiple Database Buffers on a Data-Source

pds31.p

```

/* pds31.p dataset example thirty-one - multiple database buffers on a data-source */

procedure fillds:
.
.
.
  data-source srcstuchrg:fill-where-string =
  data-source srcstuchrg:fill-where-string + " and stuchrg.chargeamt > "
  + string(chargevar).
end.

```

Student#	First Name	Last Name	GPA	Phone	Total Charges
000122	Jill	French	3.00	(843) 574-1047	32,475.00
001257	Diane	Blyth	3.20	(207) 389-6278	32,475.00
001307	Arnold	Moss	3.14	(208) 689-6595	32,475.00
002692	Lucy	Hudson	3.33	(406) 758-0141	32,475.00



Charge No.	chargeDate	chargeCode	Amount	chargeCode	Description
028488	04/02/07	Book	\$575.00	book	Book Charge
028506	04/02/08	Book	\$575.00	finance	Finance Charge
028524	04/02/09	Book	\$575.00	food	Food Charge
028481	12/02/06	Room	\$550.00	other	Other Charge
028487	03/02/07	Room	\$700.00	room	Room Charge
028499	12/02/07	Room	\$550.00	tax	Tax Charge

Fill Done

Activity: chess

Graduation Year: 2008

GPA Threshold: 3.00

Charge Amount Threshold: \$500.00

Fill Mode: Empty Merge



Advanced Reading Operations

Multiple Database Buffers on a Data-Source

- In pds31.p, the **FILL-WHERE-STRING** is used on the stuchrg buffer to further filter the records to only those charges with a charge amount greater than or equal to a value.
- This was added to the fill dialog box.



Advanced Reading Operations

Query on a child temp-table

- Sometimes, it is desirable to setup a query on a child temp-table.
- For example, suppose you want to show all of the activities for a particular student.
- The relationship between student and activity is many to many. This means that a particular student may be involved in many activities, and a particular activity may have many students engaged with it.
- In the school database, there are the student and activity master tables and the stuact cross reference table.
- The unique keys for the student and activity table are studentid and activityid respectively.
- The stuact table has a unique key based upon studentid and activityid.
- Here is how the dataset consisting of the three temp-tables appears with 3 browses:

pds32.p

Student#	First Name	Last Name	GPA	Phone
000206	Derwood	Glass	2.75	(312) 804-7430
001956	Diane	Huber	3.75	(312) 519-8147
002037	Gladys	Larson	2.83	(312) 534-0669
002819	Quincy	Jacobson	2.80	(312) 765-0009

Student#	Activity ID	Activity ID Name
002819	000013	000024 Chugging Beer
002819	000027	000026 Marching Band
002819	000031	000027 Jazz Band
		000028 Orchestra
		000031 Yachtzee

Done





Advanced Reading Operations

Query on a child temp-table

pds32.p

```
/* pds32.p dataset example three - multi-table read-only dataset with student, stuact and
activity */
.
.
define temp-table tstudent no-undo like student
index studentid is unique primary studentid .

define temp-table tstuact no-undo like stuact.

define temp-table tactivity no-undo like activity.

define dataset dsactivity for tstudent, tstuact, tactivity
data-relation stuact for tstudent, tstuact
relation-fields (studentid, studentid)
data-relation activity for tstuact, tactivity
relation-fields (activityid, activityid).

define query qstudent for student.

define data-source srcstudent
  for query qstudent
  student keys (studentid).

define data-source srcstuact for stuact.

define data-source srcactivity for activity.

define query qtstudent for tstudent.

define browse b1 query qtstudent
  display tstudent.studentid label "Student#"
  tstudent.sfirstname
  tstudent.slastname
  tstudent.gpa column-label "GPA"
  tstudent.phone Label "Phone" width 16
  with 4 down.

define query qtstuact for tstuact.

define browse b2 query qtstuact
  display tstuact.studentid label "Student#"
  tstuact.activityid
  with 6 down.

define query qtactivity for tactivity.

define browse b3 query qtactivity
  display tactivity.activityid
  tactivity.activityname
  with 6 down.
.
.
```



```

buffer tstudent:attach-data-source(data-source srcstudent:handle,"").
buffer tstuact:attach-data-source(data-source srcstuact:handle,"").
buffer tactivity:attach-data-source(data-source srcactivity:handle,"").

query qstudent:query-prepare("for each student
                             where stcode = 'IL'
                             and syear = 2007").

dataset dsactivity:fill().

buffer tstudent:detach-data-source().
buffer tstuact:detach-data-source().
buffer tactivity:detach-data-source().

enable all with frame f1.

open query qtstudent for each tstudent.
open query qtstuact for each tstuact of tstudent.
open query qtactivity for each tactivity.

b1:select-focused-row().

apply "value-changed" to b1.


wait-for close of this-procedure.

```

- It might be more convenient to view the activities in a single browse.

pds33.p

Student#	First Name	Last Name	GPA	Phone
000206	Derwood	Glass	2.75	(312) 804-7430
001956	Diane	Huber	3.75	(312) 519-8147
002037	Gladys	Larson	2.83	(312) 534-0669
002819	Quincy	Jacobson	2.80	(312) 765-0009



Activity

- golf
- Jazz Band
- Yachtzee

Done



Advanced Reading Operations

Query on a child temp-table

```

/* pds33.p dataset example three - multi-table read-only dataset with student and activity */
.
.
define temp-table tstudent no-undo like student
index studentid is unique primary studentid .

define temp-table tstuact no-undo like stuact
field activityname like activity.activityname.

define dataset dsactivity for tstudent, tstuact
data-relation stuact for tstudent, tstuact
relation-fields (studentid, studentid).

define query qstudent for student scrolling.

define data-source srcstudent
  for query qstudent
  student keys (studentid).

define query qstuact for stuact, activity scrolling.

define data-source srcstuact
  for query qstuact
  stuact keys (studentid, activityid), activity.

define query qtstudent for tstudent.

define browse b1 query qtstudent
  display tstudent.studentid label "Student#"
  tstudent.sfirstname
  tstudent.slastname
  tstudent.gpa column-label "GPA"
  tstudent.phone label "Phone" width 16
  with 4 down.

define query qtstuact for tstuact.

define browse b2 query qtstuact
  display tstuact.activityname label "Activity"
  with 5 down.
.
.
default-window:width = 110.

buffer tstudent:attach-data-source(data-source srcstudent:handle,"").
buffer tstuact:attach-data-source(data-source srcstuact:handle,"").

query qstudent:query-prepare("for each student
                             where stcode = 'IL'
                             and syear = 2007").

query qstuact:query-prepare("for each stuact where stuact.studentid = tstudent.studentid, "
                             + " each activity of stuact").

dataset dsactivity:fill().

```



```
buffer tstudent:detach-data-source().
buffer tstuact:detach-data-source().

enable all with frame f1.

open query qtstudent for each tstudent.
open query qtstuact for each tstuact of tstudent.

b1:select-focused-row().

apply "value-changed" to b1.

wait-for close of this-procedure.
```

- To accomplish this, the activityname is added to the tstuact temp-table.
- The tactivity temp-table is eliminated and removed from the dataset dsactivity.
- The query qstuact is created for the stuact and activity tables.
- The data-source srcstuact is modified to include the qstuact query and the stuact and activity tables.
- The **QUERY-PREPARE** method is created for the qstuact query. It links the stuact database table with the **tstudent** temp-table so that the **FILL** method will continue properly.



Lab 2 - Advanced Reading Operations

1. Batching records to a Prodataset
 - a. Copy coursebatch1.p in the labs folder to coursebatch1lab.p.
 - b. This program will batch course records by course name where the courseid is greater than 50
 - c. Follow the commented instructions in the internal procedure fillds.

2. Limit fields to be sent from the server
 - a. Copy coursebatch1lab.p in the labs folder to coursebatch2lab.p.
 - b. The course description is too wordy, only allow fields courseid, coursename and deptname to be filled from the server.


3. Using multiple database buffers on a Data-Source
 - a. Copy studentmultdb.p in the labs folder to studentmultdbl.p.
 - b. This query asks the question who are the students from a given zip code range that received a grade of something or better for a particular course?
 - c. This involves joining the following tables together for the **QUERY**, **DATA-SOURCE** statements and the **QUERY-PREPARE** method.
 - i. student
 - ii. registration
 - iii. grade
 - iv. offering
 - v. course
 - d. You might want to consider making the course table first in the query for performance reasons.
 - e. Follow the commented instructions in the internal procedure fillds and at the top of the program.



Lab 2 - Advanced Reading Operations

4. Query on a Child Temp-table
 - a. Copy teacheroffcourse.p to teacheroffcourselab.p.
 - b. Run the program. It is shown below:

Teacher#	First Name	Last Name
000028	June	Bartlett
000029	Byron	Crawford
000030	Darrin	Brown
000031	Kevin	McAllister
000032	Lucyna	Segal
000033	Howard	Pasquesi




Course ID	Year	Season#	Course ID	Name
000011	2006	1	000010	Introduction to Biology
000014	2006	3	000011	Intermediate Disection of Frogs
000015	2007	1	000012	Molecular Biology
000014	2007	3	000013	Zoology
000010	2008	3	000014	Botany
000015	2009	1	000015	Anatomy and Physiology
000014	2009	2		

Done

- c. Modify program to combine the two lower browses into one browse consisting of the columns Course ID, Year, Season Name and Course Name. (see below)

Teacher#	First Name	Last Name
000028	June	Bartlett
000029	Byron	Crawford
000030	Darrin	Brown
000031	Kevin	McAllister
000032	Lucyna	Segal
000033	Howard	Pasquesi



Course ID	Year	Name	Name
000011	2006	Fall	Intermediate Disection of Frogs
000014	2006	Spring	Botany
000015	2007	Fall	Anatomy and Physiology
000014	2007	Spring	Botany
000010	2008	Spring	Introduction to Biology
000015	2009	Fall	Anatomy and Physiology
000014	2009	Winter	Botany

Done

- d. Follow the commented instructions inside the program.



Maintaining ProDataSet Changes

- The next section will cover how the temp-tables in ProDataSets are updated. This area will show how flexible and easy it is to undo individual records or the entire ProDataSet.
- The following attributes and methods address updating ProDataSets:

Keyword	Type
TRACKING-CHANGES	Attribute
ROW-STATE	Attribute, Function
BEFORE-ROWID	Attribute
AFTER-ROWID	Attribute
BEFORE-TABLE	Attribute
AFTER-TABLE	Attribute
BEFORE-BUFFER	Attribute
AFTER-BUFFER	Attribute
ACCEPT-CHANGES	Method
ACCEPT-ROW-CHANGES	Method
REJECT-CHANGES	Method
REJECT-ROW-CHANGES	Method

- The first thing that is required for tracking changes in a DataSet is defining a **BEFORE-TABLE** for each temp-table that changes are to be tracked.



Maintaining ProDataSet Changes

```

/* pds6.p dataset example six - reject changes and row changes */

def var choice as log no-undo.
def var i      as int no-undo.

/* make sure all temp-tables are no-undo otherwise the refresh
after the reject-row-changes and reject-changes will not work */

define temp-table tstudent no-undo like student
before-table tstudentb4
field chargetot as decimal label "Total Charges"
index studentid is unique primary studentid .

define temp-table tstuchrg no-undo like stuchrg
before-table tstuchrgb4.

define temp-table tcharge no-undo like charge.

define dataset dsstuchrg for tstudent, tstuchrg, tcharge
data-relation stuchrg for tstudent, tstuchrg
relation-fields (studentid, studentid)
data-relation charge for tstuchrg, tcharge
relation-fields (chargecode, chargecode) reposition.

define query qstudent for student.

define data-source srcstudent
  for query qstudent
  student keys (studentid).

define data-source srcstuchrg for stuchrg.

define data-source srccharge for charge.

define query qtstudent for tstudent.

define browse b1 query qtstudent
  display tstudent.studentid label "Student#"
         tstudent.sfirstname
         tstudent.slastname
         tstudent.gpa column-label "GPA"
         tstudent.phone label "Phone" width 16
         chargetot
  enable tstudent.gpa tstudent.phone
  with 4 down.

define query qtstuchrg for tstuchrg.

define browse b2 query qtstuchrg
  display /* tstuchrg.studentid label "Student#" */
         tstuchrg.chargeno
         tstuchrg.chargedate
         tstuchrg.chargecode
         tstuchrg.chargeamt
  enable tstuchrg.chargedate tstuchrg.chargecode tstuchrg.chargeamt
  with 6 down.

define query qtcharge for tcharge.

```



```

define browse b3 query qtcharge
  display tcharge.chargecode
    tcharge.chargedesc
  with 6 down.

define button brrow label "Reject Student Charge Row".
define button brallrow label "Reject All Student Charge Rows".
define button brall label "Reject All".
define button bdone label "Done".

define variable rstable as log label "Reject"
view-as radio-set horizontal radio-buttons
"Student", yes, "Student Charge", no.

define frame f1
b1 skip(1) b2 b3 skip(1) rstable
skip(1)
brall brallrow brrow bdone
with side-labels width 100.

on value-changed of rstable
do:
  assign rstable
    brrow:label = "Reject "
  + entry(lookup(rstable:screen-value,rstable:radio-buttons) - 1,rstable:radio-buttons)
    + " Row"
    brallrow:label = "Reject All "
  + entry(lookup(rstable:screen-value,rstable:radio-buttons) - 1,rstable:radio-buttons)
    + " Rows" .
end.

on choose of brallrow
do:
  def var msg as char.
  if rstable
  then
  msg = "Do you want to reject ALL Student's row changes?".
  else
  msg = "Do you want to reject ALL Student Charges's row changes?".
  message msg skip
    view-as alert-box
    question buttons yes-no update choice.
  if choice then do:
  if rstable then do:
  /* need to find modified tstudent, since current row may not be
  changed. */
  find first tstudent where row-state(tstudent) gt 0 no-error.
  if available tstudent then do:
  find tstudentb4 where rowid(tstudentb4) = buffer tstudent:before-rowid
  no-error.
  if available tstudentb4 then do:
  buffer tstudentb4:reject-changes() no-error.
  b1:refresh().
  end. /* available tstudentb4 */
  end. /* available tstudent */
  end. /* rstable */
  else do:
  /* need to find modified tstuchrg, since current row may not be
  changed. */
  find first tstuchrg where row-state(tstuchrg) gt 0 no-error.
  if available tstuchrg then do:

```



```

find tstuchrgb4 where rowid(tstuchrgb4) = buffer tstuchrg:before-rowid
no-error.
if available tstuchrgb4 then do:
  buffer tstuchrgb4:reject-changes() no-error.
  b2:refresh().
  end. /* available tstuchrgb4 */
  end. /* available tstuchrg */
  end. /* not rstable */
  end. /* choice */
end. /* choose of brallow */

on choose of brrow
do:
  def var msg as char.
  if rstable
  then
  msg = "Do you want to reject this Student's row changes?".
  else
  msg = "Do you want to reject this Student Charges's row changes?".
  message msg skip
    view-as alert-box
    question buttons yes-no update choice.
  if choice then do:
    if rstable then do:
      find tstudentb4 where rowid(tstudentb4) = buffer tstudent:before-rowid
      no-error.
      if available tstudentb4 then do:
        buffer tstudentb4:reject-row-changes() no-error.
        b1:refresh().
        end. /* available tstudentb4 */
      end. /* rstable */
    else do:
      find tstuchrgb4 where rowid(tstuchrgb4) = buffer tstuchrg:before-rowid
      no-error.
      if available tstuchrgb4 then do:
        buffer tstuchrgb4:reject-row-changes() no-error.
        b2:refresh().
        end. /* available tstuchrgb4 */
      end. /* not rstable */
    end. /* choice */
  end. /* choose of brrow */

on choose of brall
do:
  message "Do you want to reject all changes?" view-as alert-box
    question buttons yes-no update choice.
  if choice then do:
    dataset dsstuchrg:reject-changes() no-error.
    b1:refresh().
    b2:refresh().
  end. /* choice */
end.

on value-changed of b1
do:
  open query qtstuchrg for each tstuchrg of tstudent.
  apply "value-changed" to b2.
end.

on value-changed of b2
do:
  find tchange of tstuchrg NO-ERROR.

```



Climb Aboard The ProDataSet Train

```
IF AVAILABLE tcharge THEN
reposition qtcharge to rowid rowid(tcharge).
end.

on choose of bdone apply "close" to this-procedure.

on close of this-procedure
do:
  if this-procedure:persistent then delete procedure this-procedure.
end.

default-window:width = 100.

buffer tstuchrg:set-callback-procedure
  ("after-fill", "poststuchrgFill", THIS-PROCEDURE).

buffer tstudent:attach-data-source(data-source srcstudent:handle,"").
buffer tstuchrg:attach-data-source(data-source srcstuchrg:handle,"").
buffer tcharge:attach-data-source(data-source srccharge:handle,"").

query qstudent:query-prepare("for each student
                             where stcode = 'IL'
                             and syear = 2007").

dataset dsstuchrg:fill().

buffer tstudent:detach-data-source().
buffer tstuchrg:detach-data-source().
buffer tcharge:detach-data-source().

display rstable with frame f1.

enable all with frame f1.

open query qtstudent for each tstudent.
open query qtstuchrg for each tstuchrg of tstudent.
open query qtcharge for each tcharge.

temp-table tstudent:tracking-changes = yes.
temp-table tstuchrg:tracking-changes = yes.

wait-for close of this-procedure.

procedure poststuchrgfill:
  define input parameter dataset for dsstuchrg.
  for each tstuchrg of tstudent:
    accumulate chargeamt (total).
  end.
  assign chargetot = accum total chargeamt.
end.
```



Maintaining ProDataSet Changes

- In **pds6.p**, the following temp-table definition defines a **BEFORE-TABLE**:

```
define temp-table tstuchrg no-undo like stuchrg
before-table tstuchrgb4.
```

- The **BEFORE-TABLE** phrase allows the developer access to the original values in the table before the records were changed, and before the new values are accepted.
- If there is a **BEFORE-TABLE**, then logically there must be an **AFTER-TABLE**. The name of the **AFTER-TABLE** is simply the name defined for the temp-table.
- Both the **BEFORE-TABLE** and the **AFTER-TABLE** contain a field called the **ROW-STATE**.
- The **ROW-STATE** values are as follows:

Value	Keyword
0	ROW-UNMODIFIED
1	ROW-DELETED
2	ROW-MODIFIED
3	ROW-CREATED

- The above four keywords are actually functions that return the corresponding integers.



Maintaining ProDataSet Changes

tstudent after-buffer

studentid	sfirstname	slastname	gpa	ROW-STATE
1200	Barb	Wilson	3.2	ROW-MODIFIED
3002	Joe	Hughes	3.1	ROW-CREATED

tstudentb4 before-buffer

studentid	sfirstname	slastname	gpa	ROW-STATE
1200	Barb	Miller	3.0	ROW-MODIFIED
1205	Ted	Hamilton	3.5	ROW-DELETED



Maintaining ProDataSet Changes

- This field can be accessed in two ways:

ROW-STATE attribute

ROW-STATE function

- Use the **ROW-STATE** attribute on either the **BEFORE-TABLE** or **AFTER-TABLE**.
- For example, either of these forms are equivalent:

```
if buffer tstudentb4:ROW-STATE = 2
if buffer tstudent:ROW-STATE = ROW-MODIFIED
```

- Use the **ROW-STATE** function in the where clause when reading the **BEFORE-TABLE** buffer.
- For example, either of these forms are equivalent:

```
for each tstudentb4 where ROW-STATE(tstudentb4) = 1
for each tstudentb4 where ROW-STATE(tstudentb4) = ROW-DELETED
```

- The **BEFORE-TABLE** only contains records that are created, modified or deleted. As mentioned previously, the old data is stored in these records.



Maintaining ProDataSet Changes

Student#	First Name	Last Name	GPA	Phone	Total Charges
000078	Eve	Hoffman	3.70	(312) 930-1926	37,282.24
000253	Enid	Bilocerkowyc	2.56	(312) 499-7269	37,220.00
000521	Edward	Feinstein	2.25	(312) 571-1371	36,792.99
001883	Ahmad	Jordan	2.95	(312) 938-3077	36,827.00

Charge No.	chargeDate	chargeCode	Amount	chargeCode	Description
042798	08/29/00	Book	\$325.00	book	Book Charge
042805	12/27/00	Book	\$450.00	finance	Finance Charge
042812	04/02/01	Book	\$600.00	food	Food Charge
042819	08/28/01	Book	\$325.00	other	Other Charge
042826	12/27/01	Book	\$500.00	room	Room Charge
042833	04/02/02	Book	\$575.00	tax	tax charge

Reject: Student Student Charge

Student#	First Name	Last Name	GPA	Phone	Total Charges
000078	Eve	Hoffman	3.70	(312) 930-1926	37,282.24
000253	Enid	Bilocerkowyc	2.56	(312) 499-7269	37,220.00
000521	Edward	Feinstein	2.25	(312) 571-1371	36,792.99
001883	Ahmad	Jordan	2.95	(312) 938-3077	36,827.00

Charge No.	chargeDate	chargeCode	Amount	chargeCode	Description
042798	08/29/00	Book	\$325.00	book	Book Charge
042805	12/27/00	Book	\$450.00	finance	Finance Charge
042812	04/02/01	Book	\$600.00	food	Food Charge
042819	08/28/01	Book	\$325.00	other	Other Charge
042826	12/27/01	Book	\$500.00	room	Room Charge
042833	04/02/02	Book	\$575.00	tax	tax charge

Reject: Student Student Charge

Question (Press HELP to view stack trace) ✕

Do you want to reject this Student Charges's row changes?



Maintaining ProDataSet Changes

- After a ProDataSet is filled, it is likely that changes to the data will need to be tracked.
- This is accomplished by setting the **TRACKING-CHANGES** attribute for a particular temp-table to be tracked.
- Only changes made to a temp-table while **TRACKING-CHANGES** is yes will be tracked by Progress.
- In **pds6.p**, **TRACKING-CHANGES** is set to yes for both the **tstudent** and **tstuchrg** temp-tables AFTER the **FILL** method is executed.
- Setting **TRACKING-CHANGES** to yes before doing a **FILL** method will result in a run-time error.
- Changes may be made to the **gpa** and **phone** fields in the **tstudent** temp-table directly in the first browse.
- Changes may be made to the **chargedate**, **chargecode** and **chargeamt** fields in the **tstuchrg** temp-table directly in the second browse.
- These changes are recorded in either the **tstudentb4** or the **tstuchrgb4** before-tables respectively.
- The user may select a changed student charge row in a browse and then select the **Reject Student Charge Row** button.
- If the user answers yes to the alert-box, then that specific student charge row is undone, and the old values are re-displayed in the browse.
- This trigger uses the **REJECT-ROW-CHANGES** method on the temp-table **BEFORE-TABLE** buffer handle.
 - The **BEFORE-TABLE** buffer is not available by default. However, it is easy to find the corresponding **BEFORE-TABLE** buffer for the current temp-table buffer in the browse by using the **BEFORE-ROWID** attribute on the after-table buffer in the **FIND** statement.
 - Conversely, there is also an **AFTER-ROWID** attribute on the **BEFORE-TABLE** buffer which can also be used to locate records in the other direction.



Maintaining ProDataSet Changes

- The **REJECT-CHANGES** method may be used on either the temp-table **BEFORE-TABLE** buffer or the DataSet handle.
 - If **REJECT-CHANGES** is used on the **BEFORE-TABLE** buffer, all changes made to that temp-table will be rejected. This is the result of selecting "Reject All Student Charge Rows" in pds6.p. This will undo all changes to the tstuchrg temp-table. If the student radio-button is selected, then this will undo all tstudent rows.
 - If **REJECT-CHANGES** is used on the DataSet handle, all changes to all temp-tables in the DataSet will be undone. This is the result from selecting the "Reject All" button in pds6.p.
 - The **BEFORE-TABLE** values are put back into the after-table records. After that, then all **BEFORE-TABLE** records are cleared from each **BEFORE-TABLE** in this case. Also, the **ROW-STATE** attributes in each after-table are zeroed out and the **BEFORE-ROWID** attribute has be reset to ?.
- In all of these triggers, the **REFRESH** method is used on the appropriate browse to re-display the old values back to the user.



Maintaining ProDataSet Changes

pds7.p

```

/* pds7.p dataset example seven - accept changes and row changes
   - but don't update database */
.
.
.
on choose of bsaveall
do:
  message "Do you want to accept all changes?" view-as alert-box
    question buttons yes-no update choice.
  if choice then do:
    dataset dsstuchrg:accept-changes() no-error.
  end. /* choice */
end.

on choose of bsaveallrow
do:
  def var msg as char.
  if rstable
  then
  msg = "Do you want to accept all of the Student row changes?".
  else
  msg = "Do you want to accept all of the Student Charges row changes?".
  message msg skip
    view-as alert-box
    question buttons yes-no update choice.
  if choice then do:
    if rstable then do:
      /* need to find modified tstudent, since current row may not be
         changed. */
      find first tstudent where row-state(tstudent) gt 0 no-error.
      if available tstudent then do:
        buffer tstudent:accept-changes() no-error.
      end. /* available tstudent */
    end. /* rstable */
    else do:
      /* need to find modified tstuchrg, since current row may not be
         changed. */
      find first tstuchrg where row-state(tstuchrg) gt 0 no-error.
      if available tstuchrg then do:
        buffer tstuchrg:accept-changes() no-error.
      end. /* available tstuchrg */
    end. /* not rstable */
  end. /* choice */
end. /* on choose of bsaveallrow */

on choose of bsaverow
do:
  def var msg as char.
  if rstable
  then
  msg = "Do you want to accept this Student's row changes?".
  else
  msg = "Do you want to accept this Student Charges's row changes?".
  message msg skip
    view-as alert-box
    question buttons yes-no update choice.

```



Climb Aboard The ProDataSet Train

```

if choice then do:
  if rstable then do:
    find tstudentb4 where rowid(tstudentb4) = buffer tstudent:before-rowid
    no-error.
    if available tstudentb4 then do:
      buffer tstudentb4:accept-row-changes() no-error.
    end. /* available tstudentb4 */
  end. /* rstable */
else do:
  find tstuchrgb4 where rowid(tstuchrgb4) = buffer tstuchrg:before-rowid
  no-error.
  if available tstuchrgb4 then do:
    buffer tstuchrgb4:accept-row-changes() no-error.
  end. /* available tstuchrgb4 */
end. /* not rstable */
end. /* choice */
end. /* on choose of bsaverow */

```

Student#	First Name	Last Name	GPA	Phone	Total Charges
000078	Eve	Hoffman	3.70	(312) 930-1926	37,282.24
000253	Enid	Bilocerkowyc	2.56	(312) 499-7269	37,220.00
000521	Edward	Feinstein	2.25	(312) 571-1371	36,792.99
001883	Ahmad	Jordan	2.95	(312) 938-3077	36,827.00

Charge No.	chargeDate	chargeCode	Amount	chargeCode	Description
042798	08/29/00	Book	\$325.00	book	Book Charge
042805	12/27/00	Book	\$450.00	finance	Finance Charge
042812	04/02/01	Book	\$600.00	food	Food Charge
042819	08/28/01	Book	\$325.00	other	Other Charge
042826	12/27/01	Book	\$500.00	room	Room Charge
042833	04/02/02	Book	\$575.00	tax	tax charge

Accept/Reject: Student Student Charge

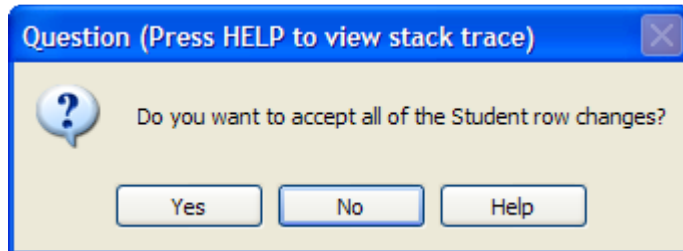
Student#	First Name	Last Name	GPA	Phone	Total Charges
000078	Eve	Hoffman	3.70	(312) 930-1926	37,282.24
000253	Enid	Bilocerkowyc	2.56	(312) 499-7269	37,220.00
000521	Edward	Feinstein	2.25	(312) 571-1371	36,792.99
001883	Ahmad	Jordan	2.95	(312) 938-3077	36,827.00

Charge No.	chargeDate	chargeCode	Amount	chargeCode	Description
042798	08/29/00	Book	\$325.00	book	Book Charge
042805	12/27/00	Book	\$450.00	finance	Finance Charge
042812	04/02/01	Book	\$600.00	food	Food Charge
042819	08/28/01	Book	\$325.00	other	Other Charge
042826	12/27/01	Book	\$500.00	room	Room Charge
042833	04/02/02	Book	\$575.00	tax	tax charge

Accept/Reject: Student Student Charge



Maintaining ProDataSet Changes



- In pds7.p, we introduce the **ACCEPT-CHANGES** and **ACCEPT-ROW-CHANGES** methods.
 - These two methods work similarly to the **REJECT-CHANGES** and **REJECT-ROW-CHANGES** of **pds6.p** except the changes are accepted and cannot be rejected or undone after they are accepted. As before, the **BEFORE-TABLE** records are cleared out, the **ROW-STATE** values are zeroed out in the after-table records and the **BEFORE-ROWID** attributes are set to ?.
 - Contrary to Progress' documentation, the **TRACKING-CHANGES** attribute doesn't have to be set to no before using **REJECT-CHANGES**, **ACCEPT-CHANGES**, **REJECT-ROW-CHANGES** and **ACCEPT-ROW-CHANGES**.
 - Both **REJECT-ROW-CHANGES** and **ACCEPT-ROW-CHANGES** must use the **BEFORE-TABLE** buffer, whereas **REJECT-CHANGES** and **ACCEPT-CHANGES** may use either the **BEFORE-TABLE** or after-table buffers.
 - Keep in mind that **ACCEPT-CHANGES** and **ACCEPT-ROW-CHANGES** do not update the database. They only prevent the temp-table records from being undone.



Saving Data from a ProDataSet to the Database

```

/* pds8.p dataset example eight - accept changes and row changes
   - and update database */

on choose of bsaveall
do:
  message "Do you want to save all changes?" view-as alert-box
    question buttons yes-no update choice.
  if choice then do transaction:
    buffer tstudent:attach-data-source(data-source srcstudent:handle,"").
    buffer tstuchrg:attach-data-source(data-source srcstuchrg:handle,"").
    for each tstudentb4 where row-state(tstudentb4) > 0:
      buffer tstudentb4:save-row-changes() no-error.
      if buffer tstudentb4:error
      then do:
        buffer tstudentb4:reject-row-changes() no-error.
        b1:refresh().
        buffer tstudentb4:error = no.
      end. /* buffer tstudentb4:error */
      else buffer tstudentb4:accept-row-changes() no-error.
    end. /* for each */
    for each tstuchrgb4 where row-state(tstuchrgb4) > 0:
      buffer tstuchrgb4:save-row-changes() no-error.
      if buffer tstuchrgb4:error
      then do:
        buffer tstuchrgb4:reject-row-changes() no-error.
        b2:refresh().
        buffer tstuchrgb4:error = no.
      end. /* buffer tstuchrgb4:error */
      else buffer tstuchrgb4:accept-row-changes() no-error.
    end. /* for each */
    buffer tstudent:detach-data-source().
    buffer tstuchrg:detach-data-source().
  end. /* choice */
end. /* on choose of bsaveall */

on choose of bsaveallrow
do:
  def var msg as char.
  if rstable
  then
  msg = "Do you want to save all of the Student row changes?".
  else
  msg = "Do you want to save all of the Student Changes row changes?".
  message msg skip
    view-as alert-box
    question buttons yes-no update choice.
  if choice then do:
    if rstable then do:
      buffer tstudent:attach-data-source(data-source srcstudent:handle,"").
      for each tstudentb4 where row-state(tstudentb4) > 0 transaction:
        buffer tstudentb4:save-row-changes() no-error.
        if buffer tstudentb4:error
        then do:
          buffer tstudentb4:reject-row-changes() no-error.
          b1:refresh().
          buffer tstudentb4:error = no.
        end. /* buffer tstudentb4:error */
        else buffer tstudentb4:accept-row-changes() no-error.
      end. /* for each */
      buffer tstudent:detach-data-source().
    end. /* rstable */
    else do:
      buffer tstuchrg:attach-data-source(data-source srcstuchrg:handle,"").
      for each tstuchrgb4 where row-state(tstuchrgb4) > 0 transaction:
        buffer tstuchrgb4:save-row-changes() no-error.

```




```

    if buffer tstuchrgb4:error
    then do:
        buffer tstuchrgb4:reject-row-changes() no-error.
        b2:refresh().
        buffer tstuchrgb4:error = no.
    end. /* buffer tstuchrgb4:error */
    else buffer tstuchrgb4:accept-row-changes() no-error.
    end. /* for each */
    buffer tstuchrg:detach-data-source().
end. /* not rstable */
end. /* choice */
end. /* on choose of bsaveallrow */

on choose of bsaverow
do:
    def var msg as char.
    if rstable
    then
        msg = "Do you want to save this Student's row changes?".
    else
        msg = "Do you want to save this Student Charges's row changes?".
    message msg skip
        view-as alert-box
        question buttons yes-no update choice.
    if choice then do:
        if rstable then do:
            find tstudentb4 where rowid(tstudentb4) = buffer tstudent:before-rowid
            no-error.
            if available tstudentb4 then do:
                buffer tstudent:attach-data-source(data-source srcstudent:handle,"").
                buffer tstudentb4:save-row-changes() no-error.
                if buffer tstudentb4:error
                then do:
                    buffer tstudentb4:reject-row-changes() no-error.
                    b1:refresh().
                    buffer tstudentb4:error = no.
                end. /* buffer tstudentb4:error */
                else buffer tstudentb4:accept-row-changes() no-error.
                buffer tstudent:detach-data-source().
            end. /* available tstudentb4 */
        end. /* rstable */
    else do:
        find tstuchrgb4 where rowid(tstuchrgb4) = buffer tstuchrg:before-rowid
        no-error.
        if available tstuchrgb4 then do:
            buffer tstuchrg:attach-data-source(data-source srcstuchrg:handle,"").
            buffer tstuchrgb4:save-row-changes() no-error.
            if buffer tstuchrgb4:error
            then do:
                buffer tstuchrgb4:reject-row-changes() no-error.
                b2:refresh().
                buffer tstuchrgb4:error = no.
            end. /* buffer tstuchrgb4:error */
            else buffer tstuchrgb4:accept-row-changes() no-error.
            buffer tstuchrg:detach-data-source().
        end. /* available tstuchrgb4 */
    end. /* not rstable */
end. /* choice */
end. /* on choose of bsaverow */

```



Saving Data from a ProDataSet to the Database

In **pds8.p**, we take it one step further by updating the database from the ProDataSet.

Here are the steps that need to be performed:

1. Find the associated **BEFORE-TABLE** buffer to the temp-table buffer.
2. If the **BEFORE-TABLE** buffer is available, then attach the data-source from the temp-table to the database table.
3. Execute the **SAVE-ROW-CHANGES** method on the **BEFORE-TABLE** buffer.
4. Assuming no errors execute the **ACCEPT-ROW-CHANGES** method on the **BEFORE-TABLE** buffer.
5. Detach the data-source from the temp-table to the database table.

Syntax

SAVE-ROW-CHANGES method

```
handle:SAVE-ROW-CHANGES( [ buffer-index | buffer-name [ , skip-list [ , no-lobs ] ] ] )
```

handle is the before-table buffer in a ProDataSet



Saving Data from a ProDataSet to the Database

SAVE-ROW-CHANGES takes the following parameters:

handle is the before-image table buffer in a ProDataSet

buffer-index is an INTEGER expression that specifies the index of the buffer in the data source's buffer list. The default value is 1.

buffer-name is a CHARACTER expression that evaluates to the name of the buffer in the data source.

skip-list is an optional character expression that evaluates to a comma-separated list of field names for fields that should not be assigned after a new row is created (that is, fields to skip). For example, a key field or other fields assigned a value by a CREATE database trigger.

no-lobs is A logical expression indicating whether to ignore BLOB and CLOB fields in the save operation. If TRUE, BLOB and CLOB fields are ignored during the save operation. If FALSE, BLOB and CLOB fields are saved along with the other fields. The default value is FALSE (that is, BLOB and CLOB fields are included in the save operation).

Please note that there is no **SAVE-CHANGES** method for the entire temp-table or DataSet. Each individual record must be dealt with the **SAVE-ROW-CHANGES** method.



Saving Data from a ProDataSet to the Database

- The **SAVE-ROW-CHANGES** methods gives the developer maximum flexibility in grouping records into transactions.
- The **SAVE-ROW-CHANGES** goes through the following steps for a modified row:
 1. Find the corresponding database record exclusive-lock based on its key. If the record is not available, the method retries every two seconds up to ten times, and then returns an error if the record is still not available.
 2. Compares the **BEFORE-TABLE** table buffer to the database buffer to see whether data has changed since it has been read. More on this in **pds10.p**.
 3. Buffer-copies changed fields in the corresponding after-table buffer to the corresponding database buffer fields. This uses the same field mapping used to **FILL** the table (as defined in the **ATTACH-DATA-SOURCE** method).
 4. Validates the updated record to force any **WRITE** or **ASSIGN** triggers to fire.
 5. Sets the **ERROR** logical attribute in the after-table row as well as in its temp-table and in the DataSet if any errors resulted from the attempted update, such as duplicate unique keys.
 6. Repopulate the after-table record from the database record, to catch any changes made by either event procedure code or trigger procedure code.
 7. Released the after-table and database table records.
- For a newly created row, **SAVE-ROW-CHANGES** creates the record in the database and buffer-copies all data table buffer fields except any create-field to the database buffer specified in the **SAVE-ROW-CHANGES** method. It then executes steps 4-7 above.
- For a delete row, **SAVE-ROW-CHANGES** deletes the corresponding row from the data-source, based on the record's keys.
- Note that **SAVE-ROW-CHANGES** can create or delete only a single database buffer at a time, not both sides of a one-to-one join.
- Even though the **SAVE-ROW-CHANGES** method updates the database, it does not count as a reference for automatically making a block a transaction block. Try to think of the **SAVE-ROW-CHANGES** method as a self-contained procedure block, where the transaction is committed at the end of the **SAVE-ROW-CHANGES** method statement.
- This means that the developer must specify the **TRANSACTION** keyword on the block that is desired for the transaction.



Saving Data from a ProDataSet to the Database

pds9.p

```
/* pds9.p dataset example nine - one table update dataset */

on choose of badd
do:
  def var saverowid as rowid.
  def var i as int no-undo.
  do with frame studadd view-as dialog-box 1 column title "Student Add":
    form   tstudent.studentid label "Student#"
          tstudent.sfirstname
          tstudent.slastname
          tstudent.city
          tstudent.stcode
          tstudent.postalcode
          tstudent.phone
          tstudent.gpa
          tstudent.syear.
  create tstudent.
  assign saverowid = rowid(tstudent).
  display tstudent.studentid.
  update  tstudent.sfirstname
          tstudent.slastname
          tstudent.city
          tstudent.stcode
          tstudent.postalcode
          tstudent.phone
          tstudent.gpa
          tstudent.syear.

  buffer tstudent:attach-data-source(data-source srcstudent:handle,"").

  find tstudentb4 where rowid(tstudentb4) = buffer tstudent:before-rowid
  no-error.
  if available tstudentb4 then do:
    buffer tstudentb4:save-row-changes("student","studentid") no-error.
    if buffer tstudentb4:error
    then do:
      buffer tstudentb4:reject-row-changes() no-error.
      buffer tstudentb4:error = no no-error.
      b1:refresh().
    end. /* buffer tstudentb4:error */
  else do:
    buffer tstudentb4:accept-row-changes() no-error.
    open query qtstudent for each tstudent.
    b1:set-repositioned-row(3,"conditional").
    reposition qtstudent to rowid saverowid no-error.
    run dispstud.
  end. /* else do */
end. /* available tstudentb4 */
end. /* with view-as dialog-box 1 column title "Student Add" */
end. /* on choose of badd */
```



Saving Data from a ProDataSet to the Database

Procedure Editor - Run

StudentID	First Name	Last Name	GPA	Phone
000078	Eve	Hoffman	3.75	(312) 930-1931
000253	Enid	Bilocerkowyc	2.57	(312) 499-7260
000521	Edward	Feinstein	2.23	(312) 571-1370
001883	Ahmad	Jordan	2.96	(312) 938-3070
001930	Jeff	Teague	2.40	(312) 929-9985
001991	Harvey	Pfeifer	2.50	(312) 265-4400

StudentID: 000078
 Name: Eve Hoffman
 City,St,Zip: Chicago IL 60683
 Phone: (312) 930-1931

Add First Next Prev Last Full Done

Student Add

Student#: 000000

First Name: Jack

Last Name: Black

city: Hinsdale

State/Province: IL

Zip Code: 60512

Phone: (708) 812-2345

GPA: 2.75

Year: 2002

StudentID	First Name	Last Name	GPA	Phone
002552	Enid	Parsons	3.19	(312) 229-6360
002640	Byron	Payne	1.91	(312) 236-6605
003001	Jack	Black	2.75	(708) 812-2345

StudentID: 003001
 Name: Jack Black
 City,St,Zip: Hinsdale IL 60512
 Phone: (708) 812-2345

Add First Next Prev Last Full Done



Saving Data from a ProDataSet to the Database

- In **pds9.p**, we will show how you can add temp-table records in a ProDataSet and then use **SAVE-ROW-CHANGES** to update the database.
 - The **SAVE-ROW-CHANGES** method will either add, change or delete the database record based upon whether the temp-table is added, changed or deleted.
 - That is why you need to use the **BEFORE-TABLE** temp-table handle for the **SAVE-ROW-CHANGES** method, since it contains the deleted temp-table record.
 - To prevent the studentid from being updated from the temp-table in the DataSet, we supply two arguments, the buffer-name and the skip-list in the **SAVE-ROW-CHANGES** method.
 - The buffer-name, student, specifies the buffer that corresponds to a buffer in the **ATTACH-DATA-SOURCE** method.
 - The skip-list, "studentid" in this case, is a comma-separated list of field names for fields that should not be assigned after a new record is created.
 - Since the studentid value is generated from studentseq sequence in the student create trigger, we don't want to overwrite this value from the DataSet. Therefore, studentid is specified in the skip list.
 - Please note that the skip list can also be specified for updating records.
 - Other fields such as calculated fields could also be specified for updating a record, such as student gpa and address-agg fields.



Saving Data from a ProDataSet to the Database

pds10.p

```

/* pds10.p dataset example ten - test conflicts and error flags */

def var choice as log no-undo.
def var i      as int no-undo.
def var swstr  as char no-undo.
def var pdflag as log no-undo label "Prefer Dataset" view-as toggle-box.
def var mbfflag as log no-undo label "Merge by Field" view-as toggle-box.
.
.
.
on value-changed of pdflag
do:
  assign pdflag
  data-source srcstudent:prefer-dataset = pdflag.
end.

on value-changed of mbfflag
do:
  assign mbfflag
  data-source srcstudent:merge-by-field = mbfflag.
end.

.
.
.
on choose of bsaverow
do:
  def var msg as char.
  if rstable
  then
  msg = "Do you want to save this Student's row changes?".
  else
  msg = "Do you want to save this Student Charges's row changes?".
  message msg skip
  view-as alert-box
  question buttons yes-no update choice.
  if choice then do:
  if rstable then do:
  find tstudentb4 where rowid(tstudentb4) = buffer tstudent:before-rowid
  no-error.
  if available tstudentb4 then do transaction:
  buffer tstudent:attach-data-source(data-source srcstudent:handle,"").
  buffer tstudentb4:save-row-changes() no-error.
message "Data source modified after save-row-changes?"
  buffer tstudentb4:data-source-modified skip
  "Prefer dataset:" data-source srcstudent:prefer-dataset skip
  "Merge By Field:" data-source srcstudent:merge-by-field skip
  "Dataset Error?" dataset dsstuchrg:error skip
  "Temp-table Student Error?" temp-table tstudent:error skip
  "Student Buffer Error?" buffer tstudentb4:error skip
  "Dataset Rejected?" dataset dsstuchrg:Rejected skip
  "Temp-table Student Rejected?" temp-table tstudent:Rejected skip
  "Student Buffer Rejected?" buffer tstudentb4:rejected skip
  view-as alert-box.
  /* also check datarow modified to see if another field was changed. */
  /* need to get that value into the temp-table */

```




```

/* refresh temp-table if conflict */
if buffer tstudentb4:error
or buffer tstudentb4:data-source-modified
then do:
  swstr = data-source srcstudent:save-where-string(1).
  message "Save Where string:" swstr skip
  "Student Buffer Error?" buffer tstudentb4:error skip
  "Data source modified after save-row-changes?"
  buffer tstudentb4:data-source-modified skip
  view-as alert-box.
  buffer student:find-first(swstr).
  find tstudent where rowid(tstudent) = buffer tstudentb4:after-rowid
  no-error.
  /* only changed fields in Student buffer will be copied */
  /* don't need this statement since save-row-changes refreshes step 6 pg 6-67 */
  buffer tstudent:buffer-copy(buffer student:handle).
  b1:refresh().
end. /* dataset dsstuchrg:error */
/* need to release student because student record scope is
containing procedure block, otherwise student will still
be share-locked at the end of the transaction. */
  release student no-error .
  buffer tstudent:detach-data-source().
  buffer tstudentb4:accept-row-changes() no-error.
end. /* available tstudentb4 */
end. /* rstable */
else do:
  find tstuchrgb4 where rowid(tstuchrgb4) = buffer tstuchrg:before-rowid
  no-error.
  if available tstuchrgb4 then do transaction:
    buffer tstuchrg:attach-data-source(data-source srcstuchrg:handle,"").
    buffer tstuchrgb4:save-row-changes() no-error.
    if buffer tstuchrgb4:error
    then do:
      buffer tstuchrgb4:reject-row-changes() no-error.
      b2:refresh().
    end. /* buffer tstuchrgb4:error */
    else buffer tstuchrgb4:accept-row-changes() no-error.
    release stuchrg no-error.
    buffer tstuchrg:detach-data-source().
  end. /* available tstuchrgb4 */
end. /* not rstable */
end. /* choice */
end. /* on choose of bsaverow */
.
.
.
assign pdfflag = data-source srcstudent:prefer-dataset
mbffflag = data-source srcstudent:merge-by-field.

display rstable pdfflag mbffflag with frame f1.

```



Saving Data from a ProDataSet to the Database

- In pds10.p, we will explore how Progress supports change conflicts with ProDataSets.
 - Because a given user is updating a temp-table in a DataSet, it is possible that another user could have changed the same record before the given user saves the row back to the database. This doesn't violate any Progress locking rules since the other user completed their transaction first before the given user tried to update the same record.
 - However, the before buffer for a given user will now be different from the current record in the database due to the other user's change. The question now arises what to do with the update? Do we disallow the update since the given user could be overriding changes the other user made, or do we allow the DataSet to take precedence and overwrite the previous changes?
 - There is also a third option which says allow the given user to update only those fields that were changed as long as those fields weren't changed by the other user.
- There are two data-source attributes that help answer these questions:

Attribute	Default Value
PREFER-DATASET	NO
MERGE-BY-FIELD	YES

- If the **PREFER-DATASET** attribute is false (no) for the data-source, then the comparison is made between the database record and the **BEFORE-TABLE** temp-table row in the DataSet. If there is any conflict, then the change is rejected and the **ERROR** attribute is set.
- If the **PREFER-DATASET** is true, then the check is not made and the changes are written to the data-source from the ProDataSet, without regard to any changes from another user that may have been made.
- If the **MERGE-BY-FIELD** attribute is true (yes) for the data-source, then the comparison is made on individual fields between the database record and the **BEFORE-TABLE** temp-table row in the DataSet. A conflict will only exist if the same field has been changed by another user and the ProDataSet. In this case, the **ERROR** attribute is set.
- If the **MERGE-BY-FIELD** attribute is false, then if another user changed any field in the same record, then the DataSet changes will be completely rejected and the **ERROR** attribute is set.



Saving Data from a ProDataSet to the Database

For example, the first user changes the gpa field in the student record and then the second user changes the phone number for the same student record.

If **MERGE-BY-FIELD** is true, then there is no conflict when the second user updates the database so field changes are kept.

If **MERGE-BY-FIELD** is false, then any change to any field by the first user results in a conflict.

Here is a table summarizing the different scenarios for the **PREFER-DATASET** and **MERGE-BY-FIELD** data-source attributes. This assumes that a change was made by the first user, causing the data-source-modified attribute is true:

PREFER-DATASET	MERGE-BY-FIELD	If Field Conflict	What is copied to DB
FALSE	TRUE	Data-Source wins	Only non-conflicting fields
FALSE	FALSE	Data-Source wins	Nothing
TRUE	FALSE	DataSet wins	All temp-table fields
TRUE	TRUE	DataSet wins	Changed Fields Only



Saving Data from a ProDataSet to the Database

Figure pds10-1

User 1

Student#	First Name	Last Name	GPA	Phone	Total Charges
000206	Derwood	Glass	2.95	(312) 804-7420	27,646.00
001956	Diane	Huber	3.75	(312) 519-8147	25,371.00
002037	Gladys	Larson	2.83	(312) 534-0669	25,362.00
002819	Quincy	Jacobson	2.80	(312) 765-0009	25,020.00

Charge No.	chargeDate	chargeCode	Amount	chargeCode	Descri
011556	08/28/06	Book	\$326.00	book	Book (
011564	12/27/06	Book	\$700.00	finance	Financ
011572	04/02/07	Book	\$669.00	food	Food C
011580	08/28/07	Book	\$325.00	other	Other I
011588	12/27/07	Book	\$450.00	room	Room
011596	04/02/08	Book	\$575.00	tax	Tax Cl

Save/Reject: Student Student Charge

Prefer Dataset Merge by Field

User 2

Student#	First Name	Last Name	GPA	Phone	Total Char
000206	Derwood	Glass	2.85	(312) 804-7421	27,646.00
001956	Diane	Huber	3.75	(312) 519-8147	25,371.00
002037	Gladys	Larson	2.83	(312) 534-0669	25,362.00
002819	Quincy	Jacobson	2.80	(312) 765-0009	25,020.00

Charge No.	chargeDate	chargeCode	Amount	chargeCode	D
011556	08/28/06	Book	\$326.00	book	B
011564	12/27/06	Book	\$700.00	finance	F
011572	04/02/07	Book	\$669.00	food	F
011580	08/28/07	Book	\$325.00	other	O
011588	12/27/07	Book	\$450.00	room	R
011596	04/02/08	Book	\$575.00	tax	T

Save/Reject: Student Student Charge

Prefer Dataset Merge by Field

Message (Press HELP to view stack trace)

Data source modified after save-row-changes? yes

Prefer dataset: no

Merge By Field: yes

Dataset Error? no

Temp-table Student Error? no

Student Buffer Error? no

Dataset Rejected? no

Temp-table Student Rejected? no

Student Buffer Rejected? no

Message (Press HELP to view stack trace)

Save Where string: WHERE student.StudentID=tstudentb4.StudentID

Student Buffer Error? no

Data source modified after save-row-changes? yes



Saving Data from a ProDataSet to the Database

- In figure pds10-1, User 1 has changed the student gpa and User 2 is changing the phone number. When User 2 goes to save their changes, no errors attributes are set since there is not a conflict with merge-by-field set to true. The result is that User 2 will show the same gpa as User 1 when the update is completed. (see pds10-2 below)

pds10-2

User 2

Student#	First Name	Last Name	GPA	Phone	Total Charges
000206	Derwood	Glass	2.95	(312) 804-7421	27,646.00
001956	Diane	Huber	3.75	(312) 519-8147	25,371.00
002037	Gladys	Larson	2.83	(312) 534-0669	25,362.00
002819	Quincy	Jacobson	2.80	(312) 765-0009	25,020.00

Charge No.	chargeDate	chargeCode	Amount	chargeCode	Descri
011556	08/28/06	Book	\$326.00	book	Book (
011564	12/27/06	Book	\$700.00	finance	Financ
011572	04/02/07	Book	\$669.00	food	Food C
011580	08/28/07	Book	\$325.00	other	Other I
011588	12/27/07	Book	\$450.00	room	Room
011596	04/02/08	Book	\$575.00	tax	Tax Cf

Save/Reject: Student Student Charge

Prefer Dataset Merge by Field



Saving Data from a ProDataSet to the Database

pds10-4

User 1

Student#	First Name	Last Name	GPA	Phone
000206	Derwood	Glass	2.85	(312) 804-7421
001956	Diane	Huber	3.75	(312) 519-8147
002037	Gladys	Larson	2.83	(312) 534-0669
002819	Quincy	Jacobson	2.80	(312) 765-0009

Charge No.	chargeDate	chargeCode	Amount	char
011556	08/28/06	Book	\$326.00	book
011564	12/27/06	Book	\$700.00	finar
011572	04/02/07	Book	\$669.00	food
011580	08/28/07	Book	\$325.00	othe
011588	12/27/07	Book	\$450.00	room
011596	04/02/08	Book	\$575.00	tax

Save/Reject: Student Student Charge

Prefer Dataset Merge by Field

User 2

Student#	First Name	Last Name	GPA	Phone
000206	Derwood	Glass	2.95	(312) 804-7420
001956	Diane	Huber	3.75	(312) 519-8147
002037	Gladys	Larson	2.83	(312) 534-0669
002819	Quincy	Jacobson	2.80	(312) 765-0009

Charge No.	chargeDate	chargeCode	Amount	cha
011556	08/28/06	Book	\$326.00	boo
011564	12/27/06	Book	\$700.00	final
011572	04/02/07	Book	\$669.00	fooc
011580	08/28/07	Book	\$325.00	othe
011588	12/27/07	Book	\$450.00	room
011596	04/02/08	Book	\$575.00	tax

Save/Reject: Student Student Charge

Prefer Dataset Merge by Field

Message (Press HELP to view stack trace)

Data source modified after save-row-changes? yes
 Prefer dataset: no
 Merge By Field: no
 Dataset Error? yes
 Temp-table Student Error? yes
 Student Buffer Error? yes
 Dataset Rejected? no
 Temp-table Student Rejected? no
 Student Buffer Rejected? no

User 2

Student#	First Name	Last Name	GPA	Phone
000206	Derwood	Glass	2.95	(312) 804-7421
001956	Diane	Huber	3.75	(312) 519-8147
002037	Gladys	Larson	2.83	(312) 534-0669
002819	Quincy	Jacobson	2.80	(312) 765-0009

Charge No.	chargeDate	chargeCode	Amount	cha
013002	08/28/06	Book	\$325.00	boo
013012	12/27/06	Book	\$500.00	final
013022	04/02/07	Book	\$575.00	fooc
013032	08/28/07	Book	\$325.00	othe
013042	12/27/07	Book	\$450.00	room
013052	04/02/08	Book	\$575.00	tax

Save/Reject: Student Student Charge

Prefer Dataset Merge by Field



Saving Data from a ProDataSet to the Database

- In figure pds10-4, it is the same scenario as before except that the merge-by-field is now false. Even though different fields were changed, the **ERROR** attribute is set to yes and no data is updated.
- If an error occurs during the save-row-changes method, then Progress sets the **ERROR** attribute on the temp-table buffer, the temp-table and the DataSet.
 - Possible reasons for the **ERROR** attribute to be set are unique index violation, a db trigger procedure that returns an error, or if the record was changed by another user.
 - The **ERROR** attribute can also be set by the developer to signal an error condition of any kind. Manually setting the **ERROR** on the buffer doesn't automatically set it on the temp-table and the DataSet.
 - You are free to set the **ERROR** attribute on those levels. The advantage to setting it at a higher level makes it easy to check if there is an error at any of the lower levels.
- There is also an **ERROR-STRING** character attribute on each temp-table, and temp-table buffer. This is never set by Progress and allows the developer to specify the exact description of the error.
- There is also a **REJECTED** logical attribute for the DataSet, each temp-table, and temp-table buffer. This is never set by Progress and allows the developer to specify that a change was not saved to the database because of an error condition. Progress does not set this attribute because it is not possible for Progress to determine the scope of the failed update.
- The **ERROR**, **ERROR-STRING**, **DATA-SOURCE-MODIFIED** and **REJECTED** attributes are all cleared for all tables and rows affected by an **ACCEPT-CHANGES**, **REJECT-CHANGES**, **MERGE-CHANGES**, or **FILL** method. Other methods that clear these attributes are **EMPTY-DATASET** and the **EMPTY-TEMP-TABLE**.
- The **ERROR**, **ERROR-STRING**, **DATA-SOURCE-MODIFIED** and **REJECTED** attributes are all also cleared for the buffer affected by an **ACCEPT-ROW-CHANGES**, **REJECT-ROW-CHANGES** or **MERGE-ROW-CHANGES**.



Saving Data from a ProDataSet to the Database

pds10-2

User 2

Student#	First Name	Last Name	GPA	Phone	Total Charges
000206	Derwood	Glass	2.95	(312) 804-7421	27,646.00
001956	Diane	Huber	3.75	(312) 519-8147	25,371.00
002037	Gladys	Larson	2.83	(312) 534-0669	25,362.00
002819	Quincy	Jacobson	2.80	(312) 765-0009	25,020.00

Charge No.	chargeDate	chargeCode	Amount	chargeCode	Descri
011556	08/28/06	Book	\$326.00	book	Book (
011564	12/27/06	Book	\$700.00	finance	Financ
011572	04/02/07	Book	\$669.00	food	Food C
011580	08/28/07	Book	\$325.00	other	Other I
011588	12/27/07	Book	\$450.00	room	Room
011596	04/02/08	Book	\$575.00	tax	Tax Cl

Save/Reject: Student Student Charge

Prefer Dataset Merge by Field

pds10-3

Message (Press HELP to view stack trace)

Save Where string: WHERE student.StudentID=tstudentb4.StudentID
Student Buffer Error? no
Data source modified after save-row-changes? yes

OK Help



Saving Data from a ProDataSet to the Database

- In pds10-2, the phone number remains the same, yet the data from User 1 is refreshed into the gpa in User 2.
- To pull in the data from User 1's changes into User 2's screen, a few steps must be performed:
 1. Store the **SAVE-WHERE-STRING** for student to the variable swstr.
 2. Use the **FIND-FIRST** method on the student buffer to find the current record
 3. Find the after-image table tstudent temp-table record using the **AFTER-ROWID** attribute of the before table tstudent temp-table record.
 4. Perform a **BUFFER-COPY** method from the student database record to the temp-table tstudent record.
 5. Refresh the b1 browse based on temp-table tstudent.

The **SAVE-WHERE-STRING** is more convenient and more portable than hard coding the **WHERE** clause manually yourself. This method is also handy in dynamic ProDataSets when it is difficult for the developer to hard code the **WHERE** clause for runtime determined tables in a DataSet. see pds10-3

We still need to re-find the tstudent record in step 3, otherwise the **BUFFER-COPY** method in step 4 tries to create a new tstudent record and an error message appears stating that "Student record already exists with studentid 206".



Saving Data from a ProDataSet to the Database

pds10-6

User 1

Student#	First Name	Last Name	GPA	Phone
000206	Derwood	Glass	3.00	(312) 804-7422
001956	Diane	Huber	3.75	(312) 519-8147
002037	Gladys	Larson	2.83	(312) 534-0669
002819	Quincy	Jacobson	2.80	(312) 765-0009

Charge No.	chargeDate	chargeCode	Amount	ch
011556	08/28/06	Book	\$326.00	bo
011564	12/27/06	Book	\$700.00	fin.
011572	04/02/07	Book	\$669.00	foc
011580	08/28/07	Book	\$325.00	otf
011588	12/27/07	Book	\$450.00	roc
011596	04/02/08	Book	\$575.00	tax

Save/Reject: Student Student Charge

Prefer Dataset Merge by Field

User 2

Student#	First Name	Last Name	GPA	Phone
000206	Derwood	Glass	2.95	(312) 804-7425
001956	Diane	Huber	3.75	(312) 519-8147
002037	Gladys	Larson	2.83	(312) 534-0669
002819	Quincy	Jacobson	2.80	(312) 765-0009

Charge No.	chargeDate	chargeCode	Amount	char
011556	08/28/06	Book	\$326.00	bool
011564	12/27/06	Book	\$700.00	finar
011572	04/02/07	Book	\$669.00	fooc
011580	08/28/07	Book	\$325.00	othe
011588	12/27/07	Book	\$450.00	roon
011596	04/02/08	Book	\$575.00	tax

Save/Reject: Student Student Charge

Prefer Dataset Merge by Field

Message (Press HELP to view stack trace) ✕

Data source modified after save-row-changes? no
 Prefer dataset: yes
 Merge By Field: no
 Dataset Error? no
 Temp-table Student Error? no
 Student Buffer Error? no
 Dataset Rejected? no
 Temp-table Student Rejected? no
 Student Buffer Rejected? no

User 2

Student#	First Name	Last Name	GPA	Phone
000206	Derwood	Glass	2.95	(312) 804-7425
001956	Diane	Huber	3.75	(312) 519-8147
002037	Gladys	Larson	2.83	(312) 534-0669
002819	Quincy	Jacobson	2.80	(312) 765-0009

Charge No.	chargeDate	chargeCode	Amount	char
011556	08/28/06	Book	\$326.00	bool
011564	12/27/06	Book	\$700.00	finar
011572	04/02/07	Book	\$669.00	fooc
011580	08/28/07	Book	\$325.00	othe
011588	12/27/07	Book	\$450.00	roon
011596	04/02/08	Book	\$575.00	tax

Save/Reject: Student Student Charge

Prefer Dataset Merge by Field



Saving Data from a ProDataSet to the Database

- In figure pds10-6, the prefer-dataset attribute is set to yes, and merge-by-field is set to no.
- These two settings are probably not desirable since the DataSet is overwriting fields that the user didn't change.
- User 1 changed the gpa to 3.76. User 2 changed the phone number. Notice that no ERROR attributes are set to yes.
- User 2 screen in the lower right in pds10-6 shows that User 1's change to the gpa does not come over since User 2 overwrites the gpa even though User 2 didn't change the gpa.



Saving Data from a ProDataSet to the Database

User 1

Student#	First Name	Last Name	GPA	Phone
000206	Derwood	Glass	3.00	(312) 804-7425
001956	Diane	Huber	3.75	(312) 519-8147
002037	Gladys	Larson	2.83	(312) 534-0669
002819	Quincy	Jacobson	2.80	(312) 765-0009

Charge No.	chargeDate	chargeCode	Amount
011556	08/28/06	Book	\$326.00
011564	12/27/06	Book	\$700.00
011572	04/02/07	Book	\$669.00
011580	08/28/07	Book	\$325.00
011588	12/27/07	Book	\$450.00
011596	04/02/08	Book	\$575.00

Save/Reject: Student Student Charge

Prefer Dataset Merge by Field

User 2

Student#	First Name	Last Name	GPA	Phone
000206	Derwood	Glass	2.95	(312) 804-7430
001956	Diane	Huber	3.75	(312) 519-8147
002037	Gladys	Larson	2.83	(312) 534-0669
002819	Quincy	Jacobson	2.80	(312) 765-0009

Charge No.	chargeDate	chargeCode	Amount
011556	08/28/06	Book	\$326.00
011564	12/27/06	Book	\$700.00
011572	04/02/07	Book	\$669.00
011580	08/28/07	Book	\$325.00
011588	12/27/07	Book	\$450.00
011596	04/02/08	Book	\$575.00

Save/Reject: Student Student Charge

Prefer Dataset Merge by Field

Message (Press HELP to view stack trace)

Data source modified after save-row-changes? yes

Prefer dataset: yes

Merge By Field: yes

Dataset Error? no

Temp-table Student Error? no

Student Buffer Error? no

Dataset Rejected? no

Temp-table Student Rejected? no

Student Buffer Rejected? no

User 2

Student#	First Name	Last Name	GPA	Phone
000206	Derwood	Glass	3.00	(312) 804-7430
001956	Diane	Huber	3.75	(312) 519-8147
002037	Gladys	Larson	2.83	(312) 534-0669
002819	Quincy	Jacobson	2.80	(312) 765-0009

Charge No.	chargeDate	chargeCode	Amount
011556	08/28/06	Book	\$326.00
011564	12/27/06	Book	\$700.00
011572	04/02/07	Book	\$669.00
011580	08/28/07	Book	\$325.00
011588	12/27/07	Book	\$450.00
011596	04/02/08	Book	\$575.00

Save/Reject: Student Student Charge

Prefer Dataset Merge by Field



Saving Data from a ProDataSet to the Database

- And finally, pds10-8 shows the last scenario where both the prefer-dataset and merge-by-field are set to yes.
- Even though prefer-dataset is set to yes, only changed fields from the ProDataSet are copied to the database.
- Since User 2 only changed the phone number, the gpa entered from User 1 will be refreshed on to User 2's screen, see lower right screen in figure pds10-8.



Sharing Datasets Between Procedures and Sessions

- Progress provides the ability to pass ProDataSets between procedures within sessions and between Progress sessions.
- This is accomplished by passing the DataSet as a parameter. The **SHARED** keyword is not allowed on the define dataset statement.
- There are two parameter forms for passing DataSets.
 - The **DATASET** parameter form passes a DataSet as a static object to another procedure in the same session or another session.
 - This is similar to the **PARAMETER TABLE** form for a temp-table.
 - The **DATASET-HANDLE** form passes the DataSet through a handle variable.
 - This form allows you to pass either a static or dynamic DataSet through its handle to another procedure in the same session or another session.
 - This form is parallel to the **PARAMETER TABLE-HANDLE** form for temp-tables.
 - Use this **PARAMETER DATASET-HANDLE** instead of **PARAMETER HANDLE** to allow passing a DataSet between sessions.
- A DataSet can be passed statically using the DataSet parameter type yet received as a dynamic object using the **DATASET-HANDLE** parameter type and vice versa.
- This allows for example, to take a statically defined DataSet on the server and pass it to a generic client procedure that receives it dynamically through a **DATASET-HANDLE**, analyzes its structure through the handle, and uses its contents for client-side objects. Static definitions on the server lend themselves for doing business logic that is specific to a distinct set of tables for validation.

Syntax

DEFINE PARAMETER statement

```
DEFINE { INPUT | OUTPUT | INPUT-OUTPUT } PARAMETER
      DATASET FOR dataset-name [ APPEND ] [ BIND ].
```

RUN statement

```
RUN procedure
  ( [ INPUT | OUTPUT | INPUT-OUTPUT ] DATASET-HANDLE handle-var BY-
  REFERENCE) .
```



Sharing Datasets Between Procedures and Sessions

Passing a DataSet BY-REFERENCE

- Progress by default passes the DataSet by value. This means all the DataSet definitions and data are copied to the called procedure. This is true whether the procedure is in the same session or a different session through an AppServer.
- The overhead of passing DataSets by value is high and should be avoided in most cases.
- Keep in mind that passing a DataSet to a remote (another) session is always copied.
- However, when DataSet is passed locally (in the same session), the call can be optimized by including the keyword **BY-REFERENCE** on the parameter in the RUN statement.
 - If the **BY-REFERENCE** keyword is used and the call is local, Progress optimizes the call by having the called procedure point to the instance of the DataSet in the calling procedure. This eliminates the copying of data to the other procedure.
 - If the **BY-REFERENCE** keyword is used and the call is remote, Progress ignores the **BY-REFERENCE** keyword and passes the DataSet **BY-VALUE**.
 - It is recommended that the **BY-REFERENCE** keyword be used in most cases because of the improved efficiency and the fact that Progress will ignore it if it has to pass the DataSet remotely without giving an error.
 - Progress decided not to make **BY-REFERENCE** the default because of the known side effects with this choice and to be consistent with how other data is passed between procedures.



Sharing Datasets Between Procedures and Sessions

Side Effect Examples

INPUT BY-REFERENCE is like **INPUT-OUTPUT** since any changes that are made inside the called procedure are passed back to the calling procedure.

OUTPUT BY-REFERENCE is like **OUTPUT APPEND** since the data is not emptied from the ProDataSet when passed to the called procedure.

If you don't want the **APPEND**, execute the `hDataSet:EMPTY-DATASET()` method.

Restrictions on DataSet usage

A DataSet cannot be defined inside an internal procedure or trigger. This also true of temp-tables.

The **BY-REFERENCE** behavior is not supported and cannot be used for temp-tables.

The static parameter form:

cannot be used in the main block of a procedure that is run persistently.



Sharing Datasets Between Procedures and Sessions

pds11.p

```
/* pds11.p dataset example eleven - pass dataset changes to another program for viewing. */
.
.
.
on choose of bviewrow
do:
  hdsstuchrg = dataset dsstuchrg:handle.
  create dataset hdschanges.
  hdschanges:create-like(hdsstuchrg).
  hdschanges:get-changes(hdsstuchrg).
  run viewdschg.p (input-output dataset-handle hdschanges by-reference).
  delete object hdschanges no-error.
end.
.
.
.
```

viewdschg.p

```
/* viewdschg.p - program to display changed records in a browse */

define temp-table tstudent no-undo like student
before-table tstudentb4
field chargetot as decimal label "Total Charges"
index studentid is unique primary studentid .

define temp-table tstuchrg no-undo like stuchrg
before-table tstuchrgb4.

define temp-table tcharge no-undo like charge.

define dataset dsstuchrg for tstudent, tstuchrg, tcharge
data-relation stuchrg for tstudent, tstuchrg
relation-fields (studentid, studentid)
data-relation charge for tstuchrg, tcharge
relation-fields (chargecode, chargecode) reposition.

define input-output parameter dataset for dsstuchrg.
.
.
.
wait-for choose of bok.
```



Sharing Datasets Between Procedures and Sessions

Progress

Student#	First Name	Last Name	GPA	Phone	Total Charges
000206	Derwood	Glass	3.00	(312) 804-7430	27,646.00
001956	Diane	Huber	3.75	(312) 519-8147	25,371.00
002037	Gladys	Larson	2.85	(312) 534-0670	25,362.00
002819	Quincy	Jacobson	2.81	(312) 765-0009	25,020.00

Charge No.	chargeDate	chargeCode	Amount	chargeCode	Description
013152	08/28/06	Book	\$325.00	book	Book Charge
013160	12/27/06	Book	\$450.00	finance	Finance Charge
013168	04/02/07	Book	\$625.00	food	Food Charge
013176	08/28/07	Book	\$325.00	other	Other Charge
013184	12/27/07	Book	\$450.00	room	Room Charge
013192	04/02/08	Book	\$575.00	tax	Tax Charge

Save/Reject: Student Student Charge

View Changed Rows

Student#	First Name	Last Name	GPA	Phone	Total Charges
002037	Gladys	Larson	2.85	(312) 534-0670	25,362.00
002819	Quincy	Jacobson	2.81	(312) 765-0009	25,020.00

Student#	Charge No.	chargeDate	chargeCode	Amount
000206	011561	02/22/07	Other	\$35.00
000206	011566	09/15/06	Other	\$55.00
001956	013022	04/02/07	Book	\$700.00
001956	013042	12/27/07	Book	\$500.00

Ok



Sharing Datasets Between Procedures and Sessions

- The purpose of pds11.p is to show how only the changed records can be passed to another procedure for viewing and updating.
- The user may select the View Changed Rows button to view only those records that have changed in the ProDataSet.
- To clarify, it will send only those records that have a **BEFORE-TABLE** buffer. Any student or student charge records that have been saved or rejected will have their **BEFORE-TABLE** buffers cleared and will not be sent to the viewdchg.p procedure called in the "on choose of bviewrow trigger".
- How is this done? It is really very easy! Here are the simple steps:
 1. Define a variable hdschanges of type handle for the change dataset.
 2. Store the handle to the DataSet dstuchrg to the variable hdsstuchrg.
 3. Create a dynamic DataSet and assign it to the variable hdschanges.
 4. Execute the create-like method on the change dataset to inherit the definitions to the origin dataset.
 5. Execute the get-changes method on the change dataset to extract the changes to the origin dataset.
- Once these steps are performed, the changed DataSet may be passed to the viewdchg.p procedure. If this procedure was located on another computer or server the network traffic will be minimized since only the changed data will be sent.
- It's important to remember to delete the dynamic object after calling viewdchg.p to avoid memory leaks.
- In this example, we are passing the DataSet handle hdschanges **BY-REFERENCE** to the viewdchg.p procedure. Because this procedure happens to be local, viewdchg.p will point to the change DataSet created in pds11.p rather than be copied. This will be more efficient. In viewdchg.p, the static DataSet definition is used to display the data in the changed ProDataSet.
- Notice that the queries between the tstudent and tstuchrg temp-tables are not linked since it's possible that there could be a changed tstuchrg record that belongs to a tstudent record that was not changed. In this example, tstudent charge 11561 belonging to tstudent 206 is displayed, but tstudent 206 was not changed and does not appear in the tstudent browse.



Sharing Datasets Between Procedures and Sessions

pds12.p

```

/* pds12.p dataset example twelve - add/change/delete stuchrg records
with update in server program illustrating merge-changes */
.
.
.
on choose of bdel
do:
  def var choice as log.
  def var saverowid as rowid.
  def var saverowid2 as rowid.
  message "Do you want to delete this Student Charge record?"
  view-as alert-box question buttons yes-no update choice.
  if choice then do:
    temp-table tstuchrg:tracking-changes = yes.
    saverowid2 = rowid(tstudent).
    delete tstuchrg.
    hdsstuchrg = dataset dsstuchrg:handle.
    create dataset hdschanges.
    hdschanges:create-like(hdsstuchrg).
    hdschanges:get-changes(hdsstuchrg).
    run upddschg.p (input-output dataset-handle hdschanges by-reference).
    temp-table tstuchrg:tracking-changes = no.
    hdschanges:merge-changes(hdsstuchrg).
    dataset dsstuchrg:accept-changes() no-error.
    delete object hdschanges no-error.
    /* need to refind tstudent record, get-changes method clears tstudent buffer */
    find tstudent where rowid(tstudent) = saverowid2.
    open query qtstuchrg for each tstuchrg of tstudent.
    apply "value-changed" to b2.
  end. /* choice */
end.

on choose of badd
do:
  def var saverowid as rowid.
  def var saverowid2 as rowid.
  temp-table tstuchrg:tracking-changes = yes.
  do with frame chrgadd view-as dialog-box 1 column title "Charge Add":
    form   tstuchrg.studentid label "Student#"
          tstuchrg.chargeno
          tstuchrg.chargedate
          tstuchrg.chargecode view-as combo-box inner-lines 6
          tstuchrg.chargeamt.
  for each charge:
    tstuchrg.chargecode:add-last(charge.chargecode).
  end.
  create tstuchrg.
  assign tstuchrg.studentid = tstudent.studentid
         tstuchrg.chargecode = tstuchrg.chargecode:entry(1)
         saverowid = rowid(tstuchrg)
         saverowid2 = rowid(tstudent).
  display tstuchrg.studentid
         tstuchrg.chargeno.
  update  tstuchrg.chargedate
         tstuchrg.chargecode
         tstuchrg.chargeamt.

```



```

hdsstuchrg = dataset dsstuchrg:handle.
create dataset hdschanges.
hdschanges:create-like(hdsstuchrg).
hdschanges:get-changes(hdsstuchrg).
run upddschg.p (input-output dataset-handle hdschanges by-reference).
temp-table tstuchrg:tracking-changes = no.
hdschanges:merge-changes(hdsstuchrg).
dataset dsstuchrg:accept-changes() no-error.
delete object hdschanges no-error.
/* need to refind tstudent record */
find tstudent where rowid(tstudent) = saverowid2.
b2:set-repositioned-row(3,"conditional").
open query qtstuchrg for each tstuchrg of tstudent.
reposition qtstuchrg to rowid saverowid no-error.
apply "value-changed" to b2.
end. /* with view-as dialog-box 1 column title "Charge Add" */
end.
.
.
.
wait-for close of this-procedure.

```

upddschg.p

```

/* upddschg.p - program to update changed records in a browse */

define temp-table tstudent no-undo like student
before-table tstudentb4
field chargetot as decimal label "Total Charges"
index studentid is unique primary studentid .

define temp-table tstuchrg no-undo like stuchrg
before-table tstuchrgb4.

define temp-table tcharge no-undo like charge.

define dataset dsstuchrg for tstudent, tstuchrg, tcharge
data-relation stuchrg for tstudent, tstuchrg
relation-fields (studentid, studentid)
data-relation charge for tstuchrg, tcharge
relation-fields (chargecode, chargecode) reposition.

define input-output parameter dataset for dsstuchrg.

define data-source srcstuchrg for stuchrg.

def var i as int no-undo.

temp-table tstuchrg:tracking-changes = no.

buffer tstuchrg:attach-data-source(data-source srcstuchrg:handle,"").

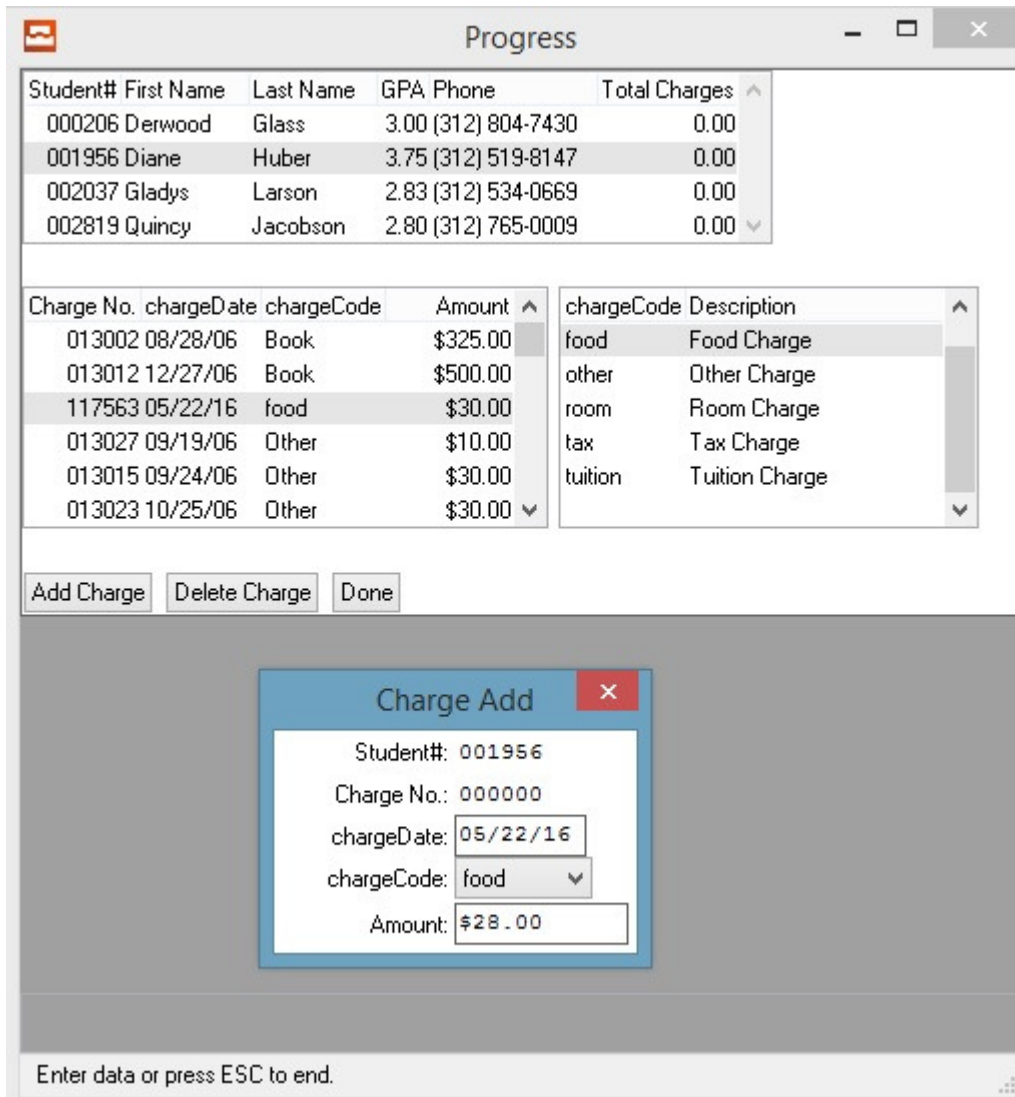
for each tstuchrgb4 where row-state(tstuchrgb4) > 0 transaction:
  buffer tstuchrgb4:save-row-changes("stuchrg","chargeno") no-error.
end. /* for each */

buffer tstuchrg:detach-data-source().

```



Sharing Datasets Between Procedures and Sessions



The screenshot shows a SAS 'Progress' window with two tables and a dialog box. The top table lists student information, and the bottom table lists charges. A 'Charge Add' dialog box is open, allowing the user to add a new charge for a specific student.

Student#	First Name	Last Name	GPA	Phone	Total Charges
000206	Derwood	Glass	3.00	(312) 804-7430	0.00
001956	Diane	Huber	3.75	(312) 519-8147	0.00
002037	Gladys	Larson	2.83	(312) 534-0669	0.00
002819	Quincy	Jacobson	2.80	(312) 765-0009	0.00

Charge No.	chargeDate	chargeCode	Amount	chargeCode	Description
013002	08/28/06	Book	\$325.00	food	Food Charge
013012	12/27/06	Book	\$500.00	other	Other Charge
117563	05/22/16	food	\$30.00	room	Room Charge
013027	09/19/06	Other	\$10.00	tax	Tax Charge
013015	09/24/06	Other	\$30.00	tuition	Tuition Charge
013023	10/25/06	Other	\$30.00		

Buttons: Add Charge, Delete Charge, Done

Charge Add dialog box:

- Student#: 001956
- Charge No.: 000000
- chargeDate: 05/22/16
- chargeCode: food
- Amount: \$28.00

Enter data or press ESC to end.



Sharing Datasets Between Procedures and Sessions

- In pds12.p, the user adds and deletes student charge temp-table records, with both add and delete triggers calling upddschg.p which commits the temp-table changes to the database.
 - This is a typical way of updating the database in many .NET applications.
 - Both add and delete trigger minimizes the data passed to upddschg.p by utilizing a dynamic change DataSet.
 - In addition, the **MERGE-CHANGES** method is used to update the origin DataSet from the change DataSet. This is especially useful for receiving generated key values and calculated values.
 - This method can be used on both the DataSet handle and the buffer temp-table handle.
 - In order to use it, you must remember to turn the **TRACKING-CHANGES** attribute off before running **MERGE-CHANGES**.
 - In pds12.p, we must save the current tstudent rowid before doing the **MERGE-CHANGES** method. This method appears to clear that buffer from memory.
 - Once the tstudent charge data is merged, the tstudent record is re-found and used to open the query for the tstuchrg records of the tstudent.
- In the add trigger, the rowid for the new tstuchrg record created is also stored and used to reposition to the new tstuchrg record in the browse.



Sharing Datasets Between Procedures and Sessions

- In upddschg.p, the **SAVE-ROW-CHANGES** method is used to update the database. The chargeno field is specified in the skip list since it is generated by the chargeseq sequence in the create schema trigger.
- One possible improvement to this application is some error checking in upddschg.p that passes an error back to pds12.p.
- One side note is that we wanted to limit the number of student charge records loaded in the tstuchrg temp-table.
- This was accomplished by changing the **FILL-WHERE-STRING** attribute on the srcstuchrg Data-Source.
 - The value of the **FILL-WHERE-STRING** on the srcstuchrg Data-Source is set to **WHERE** stuchrg.studentid = tstudent.studentid because of the value of the Data-Relation stuchrg in the DataSet definition.
 - Before the **FILL** method is executed, the **FILL-WHERE-STRING** attribute is changed to include only those charges with dates less than 01/01/07.
 - Use the **FILL-WHERE-STRING** on a dependent Data-Source where there is no query defined. Since the student charge table is dependent on the student table in this DataSet, the **FILL-WHERE-STRING** is a convenient way to filter *additional* child records.
 - It is possible to define a query on the child Data-Source to achieve filtering but it is more work and less convenient than the **FILL-WHERE-STRING**.



Sharing Datasets Between Procedures and Sessions

pds13.p

```

/* pds13.p dataset example thirteen - add/change/delete stuchrg records
in 1 transaction with update in server program illustrating merge-changes */
.
.
.
on choose of bcmt
do:
  def var choice as log.
  message "Do you want to commit the changes to the Student Charge records?"
  view-as alert-box question buttons yes-no update choice.
  if choice then do:
    hdsstuchrg = dataset dsstuchrg:handle.
    create dataset hdschanges.
    hdschanges:create-like(hdsstuchrg).
    hdschanges:get-changes(hdsstuchrg).
    run upddschg.p (input-output dataset-handle hdschanges by-reference).
    temp-table tstuchrg:tracking-changes = no.
    hdschanges:merge-changes(hdsstuchrg).
    dataset dsstuchrg:accept-changes() no-error.
    delete object hdschanges no-error.
    disable bcmt with frame f1.
    b2:refresh().
  end. /* choice */
end.

on choose of bdel
do:
  def var choice as log.
  def var saverowid as rowid.
  message "Do you want to delete this Student Charge record?"
  view-as alert-box question buttons yes-no update choice.
  if choice then do:
    enable bcmt with frame f1.
    temp-table tstuchrg:tracking-changes = yes.
    delete tstuchrg.
    open query qtstuchrg for each tstuchrg of tstudent.
    apply "value-changed" to b2.
  end. /* choice */
end.

on choose of badd
do:
  def var saverowid as rowid.
  temp-table tstuchrg:tracking-changes = yes.
  do with frame chrgadd view-as dialog-box 1 column title "Charge Add":
    form   tstuchrg.studentid label "Student#"
          tstuchrg.chargeno
          tstuchrg.chargedate
          tstuchrg.chargecode view-as combo-box inner-lines 6
          tstuchrg.chargeamt.
  for each charge:
    tstuchrg.chargecode:add-last(charge.chargecode).
  end.
  create tstuchrg.
  assign tstuchrg.studentid = tstudent.studentid
         tstuchrg.chargecode = tstuchrg.chargecode:entry(1)

```



Climb Aboard The ProDataSet Train

```

    tstuchrg.chargeno = ?
    saverowid = rowid(tstuchrg).
display tstuchrg.studentid
    tstuchrg.chargeno.
update  tstuchrg.chargedate
    tstuchrg.chargecode
    tstuchrg.chargeamt.
enable bcmt with frame f1.
b2:set-repositioned-row(3,"conditional").
open query qtstuchrg for each tstuchrg of tstudent.
reposition qtstuchrg to rowid saverowid no-error.
apply "value-changed" to b2.
end. /* with view-as dialog-box 1 column title "Charge Add" */
end.
.
.
.

```

Student#	First Name	Last Name	GPA	Phone	Total Charges
000206	Derwood	Glass	3.00	(312) 804-7430	0.00
001956	Diane	Huber	3.75	(312) 519-8147	0.00
002037	Gladys	Larson	2.83	(312) 534-0669	0.00
002819	Quincy	Jacobson	2.80	(312) 765-0009	0.00

Charge No.	chargeDate	chargeCode	Amount	chargeCode	Description
014933	11/07/06	Other	\$30.00	other	Other Charge
014931	11/20/06	Other	\$50.00	room	Room Charge
?	05/28/06	other	\$10.00	tax	Tax Charge
014909	08/02/06	Room	\$400.00	tuition	Tuition Charge
014919	12/02/06	Room	\$550.00		
014908	08/01/06	Tuition	\$3,000.00		



Sharing Datasets Between Procedures and Sessions

Grouping Changes into One Commit

- ProDataSets provide the developer many ways to structure transactions.
- In pds13.p, all the changes, both add and delete of the tstuchrg record are committed in one transaction through the on choose of bcmt trigger.
 - Like the previous example, the upddschg.p procedure is called. Please note that the transaction block is explicitly set for the **FOR EACH** block. If an error occurs on the **SAVE-ROW-CHANGES** method on one of the iterations of the **FOR EACH** block, then it will not affect the other records since they will be in separate transactions.
 - Unlike the previous example, since the **ACCEPT-CHANGES** method is performed in the bcmt trigger and not in the badd or bdel triggers, it is no longer necessary to re-find the tstudent record with the saverowid2 variable.
 - Notice that when the tstuchrg record is added, the charge number is assigned to ? (unknown). If we let the chargeno default to 0, a duplicate key error would result on attempting to create the second tstuchrg record. Remember that the chargeno field key is generated from the schema trigger and not updated until the bcmt trigger.



Sharing Datasets Between Procedures and Sessions

pds14.p

```
/* pds14.p dataset example fourteen - add/change/delete stuchrg records
in 1 transaction with update on appserver */
.
.
on choose of bcmt
do:
  def var hapsrv as handle no-undo.
  def var ok      as logical no-undo.

  def var choice as log.
  message "Do you want to commit the changes to the Student Charge records?"
    view-as alert-box question buttons yes-no update choice.
  if choice then do:
    create server hapsrv.

    ok = hapsrv:connect("-AppService asdbaschool -H localhost -sessionModel Session-free ").
    if not ok then do:
      message "Failed to connect to Appserver" view-as alert-box.
      return no-apply.
    end.
    else message "Connection successful" view-as alert-box.

    hdsstuchrg = dataset dsstuchrg:handle.
    create dataset hdschanges.
    hdschanges:create-like(hdsstuchrg).
    hdschanges:get-changes(hdsstuchrg).
    run upddschg.p on server hapsrv
      (input-output dataset-handle hdschanges by-reference).

    ok = hapsrv:disconnect().
    delete object hapsrv.

    temp-table tstuchrg:tracking-changes = no.
    hdschanges:merge-changes(hdsstuchrg).
    temp-table tstuchrg:tracking-changes = yes.
    dataset dsstuchrg:accept-changes() no-error.
    delete object hdschanges no-error.
    disable bcmt with frame f1.
    b2:refresh().
  end. /* choice */
end.
```



Sharing Datasets Between Procedures and Sessions

Passing a ProDataSet to another Session

- In pds14.p, upddschg.p is called from the asdbaschool AppServer.
- Notice the call to upddschg.p is a little different than before, where we have added the "ON happsrv" phrase to specify that the procedure is to be run on the AppServer.
- Review the 5 steps to using an AppServer:
 1. Create a server handle and store the value in a variable
 2. Use the CONNECT method to connect to an AppServer
 3. Run the program on the AppServer
 4. Use the DISCONNECT method to disconnect from an AppServer
 5. Delete the server handle instance

AdminServer: pgawin81pro641
AppServer: asdbaschool
 Configuration

[Edit](#)

Broker **Agent** SSL Messaging Environment Variables

General

Server executable file:	"@{Startup\DLC}\bin_proapsv.exe"
Server startup parameters:	-db c:\wrk116\db\asdbaschool -ld school
PROPATH:	c:\wrk11564;c:\workspaces\pga\trigpgm;c:\workspaces\pdstrain;c:\workspaces\ws\appserver;c:\courses\keys2oe\examples
Minimum port number:	2002
Maximum port number:	2202
Flush statistical data:	255

Logging Setting

Server log filename:	c:\wrk116\asdbaschool.server.log
Server logging level:	Basic
Append to server log file:	<input checked="" type="checkbox"/>
Server logging entry types:	ASPlumbing,DB.Connects
Server log file threshold size:	0
Maximum number of server log files:	3
Server watch dog interval:	60

Pool Range

Initial number of servers to start:	2
Minimum servers:	1
Maximum servers:	2



Lab 3 – Maintaining ProDataSets Between Procedures and Sessions

1. Update Teacher and Registration Tables Part 1
 - a. Copy teachergrade1.p in the labs folder to teachergrade1lab.p.
 - b. Open it in PDSOE and run the procedure.

teacher#	First Name	Last Name	Email
000001	Bobby	Falk	bfalk@gmail.com
000002	Edgar	Cassidy	ecassidy@gmail.com
000003	Dorothy	Marder	
000004	Giovani	Russell	

Year	Season	Name	Student#	First Name	Last Name	GradePoint
2009	Fall	Introduction to Physics	000905	Kevin	Quinn	?
2009	Fall	Introduction to Physics	002495	Guynell	Scanlon	?
2009	Fall	Introduction to Physics	000801	Chuck	Smythe	?
2009	Fall	Introduction to Physics	000481	Craig	Sorrentino	3.00
2009	Fall	Introduction to Physics	000988	Ellen	Terry	?
2010	Fall	Introduction to Physics	001934	Elise	Perkins	?

Save/Reject: Teacher Student Grade

- c. The top browse lists physics teachers that have offerings in 2009. The user has the ability to update the teacher’s email address.
- d. The bottom browse lists the 2009 Fall offerings of the physics teachers and the students that have registered for them whose last names begin with ‘P’ through ‘Z’.
- e. Update the reject triggers to allow the user to undo changes to temp-tables for both the teacher’s email and the student’s course grade (registration).
- f. Update the save triggers to allow the user to save changes to temp-tables for both the teacher’s email and the student’s course grade (registration).
- g. Follow the commented areas in the triggers.



Lab 3 – Maintaining ProDataSets Between Procedures and Sessions

2. Update Teacher and Registration Tables Part 2

- a. Copy either your completed lab in part 1 teachergrade1lab.p to teachergrade2lab.p, or use the solution in solutions/teachergrade1sol.p to copy to teachergrade2lab.p.

teacher#	First Name	Last Name	Email
000001	Bobby	Falk	bfalk@gmail.com
000002	Edgar	Cassidy	ecassidy@gmail.com
000003	Dorothy	Marder	
000004	Giovani	Russell	

Year	Season	Name	Student#	First Name	Last Name	LetterGrade
2009	Fall	Electricity and Magnetism	001699	Karen	Perez	C+
2009	Fall	Electricity and Magnetism	002286	Guy	Petrova	A+
2009	Fall	Electricity and Magnetism	001994	Phillip	Phillips	B+
2009	Fall	Electricity and Magnetism	000698	Joseph	Pierce	C
2009	Fall	Electricity and Magnetism	000713	Lucy	Piper	B
2009	Fall	Electricity and Magnetism	002311	Helen	Schroeder	?

Save/Reject: Teacher Student Grade

- b. Replace the GradePoint column in browse b2 above with the LetterGrade column (stugrade.gradename). The gradename will automatically be populated with the letter grade from the grade.gradename field if the registration.gradepoint found in the grade table.
- c. Allow the gradename column to be updated (enable) in the browse.
- d. At the top of the program, define a named buffer gradebuf for grade, define a temp-table tgrade like grade, a dataset dsgrade for tgrade and a data-source for gradebuf.
- e. At the bottom of the program, fill the dsgrade dataset with all the grade table records from the database.
- f. Modify the save triggers so that the letter grade is converted to a grade point to be stored in the registration.gradepoint field in the database.
- g. Before the tstugradeb4:save-row-changes() method in each of the triggers, perform the following:
 - i. Find the tstugrade record from the tstugradeb4 record.
 - ii. Find the tgrade record based upon the entered letter grade.
 - iii. Assign the tgrade.gradepoint to the tstugrade.gradepoint field.
- h. Follow the commented areas in the triggers.



Lab 3 – Maintaining ProDataSets Between Procedures and Sessions

3. Move Processing to the AppServer for the trigger on the Save All button
 - a. Copy appserver.p in the labs folder to appserverlab.p.
 - b. Copy updteachergrade.p to updteachergradelab.p.
 - c. Convert the Save All button to call the appserver.
 - d. Follow the comments in the trigger.
 - e. Examine the contents of the updteachergradelab.p.
 - f. Follow the comments in that procedure.



Advanced Topics

Read and Write ProDataSets from JSON

Write-JSON Method

```
/* dsstuchrgwritejson.p - create xml file from prodataset */

define temp-table tstudent no-undo like student
field chargetot as decimal label "Total Charges"
index studentid is unique primary studentid .

define temp-table tstuchrg no-undo like stuchrg.

define temp-table tcharge no-undo like charge.

define dataset dsstuchrg for tstudent, tstuchrg, tcharge
data-relation stuchrg for tstudent, tstuchrg
relation-fields (studentid, studentid)
data-relation charge for tstuchrg, tcharge
relation-fields (chargecode, chargecode).
define query qstudent for student.

define data-source srcstudent
  for query qstudent
  student keys (studentid).

define data-source srcstuchrg for stuchrg.

define data-source srccharge for charge.

query qstudent:query-prepare("for each student
                             where stcode = 'IL'
                             and syear = 2007").

buffer tstudent:attach-data-source(data-source srcstudent:handle,"").
buffer tstuchrg:attach-data-source(data-source srcstuchrg:handle,"").
buffer tcharge:attach-data-source(data-source srccharge:handle,"").

dataset dsstuchrg:fill().

buffer tstudent:detach-data-source().
buffer tstuchrg:detach-data-source().
buffer tcharge:detach-data-source().

buffer tstudent:buffer-field("picture"):SERIALIZE-HIDDEN = true.
dataset dsstuchrg:write-json("file","dsstuchrg.json",true /* formatted */).
buffer tstudent:write-json("file","tstudent.json",true /* formatted */).
find first tstudent.
buffer tstudent:serialize-row("json","file","tstudentrow.json", true /* formatted */).
```



Read and Write ProDataSets from JSON

dsstuchrg.json

```
{
  "dsstuchrg": {
    "tstudent": [
      {
        "StudentID": 206,
        "sfirstName": "Derwood",
        "slastName": "Glass",
        "address1": "443 River Avenue",
        "address2": "",
        "address3": "",
        "city": "Chicago",
        "stCode": "IL",
        "postalCode": "60639",
        "countryCode": "USA",
        "addressAgg": "443 River Avenue Chicago IL 60639 USA",
        "Phone": "3128047418",
        "email": "C:\\wav\\tv\\nipitbud.wav",
        "ethnicId": 1,
        "sex": true,
        "bday": "1985-05-17",
        "syear": 2007,
        "seasonNo": 3,
        "graduated": true,
        "GPA": 2.75,
        "balanceAmt": 7239.00,
        "chargetot": 0.0
      },
      .
      .
    ],
    "tstuchrg": [
      {
        "chargeNo": 11554,
        "studentId": 206,
        "chargeCode": "Tuition",
        "chargeDate": "2006-08-01",
        "chargeAmt": 3000.00,
        "studentChargeDescription": "Tuition for Fall of 2006"
      },
      {
        "chargeNo": 11555,
        "studentId": 206,
        "chargeCode": "Room",
        "chargeDate": "2006-08-02",
        "chargeAmt": 400.00,
        "studentChargeDescription": "Room charge for Fall of 2006"
      },
      .
      .
      .
    ],
    "tcharge": [
      {
        "chargeCode": "food",
        "chargeDescription": "Food Charge"
      },
      {
        "chargeCode": "tuition",
        "chargeDescription": "Tuition Charge"
      }
    ]
  }
}
```



}}

Read and Write ProDataSets from JSON

Write-JSON Method

- Unlike XML, there is no **WRITE-JSONSCHEMA** method. The database structure is inferred by the JSON output.
- The lack of schema information means that the JSON output does not explicitly include indexes and data-relations.
- If this method is used on a temp-table buffer then all the records of the temp-table are written out, not just the record currently in the record buffer.
- If only a single record being output is desired, then use the **SERIALIZE-ROW()** method.
- In dsstuchrgwritejson.p, the **WRITE-JSON()** method is executed after the dataset is filled. The true value is specified for formatting.
- Since the picture buffer-field is a blob, we decided to omit that from the json output by setting the **SERIALIZE-HIDDEN** attribute to true. This is optional as the blob would be output as base64 encoded equivalent of the binary data.
- Using the buffer tstudent with the **WRITE-JSON** method outputs all student rows in the tstudent temp-table.
- In dsstuchrgwritejson.p, the find first tstudent is done before executing the **SERIALIZE-ROW()** method, which output only the current tstudent record.

tstudent.json

```
{
  "tstudent": {
    "StudentID": 206,
    "sfirstName": "Derwood",
    "slastName": "Glass",
    "address1": "443 River Avenue",
    "address2": "",
    "address3": "",
    "city": "Chicago",
    "stCode": "IL",
    "postalCode": "60639",
    "countryCode": "USA",
    "addressAgg": "443 River Avenue Chicago IL 60639 USA",
    "Phone": "3128047418",
    "email": "C:\\wav\\tv\\nipitbud.wav",
    "ethnicId": 1,
    "sex": true,
    "bday": "1985-05-17",
    "syear": 2007,
    "seasonNo": 3,
    "graduated": true,
    "GPA": 2.75,
    "balanceAmt": 7239.00,
    "chargetot": 0.0
  }
}
```



Read and Write ProDataSets from JSON

Write-JSON Method

The following table shows the JSON related attributes for ProDataSets and Temp-tables.

Attribute	Data Type	Applies To	Description
Foreign-Key-Hidden	Logical	Data-Relation	Specifies whether the WRITE-JSON() method should hide foreign key fields in the child records of a nested data-relation in a ProDataSet.
Nested	Logical	Data-Relation	Specifies whether the AVM embeds child rows within a parent row in the JSON. This affects both the data and schema.
Serialize-Hidden	Logical	Buffer-Field	Indicates whether this field is written when the temp-table is serialized, for example into JSON or XML. This attribute interacts with the XML-NODE-TYPE attribute.
Serialize-Name	Char	ProDataSet temp-table temp-table buffer temp-table buffer field	Optionally specifies the name of a ProDataSet, a temp-table, a temp-table buffer, or a temp-table buffer-field object as it should appear when serialized, for example into JSON or XML. This attribute interacts with the XML-NODE-NAME attribute.

Syntax

```
WRITE-JSON ( target-type, {file|stream |stream-handle |memptr|longchar}
[, formatted[, encoding[, omit-initial-values[, omit-outer-object, [, write-before-image]]]] ] )
```

```
SERIALIZE-ROW ( target-format, target-type, {file|stream|stream-handle|memptr|longchar}
[, formatted[, encoding[, omit-initial-values [, omit-outer-object]]]] )
```



Read and Write ProDataSets from JSON

Read-JSON Method

```
/* dsstuchrgreadjson.p - dynamically create prodataset from json schema and data files */

define var dshand as handle.
define var thand as handle.
define var tbuf as handle.
define var qh as handle.
define var fh1 as handle.
define var fh2 as handle.
define var fh as handle.
define var i as int.
define var j as int.
define var ftitle as char.

create dataset dshand.

dshand:read-json("file","dsstuchrg.json","empty").

do i = 1 to dshand:num-buffers with frame a down stream-io:
  tbuf = dshand:get-buffer-handle(i).

  create query qh.

  qh:set-buffers(tbuf).

  qh:query-prepare("for each " + tbuf:name).
  qh:query-open().
  qh:get-first().

  hide frame a.
  frame a:title = "Read Dataset json for " + tbuf:name.
  do while tbuf:available :
    clear frame a all.
    do j = 1 to tbuf:num-fields
      with frame a:
        fh = tbuf:buffer-field(j).
        display fh:name label "Field" format "x(30)"
          string(fh:buffer-value) label "Value" format "x(30)".
      down.
    end.
    qh:get-next().
  end.

end. /* do i = 1 to */

delete object fh no-error.
delete object qh no-error.
delete object tbuf no-error.
delete object dshand no-error.
```



Read and Write ProDataSets from JSON

Read-JSON Method

Here is the display output:

Read Dataset json for tstudent	
Field	Value
StudentID	206
sfirstName	Derwood
slastName	Glass
address1	443 River Avenue
address2	
address3	
city	Chicago
stCode	IL
postalCode	60639
countryCode	USA
addressAgg	443 River Avenue Chicago IL
Phone	3128047418
email	C:\wav\tv\nipitbud.wav
ethnicId	1
sex	yes
bday	1985-05-17
syear	2007
seasonNo	3
graduated	yes
GPA	2.75
balanceAmt	7239
chargetot	0

Read Dataset json for tstuchrg	
Field	Value
chargeNo	11554
studentId	206
chargeCode	Tuition
chargeDate	2006-08-01
chargeAmt	3000
studentChargeDescription	Tuition for Fall of 2006

Read Dataset json for tcharge	
Field	Value
chargeCode	tuition
chargeDescription	Tuition Charge



Read and Write ProDataSets from JSON

Read-JSON Method

- The **READ-JSON()** method loads data into static or dynamic temp-table, temp-table buffer, and ProDataSet objects from a JSON string.
- If the ABL data object has a defined schema, the AVM assumes that the JSON values match up with the ABL fields' data types.
- **READ-JSON()** generates an error message and returns FALSE if the JSON value cannot be converted to the expected ABL data type.
- **READ-JSON()** also accepts JsonObject and JsonArrays as source-types since the ObjectModelParser creates a tree of constructs consisting of JsonObjects and JsonArrays. This makes conversion between JsonConstruct trees and temp-tables or ProDataSets easy.
- If a JSON string contains ProDataSet before-image data, the **READ-JSON()** method populates the after-table and **BEFORE-TABLE** data for the ProDataSet.
- In procedure dsstuchrgreadjson.p, the dsstuchrg.json file is read into the dynamic dataset defined by handle variable dshand.
- Unlike XML, JSON does not have a standard schema language. Because the format of each JSON value indicates its data type, the AVM can infer a schema from a JSON string.
- This is the case in dsstuchrgreadjson.p since there is no schema defined for this dynamic dataset.
- Please be aware that inferring schema is an inexact science. OpenEdge makes certain assumptions about the json string when constructing the schema. Unpredictable results may occur.
- When the AVM has to infer schema for the data object, the AVM makes two passes through the JSON data: one to build the schema and one to fill in the data.
- Please see the OpenEdge development documentation “Working with JSON” for guidelines on inferring ABL schema from JSON data.

Syntax

```
READ-JSON ( source-type, {file|memptr|handle|longchar | JSONArray |  
JsonObject}{[, read-mode] )
```



Using .NET Grids

addgrid.p

```

/* addgrid.p - add ultragrid control to form */

using System.Windows.Forms.*.

def var form1 as Progress.Windows.Form no-undo.
def var ustudGrid as Infragistics.Win.UltraWinGrid.UltraGrid no-undo.
def var bsstudent as Progress.Data.BindingSource no-undo.

def query qstudent for student scrolling.

form1 = new Progress.Windows.Form ().
ustudGrid = new Infragistics.Win.UltraWinGrid.UltraGrid().

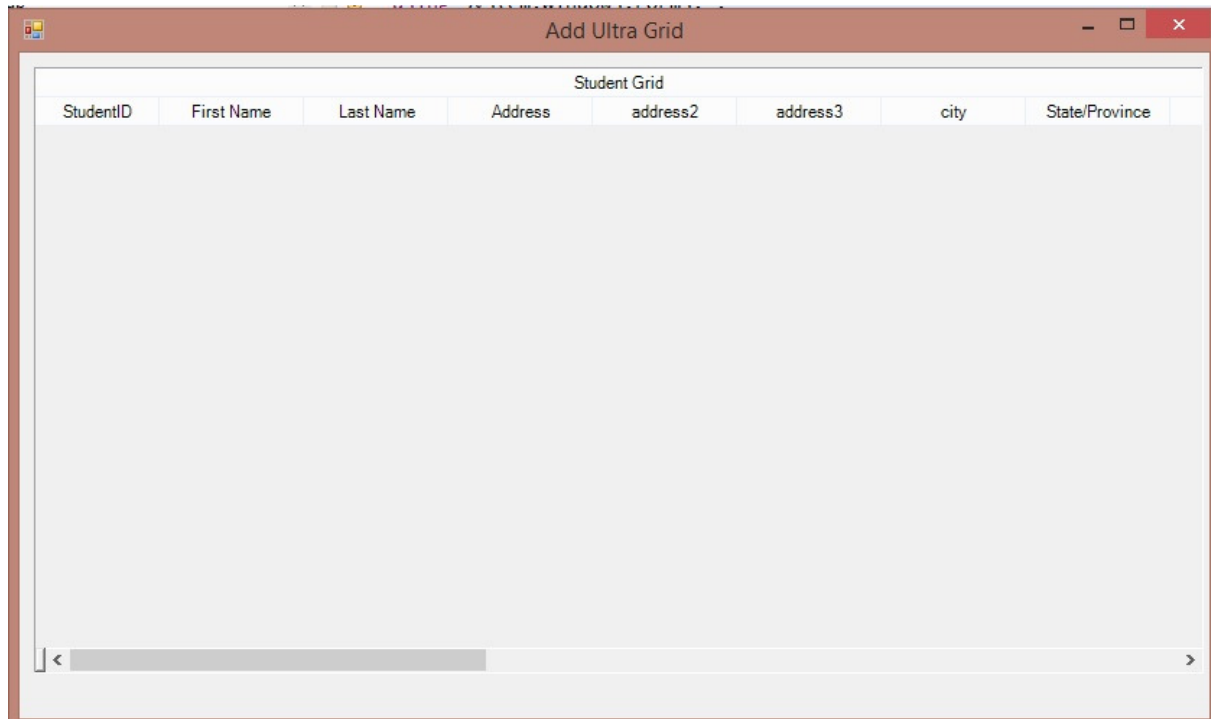
/* open query qstudent for each student no-lock where student le 100. */
bsstudent = new Progress.Data.BindingSource().
bsstudent:handle = query qstudent:handle no-error.

form1:Width = 840.
form1:Height = 500.
form1:text = "Add Ultra Grid".

ustudgrid:Left = 10.
ustudgrid:Top = 10.
ustudgrid:Width = 810.
ustudgrid:Height = 420.
ustudgrid:Text = "Student Grid".
ustudgrid:DataSource = bsstudent.
form1:controls:Add(ustudGrid).

wait-for Application:Run(form1).

```





Using .NET Grids

- .NET Grids are similar to a browse in traditional GUI but with many more visual options.
- The Infragistics UltraGrid is particularly robust in how tabular data can be displayed.
- In addgrid.p, the UltraGrid's DataSource property is assigned to the binding source bsstudent.
- Notice that the headings are displayed but there is no data. This is because the query qstudent is assigned to the binding source but the query has not been opened.
- The query needs to be assigned to the binding source in order for the column headings to appear.



Using a Dataset with a .NET Grid

addgridpds.p

```
/* addgridpds.p - add a grid with binding to the student table */  
  
using System.Windows.Forms.*.  
  
def var form1 as Progress.Windows.Form no-undo.  
def var ustudGrid as Infragistics.Win.UltraWinGrid.UltraGrid no-undo.  
def var bsstudent as Progress.Data.BindingSource no-undo.  
  
def temp-table tstudent no-undo like student.  
  
def dataset dsstudent for tstudent.  
  
def query qstudent for student scrolling.  
  
def data-source srcstudent for query qstudent student keys (studentid).  
  
buffer tstudent:attach-data-source(data-source srcstudent:handle).  
  
query qstudent:query-prepare("for each student NO-LOCK where studentid <= 100").  
  
dataset dsstudent:fill().  
  
buffer tstudent:detach-data-source().  
  
form1 = new Progress.Windows.Form ().  
ustudGrid = new Infragistics.Win.UltraWinGrid.UltraGrid().  
  
bsstudent = new Progress.Data.BindingSource().  
bsstudent:handle = dataset dsstudent:handle.  
  
form1:Width = 840.  
form1:Height = 500.  
form1:text = "Add Ultra Grid".  
  
ustudgrid:Left = 10.  
ustudgrid:Top = 10.  
ustudgrid:Width = 810.  
ustudgrid:Height = 420.  
ustudgrid:Text = "Student Grid with ProDataSet".  
ustudgrid:DataSource = bsstudent.  
form1:controls:Add(ustudGrid).  
  
wait-for Application:Run(form1).
```



Using a Dataset with a .NET Grid

addgridpds.p

Add Ultra Grid

Student Grid with ProDataSet							
StudentID	First Name	Last Name	Address	address2	address3	city	State/Province
1	Derwood	Serck	494 Sixth Place	Suite 10		Providence	RI
2	Emily	Levy	622 Fifth Place	Suite 2		Columbus	OH
3	Laura	Dunn	182 Massachuset	Suite 4		Little Rock	AR
4	Dorothy	Davidson	225 Golf Place	Suite 4		Houston	TX
5	Raymond	Olson	614 Church Court	Suite 5		Wichita	KS
6	Barry	Fiocchi	591 Oriole Boulev	Apt. 24		Montreal	QUE
7	Larry	Bobbitt	464 King Avenue			Milwaukee	WI
8	Kyle	Talbot	57 Lincoln Avenu	Apt. 28		San Francisco	CA
9	Atwood	Cunningham	449 Eastern Boul	Apt. 23		Richmond	VA
10	Dana	Pollack	816 Cardinal Plac			Omaha	NE
11	Rory	Liebovich	968 Second Stree	Apt. 21		Salt Lake City	UT
12	Athleen	Sweeney	402 Lincoln Court			Little Rock	AR
13	Raymond	Torres	94 3rd Court	Apt. 9		Wilmington	DE
14	Richardo	Sweeney	445 Northern Ave			Dallas	TX
15	Jennifer	Marsh	813 Georgia Park			Los Angeles	CA
16	Harvey	Hawkins	672 Cedar Avenu	Apt. 25		Chicago	IL
17	Diane	Shapiro	723 Cedar Court			Las Vegas	NV

- In addgridpds.p, a dataset called dsstudent is defined for the temp-table tstudent.
- To use a dataset, at least one data-source is typically defined.
- A query is prepared on the top level temp-table to limit the number of records, in this case the first 100 student records are read.
- After the data-source is attached, the dataset is populated using the fill() method before the data-source is detached.
- Last but not least, the dataset is attached to the probindingsource. This allows the data in the prodataset to be populated in the grid.



Using a multi-table Dataset with a .NET Grid

addgridpds2.p

```
/* addgridpds2.p - add multiple tables to a grid, student, student activity and activity records
*/

using System.Windows.Forms.*.

def var form1 as Progress.Windows.Form no-undo.
def var ustudGrid as Infragistics.Win.UltraWinGrid.UltraGrid no-undo.
def var bsstudent as Progress.Data.BindingSource no-undo.

def temp-table tstudent no-undo like student.
def temp-table tstuact no-undo like stuact.
def temp-table tactivity no-undo like activity.

def dataset dsstuact for tstudent, tstuact, tactivity
data-relation stuact for tstudent, tstuact
relation-fields (studentid,studentid)
data-relation activy for tstuact, tactivity
relation-fields (activityid,activityid).

def query qstudent for student scrolling.

def data-source srcstudent for query qstudent student keys (studentid).
def data-source srcstuact for stuact.
def data-source srcactivity for activity.

buffer tstudent:attach-data-source(data-source srcstudent:handle).
buffer tstuact:attach-data-source(data-source srcstuact:handle).
buffer tactivity:attach-data-source(data-source srcactivity:handle).

query qstudent:query-prepare("for each student NO-LOCK where studentid <= 100").

dataset dsstuact:fill().

buffer tstudent:detach-data-source().
buffer tstuact:detach-data-source().
buffer tactivity:detach-data-source().

form1 = new Progress.Windows.Form ().
ustudGrid = new Infragistics.Win.UltraWinGrid.UltraGrid().

bsstudent = new Progress.Data.BindingSource().
bsstudent:handle = dataset dsstuact:handle.
.
.
.
ustudgrid:Text = "Student/Activity Grid with ProDataSet".
ustudgrid:DataSource = bsstudent.
form1:controls:Add(ustudGrid).

wait-for Application:Run(form1).
```



Using a multi-table Dataset with a .NET Grid

addgridpds2.p

Add Ultra Grid

StudentID	First Name	Last Name	Address	address2	address3	city	State/F
1	Derwood	Serck	494 Sixth Place	Suite 10		Providence	RI
studentId		Activity ID					
1	7						
1	15						
1	24						
StudentID	First Name	Last Name	Address	address2	address3	city	State/F
2	Emily	Levy	622 Fifth Place	Suite 2		Columbus	OH
studentId		Activity ID					
2	2						
Activity ID		Name					
2	baseball						
StudentID	First Name	Last Name	Address	address2	address3	city	State/F
3	Laura	Dunn	182 Massachuset	Suite 4		Little Rock	AR
4	Dorothy	Davidson	225 Golf Place	Suite 4		Houston	TX
5	Raymond	Olson	614 Church Court	Suite 5		Wichita	KS
6	Barry	Fiocchi	591 Oriole Boulev	Apt. 24		Montreal	QUE
7	Larry	Bobbitt	464 King Avenue			Milwaukee	WI

- In addgridpds2.p, a multi-table dataset is created consisting of the student, activity and stuact tables.
- The stuact table is the cross reference table between the student and activity tables.
- The dsstuact dataset has 3 levels, starting with the student table, then filling all the stuact records, then getting the single activity record for each stuact record.
- By assigning the dataset dsstuact to the bsstudent binding source, automatically creates three levels in the UltraGrid.



Using a multi-table Dataset with a .NET Grid

addgridpds3.p

```
/* addgridpds3.p - add grid for student and stuact records adding activity name to temp-table */
.
.
.
def temp-table tstudent no-undo like student.
def temp-table tstuact no-undo like stuact
field activityname like activity.activityname.

def dataset dsstuact for tstudent, tstuact
data-relation stuact for tstudent, tstuact
relation-fields (studentid,studentid).

def query qstudent for student scrolling.

def data-source srcstudent for query qstudent student keys (studentid).
def data-source srcstuact for stuact.

buffer tstudent:attach-data-source(data-source srcstudent:handle).
buffer tstuact:attach-data-source(data-source srcstuact:handle).

dataset dsstuact:set-callback-procedure ("after-fill", "postdsstuactFill", THIS-PROCEDURE).

query qstudent:query-prepare("for each student NO-LOCK where studentid <= 100").

dataset dsstuact:fill().

buffer tstudent:detach-data-source().
buffer tstuact:detach-data-source().

form1 = new Progress.Windows.Form ().
ustudGrid = new Infragistics.Win.UltraWinGrid.UltraGrid().

bsstudent = new Progress.Data.BindingSource().
bsstudent:handle = dataset dsstuact:handle.
.
.
.
wait-for Application:Run(form1).
procedure postdsstuactfill:
define input parameter dataset for dsstuact.
for each tstuact,
each activity no-lock where activity.activityid = tstuact.activityid:
assign tstuact.activityname = activity.activityname.
end. /* for each stuact */
end. /* proc postdsstuactfill */
```



Using a multi-table Dataset with a .NET Grid

addgridpds3.p

Add Ultra Grid

StudentID	First Name	Last Name	Address	address2	address3	city	State/Provir												
1	Derwood	Serck	494 Sixth Place	Suite 10		Providence	RI												
<table border="1"> <thead> <tr> <th>studentId</th> <th>Activity ID</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>7</td> <td>government</td> </tr> <tr> <td>1</td> <td>15</td> <td>field hockey</td> </tr> <tr> <td>1</td> <td>24</td> <td>Chugging Beer</td> </tr> </tbody> </table>								studentId	Activity ID	Name	1	7	government	1	15	field hockey	1	24	Chugging Beer
studentId	Activity ID	Name																	
1	7	government																	
1	15	field hockey																	
1	24	Chugging Beer																	
2	Emily	Levy	622 Fifth Place	Suite 2		Columbus	OH												
<table border="1"> <thead> <tr> <th>studentId</th> <th>Activity ID</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>2</td> <td>2</td> <td>baseball</td> </tr> </tbody> </table>								studentId	Activity ID	Name	2	2	baseball						
studentId	Activity ID	Name																	
2	2	baseball																	
3	Laura	Dunn	182 Massachuset	Suite 4		Little Rock	AR												
4	Dorothy	Davidson	225 Golf Place	Suite 4		Houston	TX												
5	Raymond	Olson	614 Church Court	Suite 5		Wichita	KS												
6	Barry	Fiocchi	591 Oriole Boulev	Apt. 24		Montreal	QUE												
7	Larry	Bobbitt	464 King Avenue			Milwaukee	WI												
8	Kyle	Talbot	57 Lincoln Avenue	Apt. 28		San Francisco	CA												
9	Atwood	Cunningham	449 Eastern Boul	Apt. 23		Richmond	VA												

- In the previous example, addgridpds2.p, the activity name was on the third level and took two clicks to get there to see what the activity was.
- In addgridpds3.p, the dsstuact dataset was reduced to two levels, using just the tstudent and tstuact temp-tables.
- The activity name was added to the tstuact temp-table.
- So how is the activity name populated?
- The answer is the dataset set-callback-procedure method.
- After filling the entire dataset, all the tstuact temp-table records are re-read and joined with the activity record to populate the tstuact.activityname field.
- This could also have been achieved using a query on the child dataset as in pds33.p.



ProDataSet Summary

In this workshop, we have shown how to:

- Fill DataSets
- Advanced Record Reading – batching, multiple buffers and child level queries
- Change DataSets
- Update the Database
- Error Processing
- Passing DataSets to other procedures and the AppServer
- Advanced Topics – Read/Write JSON and Using .NET Grid