PDS Fill From the VST

By Tim Kuehn
Senior Consultant

www.tdkcs.com

Who Am I?

- Tim Kuehn
- Senior Consultant / Owner of TDK Consulting Services Inc
- In the Progress world since v8.1 was “new” (v12.1 is the current version`)
- Presented at many PUG and Progress conferences
- One of the premier ABL language experts in the Progress world.

Agenda

- Talk about database VSTs
- Describe Dataset technology
- Present examples of how Dataset technology can be used to read Schema VST data

What are VSTs?

From the Progress Documentation site:

Virtual system tables (VSTs) are tables that give ABL and SQL applications access to the same type of database information that you collect with the OpenEdge Monitor (PROMON) utility. Virtual system tables enable an application to examine the status of a database and monitor its performance. With the database broker running, ABL and SQL applications can query a VST and retrieve the specified information as run-time data.

https://docs.progress.com/bundle/manage-database/page/Virtual-System-Tables.html

What the Progress Docs leave out:

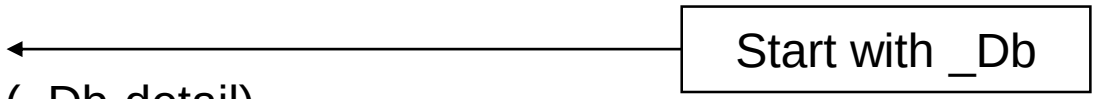
VSTs hold schema related information for a database's structure which includes tables, fields, indexes, and index structures that can be queried from ABL and SQL.*

** You can also update the schema – at your own risk*

An Inventory of Schema VSTs

Schema related VSTs:

- Database (_Db)
- Database Details (_Db-detail)
- Tables (_File)
- Table Fields (_Field)
- Table Indexes (_Index)
- Index Fields (_Index-Field)
- Sequence (_Sequence)



PUG Sessions with details on VSTs:

PUG Challenge 2019: http://pugchallenge.org/downloads2019.html
364: **Progress System Tables** Dan Foreman White Star Software

PUG Challenge 2015 http://pugchallenge.org/downloads2015.html
448: Secrets Of The OpenEdge RDBMS: Schema Tables — Gus Bjorklund, Progress Software

Direct link: http://pugchallenge.org/downloads2015/448_schema_tables_v04.pdf

_Db: The Database VST


```
Field-Name         Ext  Data-Type
------------------ ---  --------------
_Db-addr                character
_Db-coll-name           character
_Db-collate         10  raw
_Db-comm                character
_Db-guid                character
_Db-local               logical
_Db-misc1            8  integer
_Db-misc2            8  character
_Db-name                character
_Db-res1             8  integer
_Db-res2             8  character
_Db-revision            integer
_Db-slave               logical
_Db-type                character
_Db-xl-name             character
_Db-xlate            2  raw
_User-Misc              character
```

Record contents

What are ProDatasets?

The Long Story (from the 11.7 ABL Doc “ProDatasets”):

- Extends your ability to define complex business objects with many levels of related data.
- Lets you define the relationships between those levels of data.
- Lets you associate each level with a distinct data source when you need to fill the ProDataSet with data or pass updates back to the database.
- Allows you to define a mapping between the database tables that are the source of your data and its representation within the ProDataSet. The internal view of the data can be significantly different from how it is physically stored.
- Defines hooks that let you associate your own custom procedures with many events in the life cycle of a ProDataSet.
- Logs changes to ProDataSet records so that you can pass the ProDataSet to a separate procedure to write those changes back to the database.
- Allows you to pass the ProDataSet as a single parameter with a single handle from one procedure to another, within a single ABL session or between sessions.

The Tim Version:

Prodatasets are a collection of technologies that describe a set of temp-tables, their relationships, persistent tables, and ways to marshal data between the dataset temp tables and a persistent data store.

The technologies that implement prodatasets include:

- Temp Tables
- Database Tables
- Data Sources
- Data Set
- Data Relations

PDS read _Db: Temp Table

Temp Table Definition:

```
DEFINE TEMP-TABLE tt-db NO-UNDO LIKE _Db.
```

A temp-table is a local table that is not part of a persistent database. Temp-tables can be used within a session or passed between sessions. They also provide in-memory buffering of data and transparent overflow to a temporary database on disk when necessary.

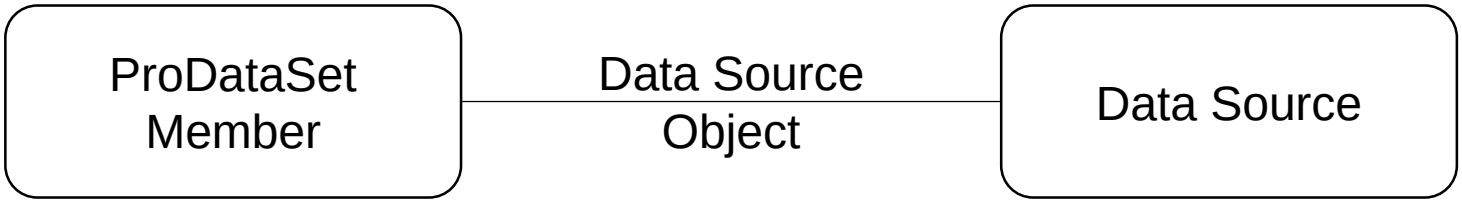
PDS read _Db: Data Source

Data Source Definition:

```
DEFINE DATA-SOURCE dsrc-db FOR _Db.
```

- A data source is a standard OpenEdge database or other database accessible va an OpenEdge Data Server and is described by a *Data-Source* object.
- Attaching a Data-Source to a ProDataSet temp-table buffer provides a way for moving data between the ProDataSet and the data source.

Just like separating a UI from BL, data source technology separates a ProDataSet from database-specific definitions or dependencies. This allows an application to use a variety of Data-Source configurations for given ProDataSet.



PDS read _Db: ProDataSet

Temp Table:

```
DEFINE TEMP-TABLE tt-db NO-UNDO LIKE _Db.
```

Data Source Definition:

```
DEFINE DATA-SOURCE dsrc-db FOR _Db.
```

Data Set Definition:

```
DEFINE DATASET pds-db FOR tt-db.
```

A ProDataSet describes a complex business entity consisting of one or more temp-tables, their relationships, and their related data sources.

DataSets provide a way to

- define complex business objects with many levels of related data.
- associate each member temp-table with a distinct data source
- associate custom procedures with each member temp-table
- log data changes to update a data source
- pass a ProDataSet as a single parameter from one procedure to another, within a single ABL session or between sessions.

PDS read _Db: Attach Data-Source

Temp Table:

```
DEFINE TEMP-TABLE tt-db NO-UNDO LIKE _Db.
```

Data Source Definition:

```
DEFINE DATA-SOURCE dsrc-db FOR _Db.
```

Data Set Definition:

```
DEFINE DATASET pds-db FOR tt-db.
```

Attach Data Source to a Buffer:

```
BUFFER tt-db:ATTACH-DATA-SOURCE(DATA-SOURCE dsrc-db:HANDLE).
```

Attaching a data source to a dataset temp table associates the temp-table with a data source table.

PDS read _Db: Filling the PDS

Fill the Datatset:

```
DATASET pds-db:FILL().
```

The PDS FILL method

1. Identifies all the top-level buffers in the ProDataSet
2. Starts a nested filling operation on each top-level buffer.
3. Using the Data-Relation for which the top buffer as the parent, the FILL process then proceeds through parent-child relationships iteratively by loading the parent temp-table, then filling each child temp-table with those records related to the current parent.
4. This process cascades through all child temp tables until there are no more records to load.

PDS read _Db: Putting it all Together

```
/* db.p */
```

Temp Table:

```
  DEFINE TEMP-TABLE tt-db NO-UNDO LIKE _Db.
```

Data Source Definition:

```
  DEFINE DATA-SOURCE dsrc-db FOR _Db.
```

```
Data Set Definition:
```

```
  DEFINE DATASET pds-db FOR tt-db.
```

```
Attach Data Source to a Buffer:
```

```
  BUFFER tt-db:ATTACH-DATA-SOURCE(DATA-SOURCE dsrc-db:HANDLE).
```

```
Fill the PDS:
```

```
  DATASET pds-db:FILL().
```

PDS read of the _Db table

Outputting the resulting TT:

```
FOR EACH tt-db:
  DISPLAY tt-db._db-name
          tt-db._db-type
          tt-db._db-guid
      WITH DOWN.
END.
```

Result:

PDS: Reading more than one table

We've read _DB into a TT
How do we read _Db-detail into another TT in the same PDS?

_Db-detail: The Database Detail VST


```
Field-Name          Ext  Data-Type
------------------- ---  ---------------
_Db-custom-detail        character
_Db-description          character
_Db-guid                 character
_Db-mac-key              raw
```

Database GUID

_Db-guid is used to identify a distinct instance of a database

procopy preserves _Db-guid

procopy –newinstance changes _Db-guid and adds a new _Db-detail record

_Db and _Db-Detail data

Sports2000 _Db record:

Original sports2000 structure

```
          Db-name: ?
          Db-addr: ?
          Db-type: PROGRESS
          Db-comm:
      Db-revision: ?
         Db-slave: no
         Db-local: yes
        User-Misc: ?
          Db guid: 7RDfBfV1KKAiFGD4vP9NrQ
```

Sports2000 _Db-detail record:

```
Database custom :
Database descrip: sports2000
   Database guid: 7RDfBfV1KKAiFGD4vP9NrQ
```

More than one _db-detail record

sports2000 structure after procopy -newinstance

Sports2000 _Db-detail before copy:

```
Database custom :
Database descrip: sports2000
   Database guid: 7RDfBfV1KKAiFGD4vP9NrQ
```

Make a new copy of the database:

Procopy sports2000 copy2 –newinstance

Sports2000 _Db-detail contents after copy:

```
Database custom :
Database descrip: sports2000
   Database guid: 7RDfBfV1KKAiFGD4vP9NrQ

Database custom :
Database descrip: copy2
   Database guid: wAJwszCKKr5YFLv0ZA34AQ
```

New _Db-detail record with new _Db-guid

More on when a new _db-guid is created in _db-detail:
https://knowledgebase.progress.com/articles/Article/000054446

PDS read of the _Db, _Db-Detail tables

Add or update the _Db PDS example by adding the following code:

```
/* db-detail.p  */
```

Temp Table:

```
DEFINE TEMP-TABLE tt-db-detail NO-UNDO LIKE _Db-detail.
```

Data Source Definition:

```
DEFINE DATA-SOURCE dsrc-db-detail FOR _Db-detail.
```

Data Set Definition:

```
DEFINE DATASET pds-db-detail FOR tt-db, tt-db-detail.
```

Attach Data Source to a Buffer:

```
   BUFFER tt-db-detail:ATTACH-DATA-SOURCE(
                              DATA-SOURCE dsrc-db-detail:HANDLE).
```

Fill the Datatset:

```
DATASET pds-db-detail:FILL().
```

PDS Read: _Db, _Db-Detail


```
DATASET pds-db-detail:FILL().

FOR EACH tt-db NO-LOCK:
    DISPLAY tt-db._db-name
            tt-db._db-type
            tt-db._db-guid
        WITH FRAME f-db
            2 DOWN.
END.

FOR EACH tt-db-detail NO-LOCK:
    DISPLAY tt-db-detail._Db-description    FORMAT "X(15)"
            tt-db-detail._Db-guid
        WITH FRAME f-detail
            1 COL 2 DOWN.
END.
```

PDS: Reading Related Records

Loading all the records in a table is good for some use cases

What if you only want related child records?

PDS Data-Relation

A Data-Relation defines a relationship between a parent buffer and a child buffer in the ProDataSet. These relations name the fields in the parent and child temp tables that form a primary-foreign key relationship between the buffers.

When the ProDataSet is populated, or filled, the AVM uses this relation to automatically locate and read the child records for a given parent record.

The Data-Relation can also used in "navigation" mode where the application traverses a ProDataSet to access data, make changes, add records, and so on. In this mode, the Data-Relations are used to automatically create a dynamic query to link parent records to child records with the same keys. Use cases for this include displaying temp-table records in a browse, or when traversing child records related to the current parent.

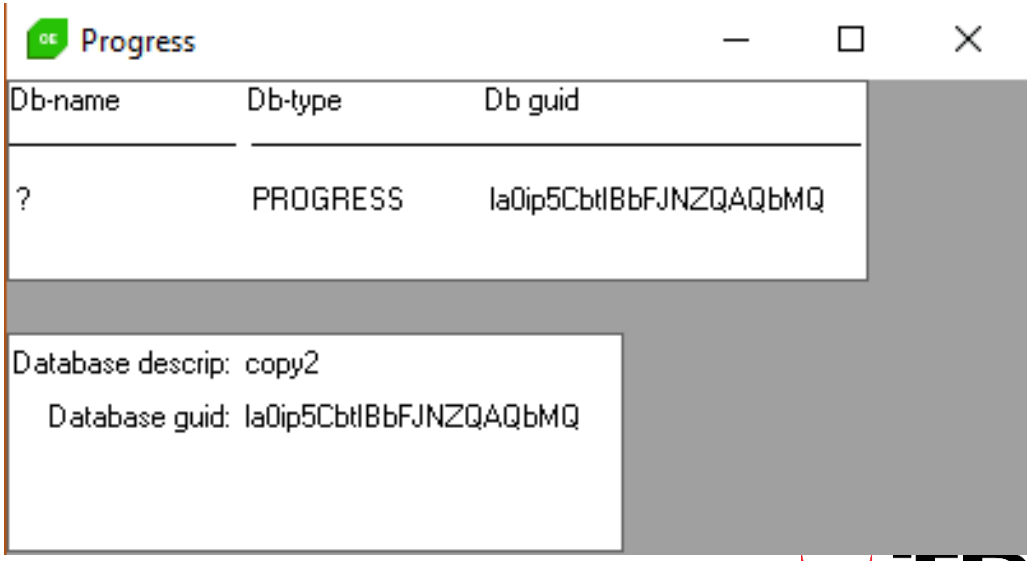
PDS: Relating _Db to _Db-detail


```
/* db-db-detail-relation.p  */

DEFINE TEMP-TABLE tt-db           NO-UNDO LIKE _Db.
DEFINE TEMP-TABLE tt-db-detail    NO-UNDO LIKE _Db-detail.

DEFINE DATA-SOURCE dsrc-db         FOR _Db.
DEFINE DATA-SOURCE dsrc-db-detail FOR _Db-detail.

DEFINE DATASET pds-db-detail
   FOR tt-db,
       tt-db-detail
   DATA-RELATION r1 FOR tt-db,
                        tt-db-detail
                    RELATION-FIELDS(_db-guid, _db-guid).
```

PDS Data-Relation

Results Before Data-Relation

Results After Data-Relation



What is _File?

_File holds all the table-level information for a database including the table name, dump and load name, file level triggers, and the like.

_File Table Structure

Field-Name	Ext	Data-Type
_Cache		raw
_Can-Create		character
_Can-Delete		character
_Can-Dump		character
_Can-Load		character
_Can-Read		character
_Can-Write		character
_category		character
_CRC		integer
_Creator		character
_DB-lang		integer
_Db-recid		recid
_Desc		character
_dft-pk		logical
_Dump-name		character
_Field-map		raw
_Fil-misc1	8	integer
_Fil-misc2	8	character
_Fil-Misc3	8	integer
_Fil-Misc4	8	character
_Fil-res1	8	integer
_Fil-res2	8	character
_File-Attributes	64	logical
_File-Label		character
_File-Label-SA		character
_File-Name		character
_File-Number		integer
_For-Cnt1		integer
_For-Cnt2		integer

Field-Name	Ext	Data-Type
_For-Flag		integer
_For-Format		character
_For-Id		integer
_For-Info		character
_For-Name		character
_For-Number		integer
_For-Owner		character
_For-Size		integer
_For-Type		character
_Frozen		logical
_Has-Ccnstrs		character
_Has-Fcnstrs		character
_Has-Pcnstrs		character
_Has-Ucnstrs		character
_Hidden		logical
_ianum		integer
_Last-change		integer
_Last-modified		datetime-tz
_Mod-sequence		integer
_numfld		integer
_numkcomp		integer
_numkey		integer
_numkfld		integer
Owner		character
_Prime-Index		recid
_Rssid		integer

_File Table Structure


```
Field-Name         Ext Data-Type
------------------ --- ---------------
 Tbl-Status            character
_Tbl-Type              character
_Template              recid
_User-Misc             character
_Valexp                character
_Valmsg                character
_Valmsg-SA             character
_Version               integer
```

PDS read _File


```
/* file.p */
DEFINE TEMP-TABLE tt-file NO-UNDO   LIKE _File.
DEFINE DATA-SOURCE dsrc-file        FOR  _File.

DEFINE DATASET pds-file             FOR tt-file.

BUFFER tt-file:ATTACH-DATA-SOURCE(DATA-SOURCE dsrc-file:HANDLE).

DATASET pds-file:FILL().

FOR EACH tt-file:
    DISPLAY tt-file._file-name
            tt-file._tbl-type
        WITH DOWN.
END.
```

PDS read _File

File-Name	Tbl-Type
Benefits	T
BillTo	T
Bin	T
Customer	T
Department	T
Employee	T
Family	T
Feedback	T
InventoryTrans	T
Invoice	T
Item	T
LocalDefault	T
Order	T
OrderLine	T
POLine	T
PurchaseOrder	T
RefCall	T
Salesrep	T
ShipTo	T
State	T

File-Name	Tbl-Type
Supplier	T
SupplierItemXref	T
SYSBIGINTSTAT	V
SYSCALCTABLE	V
SYSCHARSTAT	V
SYSCHKCONSTR_NAME_MAP	V
SYSCOLAUTH	V
SYSCOLSTAT	V
SYSCOLUMNS	V
SYSCOLUMNS_FULL	V
SYSDATATYPES	V
SYSDATESTAT	V
SYSDBAUTH	V
SYSFLOATSTAT	V
SYSIDXSTAT	V
SYSINDEXES	V
SYSINTSTAT	V
SYSNCHARSTAT	V
SYSNUMSTAT	V
SYSNVARCHARSTAT	V

File-Name	Tbl-Type
SYSTSSTAT	V
SYSTSTZSTAT	V
SYSVARCHARSTAT	V
SYSVIEWS	V
SYSVIEWS_NAME_MAP	V
SYS_CHKCOL_USAGE	V
SYS_CHK_CONSTRS	V
SYS_KEYCOL_USAGE	V
SYS_REF_CONSTRS	V
SYS_TBL_CONSTRS	V
TimeSheet	T
Vacation	T
Warehouse	T
_ActAILog	S
_ActBILog	S
_ActBuffer	S
_ActIndex	S
_ActIOFile	S
_ActIOType	S
_ActLock	S

PDS read _File: The Problem to Solve

File-Name	Tbl-Type
SYSTSSTAT	V
SYSTSTZSTAT	V
SYSVARCHARSTAT	V
SYSVIEWS	V
SYSVIEWS_NAME_MAP	V
SYS_CHKCOL_USAGE	V
SYS_CHK_CONSTRS	V
SYS_KEYCOL_USAGE	V
SYS_REF_CONSTRS	V
SYS_TBL_CONSTRS	V
TimeSheet	T
Vacation	T
Warehouse	T
_ActAILog	S
_ActBILog	S
_ActBuffer	S
_ActIndex	S
_ActIOFile	S
_ActIOType	S
_ActLock	S

How to get just the “T” types?

PDS read _File: FILL-WHERE-STRING


```
/* file-filter.p   */
DEFINE TEMP-TABLE tt-file NO-UNDO   LIKE _File.

DEFINE DATA-SOURCE dsrc-file        FOR _File.

DEFINE DATASET pds-file             FOR tt-file.

BUFFER tt-file:ATTACH-DATA-SOURCE(DATA-SOURCE dsrc-file:HANDLE).

DATA-SOURCE dsrc-file:FILL-WHERE-STRING
                                    = "WHERE _File._Tbl-Type = ""T""".

DATASET pds-file:FILL().

FOR EACH tt-file:
    DISPLAY tt-file._file-name
            tt-file._tbl-type
        WITH DOWN.
END.
```

PDS read _File: FILL-WHERE-STRING

File-Name	Tbl-Type
Benefits	T
BillTo	T
Bin	T
Customer	T
Department	T
Employee	T
Family	T
Feedback	T
InventoryTrans	T
Invoice	T
Item	T
LocalDefault	T
Order	T
OrderLine	T
POLine	T
PurchaseOrder	T
RefCall	T
Salesrep	T
ShipTo	T
State	T

File-Name	Tbl-Type
Supplier	T
SupplierItemXref	T
TimeSheet	T
Vacation	T
Warehouse	T

Relation between _Db and _File

Now that we've figured out how to load schema tables from _File,
How do we model the relation between _File and _Db?

The relation between _Db and _File is

RECID(_Db) = _File._Db-recid

DATA-RELATION does not support RECID relations….

PDS PARENT-ID-RELATION _Db to _File


```
DEFINE DATASET pds-db-file
  FOR tt-db, tt-file

  PARENT-ID-RELATION db-file
      FOR   tt-db,
            tt-file
      PARENT-ID-FIELD _db-recid
      .
```

Specifies link between RECID **of** the parent table and a RECID **field in** a child table

PDS Read _db, _file


```
/* db-file-filter.p  */
DEFINE DATASET pds-db-file FOR tt-db, tt-file
   PARENT-ID-RELATION db-file
      FOR   tt-db,
            tt-file
      PARENT-ID-FIELD _db-recid
      .

BUFFER tt-db:ATTACH-DATA-SOURCE(DATA-SOURCE dsrc-db:HANDLE).
BUFFER tt-file:ATTACH-DATA-SOURCE(DATA-SOURCE dsrc-file:HANDLE).

DATA-SOURCE dsrc-file:FILL-WHERE-STRING
                                    = "WHERE _File._Tbl-Type = ""T""".

DATASET pds-db-file:FILL().

FOR EACH tt-db:
    DISPLAY tt-db._db-type tt-db._db-guid
      WITH FRAME f-db.
END.

FOR EACH tt-file:
    DISPLAY tt-file._file-name tt-file._tbl-type
        WITH DOWN.
END.
```

PDS Read _Db, _File

Db-type	Db guid
PROGRESS	la0ip5CbtIBbFJNZQAQbMQ

File-Name	Tbl-Type
Benefits	T
BillTo	T
Bin	T
Customer	T
Department	T
Employee	T
Family	T
Feedback	T
InventoryTrans	T
Invoice	T
Item	T
LocalDefault	T
Order	T
OrderLine	T
POLine	T
PurchaseOrder	T

tt-db

tt-file

Saving RECID(_Db) during PDS Read

- We have _File._Db-Recid in a TT
- We don't save RECID(_Db) in a TT
- What to do?
- Use event callbacks

Datasets have a set of standard events that can be linked to procedures:

Dataset:
- "BEFORE-FILL"
- "AFTER-FILL"

Dataset Buffer:
- "BEFORE-FILL"
- "AFTER-FILL"
- "BEFORE-ROW-FILL"
- "AFTER-ROW-FILL"
- "ROW-CREATE"
- "ROW-DELETE"
- "ROW-UPDATE"
- "FIND-FAILED"
- "SYNCHRONIZE"

Linking _Db to _File._Db-recid


```
/* db-file-filter-callback.p */
DEFINE TEMP-TABLE tt-db NO-UNDO
   LIKE  _Db
   FIELD _Db-recid AS RECID.

DATA-SOURCE dsrc-file:FILL-WHERE-STRING
                                = "WHERE _File._Tbl-Type = ""T""".

BUFFER tt-db:SET-CALLBACK-PROCEDURE("AFTER-ROW-FILL",
                                    "AfterRowFill",
                                    THIS-PROCEDURE).


PROCEDURE AfterRowFill:
DEFINE INPUT PARAMETER DATASET FOR pds-db-file.
ASSIGN tt-Db._Db-recid = RECID(_db).
END PROCEDURE.
```

Linking _Db to _File._Db-recid

tt-db

Db-type	Db guid	Db Recid
PROGRESS	Ia0ip5CbtIBbFJNZQAQbMQ	4032

tt-file

File-Name	Tbl-Type	Db Recid
Benefits	T	4032
BillTo	T	4032
Bin	T	4032
Customer	T	4032
Department	T	4032
Employee	T	4032
Family	T	4032
Feedback	T	4032
InventoryTrans	T	4032
Invoice	T	4032
Item	T	4032
LocalDefault	T	4032
Order	T	4032
OrderLine	T	4032
POLine	T	4032

```
/* db-file-filter-callback.p */
```

Bonus Round


```
/* db-db-detail-file-filter.p */
```

- Link _Db to _Db-Detail and _File
- Load the PDS TTs from the Schema VSTs
- Output the TT records

Congratulations!

You've learned the standard pattern for linking parent schema tables with child tables and how to save the resulting data into a dataset.

Here's the parent / child mapping for the Schema VSTs:

Parent	Child
_Db	_File
_File	_Field
_File	_Index
_Index	_Index-Field
_Index-Field	_Field

Notes:

- If a table has no indexes, that table's _Index record will not have any corresponding _Index-Field records.
- When _File._Prime-Index = RECID(_Index), that _Index is the primary index for that _File table
- See Gus's presentation for more details

Performance Related VSTs

From the Progress Docs

- After-image log activity (_ActAILog)
- Before-image log activity (_ActBILog)
- Buffer activity (_ActBuffer)
- Index activity (_ActIndex)
- Input/output activity (_ActIOFile)
- Input/output type activity (_ActIOType)
- Lock table activity (_ActLock)
- Other activity (_ActOther)
- Page writer activity (_ActPWs)
- Record activity (_ActRecord)
- Server activity (_ActServer)
- Space allocation activity (_ActSpace)
- Summary activity (_ActSummary)
- Area status (_AreaStatus)
- Area threshold (_AreaThreshold)
- Block (_Block)
- Buffer status (_BuffStatus)
- Cache (_Cache)
- Checkpoint (_Checkpoint)
- Code Features (_Code-Feature)
- Database connection (_Connect)
- Database features (_Database-Feature)
- Database startup parameters file status (_DbParams)
- Database status (_DbStatus)
- Database file status (_Filelist)
- Index statistics (_IndexStat)
- Latch statistics (_Latch)
- License management (_License)
- Lock table status (_Lock)
- Lock request (_LockReq)
- Logging (_Logging)
- Master block (_MstrBlk)
- User connection (_MyConnection)
- Resource queue statistics (_Resrc)
- Segments (_Segments)

From https://docs.progress.com/bundle/manage-database/page/Virtual-system-table-summaries.html

Performance Related VSTs

- Servers (_Servers)
- SQL query plan (_SQL_Qplan)
- Table statistics (_TableStat)
- Transaction (_Trans)
- Transaction end lock statistics (_TxeLock)
- User index activity (_UserIndexStat)
- Database input/output (_UserIO)
- Record locking table (_UserLock)
- User status (_UserStatus)
- User table activity (_UserTableStat)

Questions?

Thank you for your time!

Tim Kuehn

Email: tim.kuehn@tdkcs.com
Ph: 519-781-0081