ARCH-11: Designing a 3-tier framework based on the ProDataSet

Gunnar Schug

proALPHA Software

Content

proALPHA - Company and Product

OpenEdge® Reference Architecture Basics

An OpenEdge RA compliant framework

Summary

Company & Product – OERA Basics – Framework – Summary

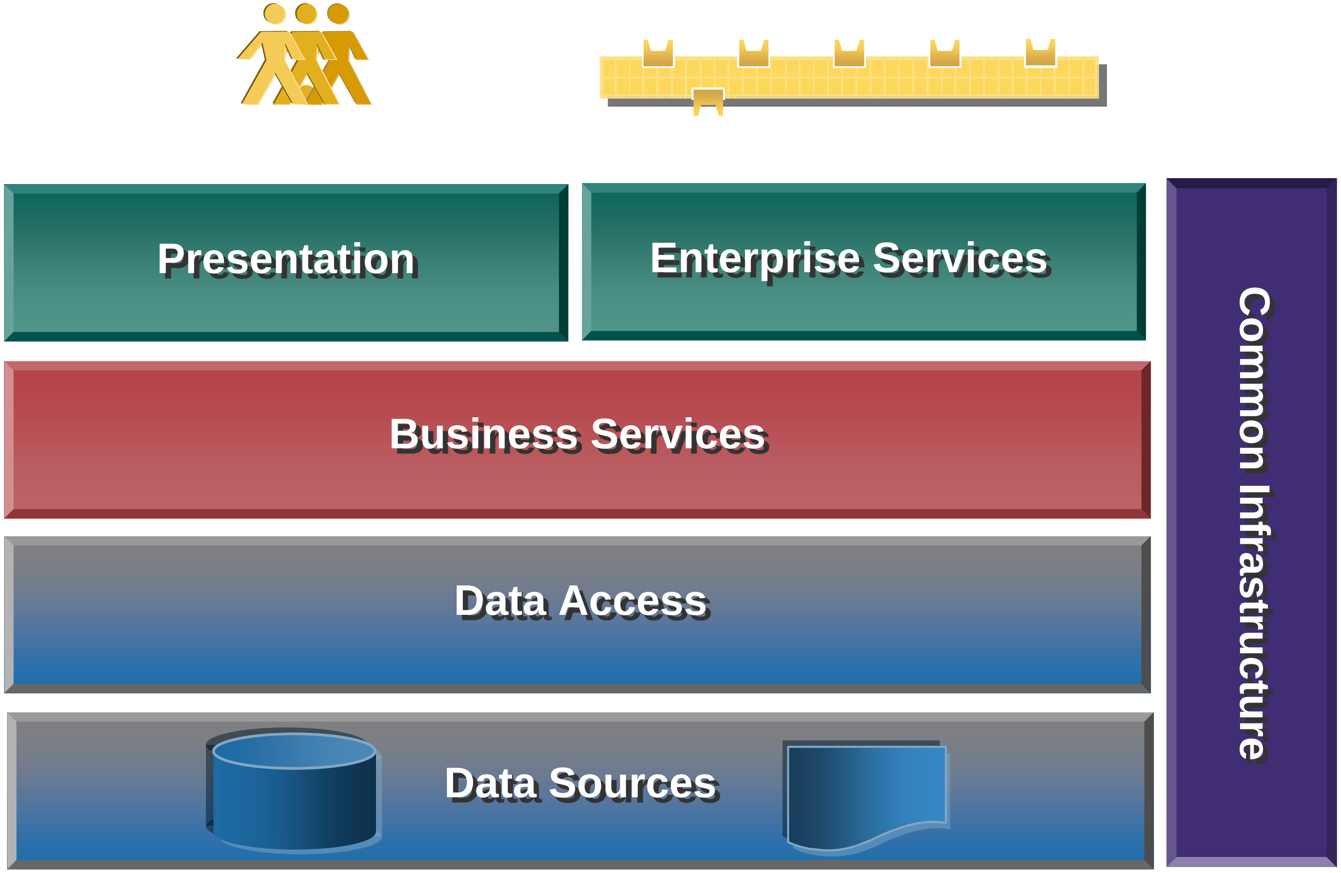
proALPHA
Company and Product

proALPHA – Partner for medium-sized Business

Product:	proALPHA® standard software	
Services:	IT consulting, project management, implementation, seminars, maintenance, support, and hotline	
Target market:	Medium-sized industrial and trade companies	
Business numbers:	Customer Base	> 1,100
	Sales 2003/2004	> 32.0 mill. € (e)
	EBIT	> 3.1 mill. € (e)
	Employees (as of 3/2005)	320
Managing board:	Leo Ernst	Commercial Management
	Werner Ernst	Technical Management
Supervisory board:	Dr. W. Wawrzinek, RA and WP in Hamburg	
	Prof. Dr. H. Müller-Merbach, University of Kaiserslautern	
	Dr. C. Segal, Berlin Capital Fund GmbH	
Shareholders:	Employees	58 %
	Berlin Capital Fund	21 %
	Others	21 %

OERA Basics

The well known picture...

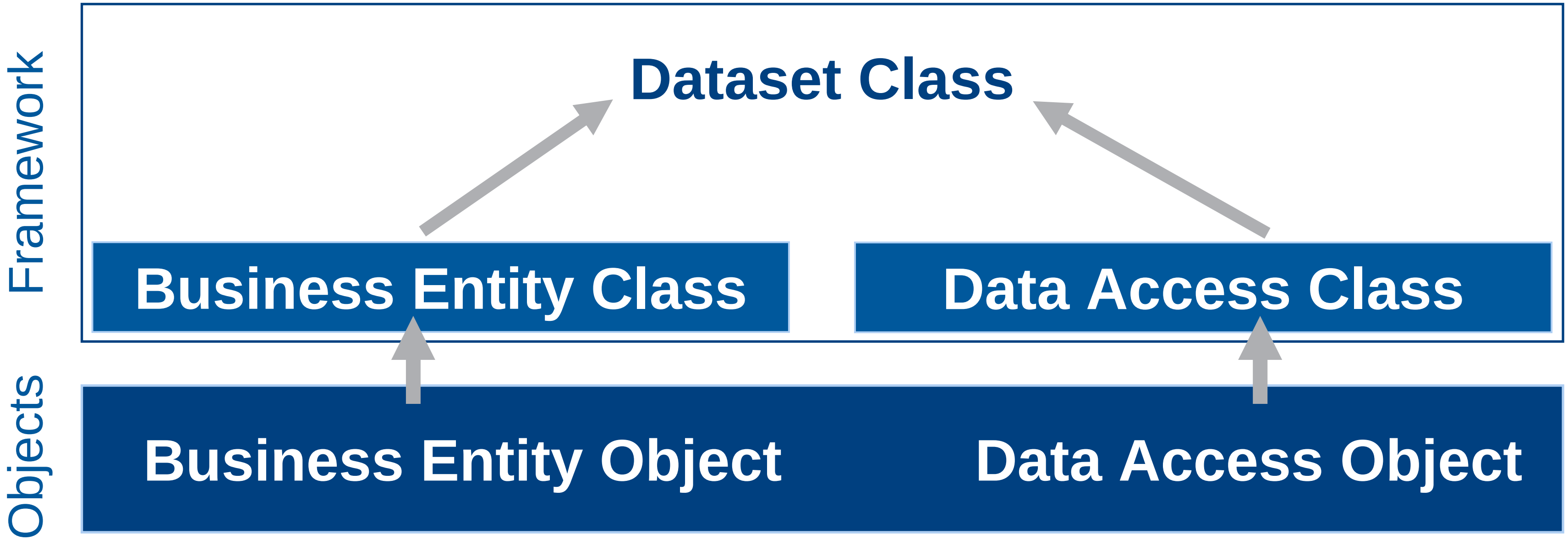


Company & Product – **OERA Basics** – Framework – Summary

Why do I need a framework?

- Provide common API
- Components instead of quick fixes
- Reduce brainless work
- Focus on Business Logic

Dataset Class Hierarchy

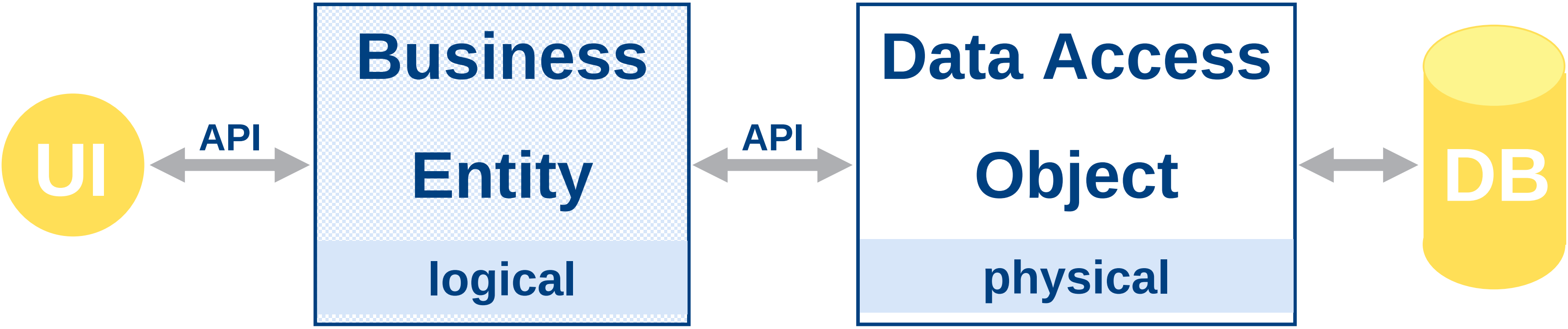


Dataset Class

- Initialization
- Tracking changes ON/OFF

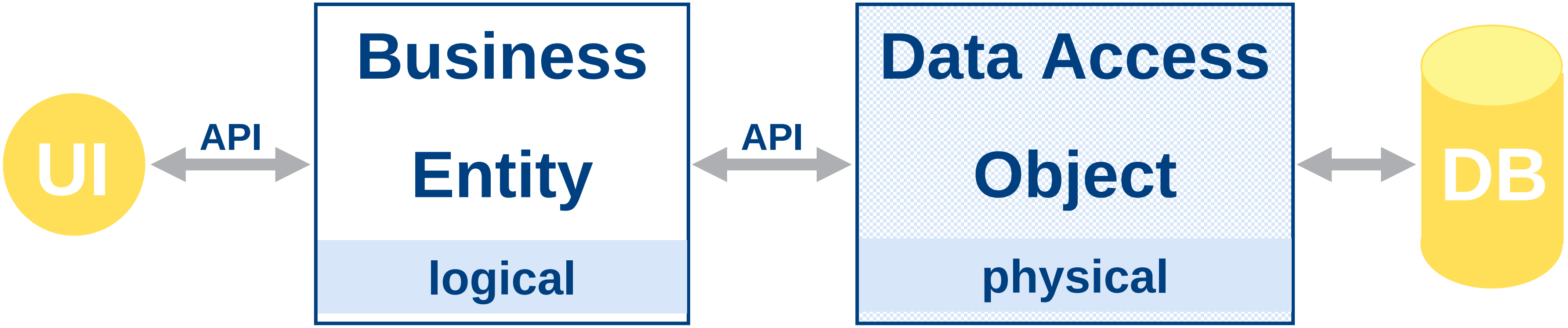
Business Entity Class

- Represents the „logical“ data view
- Provides the „API“ of an object
- Validations at logical level without DB-Connection



Data Access

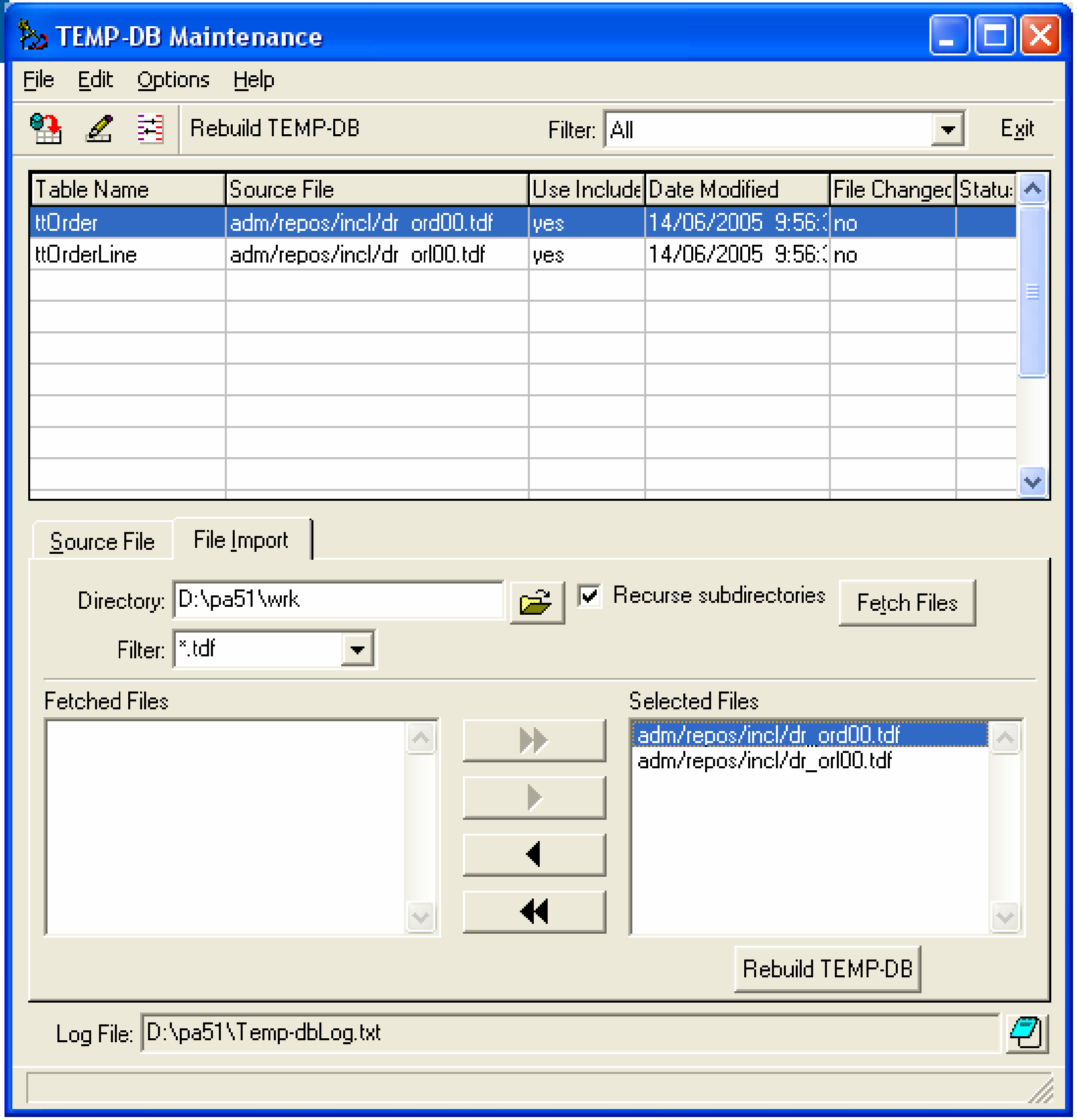
- Maps logical to physical view and vice versa
- Calculated Fields



ProDataSet Definition File

- Define it in an include file (.pds)
- Define each Temp-Table in one include file (.tdf)
- Use Temp-DB Maintenance Tool to integrate .tdfs in .pds and in visual objects
- Include .pds in every object that needs access to the dataset

Temp-DB Maintenance Tool – Best Practices



Filter: All

Table Name	Source File	Use Include	Date Modified	File Changec	Statu:
ttOrder	adm/repos/incl/dr_ord00.tdf	yes	14/06/2005 9:56::	no	
ttOrderLine	adm/repos/incl/dr_orl00.tdf	yes	14/06/2005 9:56::	no	

Directory: D:\pa51\wrk

☑ Recurse subdirectories

Filter: *.tdf

Selected Files:
adm/repos/incl/dr_ord00.tdf
adm/repos/incl/dr_orl00.tdf

Log File: D:\pa51\Temp-dbLog.txt

- Use unique extension to find ONLY the Temp-Table Definition files

- „Use Include“ allows you to change Temp-Table w/o metaschema changes in „Temp-DB“

Temp-Tables – Best practice

```
define buffer ttCustomer for ttCustomer.
create ttCustomer.
```

This would create records in the „Temp-DB“ Database!!!

Use the new „for temp-table“ option

```
define buffer ttCustomer for temp-table ttCustomer.
create ttCustomer.
```

Context Management Options

Additional Temp-Tab

- unlimited amount of data
- “travels” with the dataset e over session boundaries
- easy implementing of “get” and “set” functions

ute

rent

should care about different –numsep, -numdec, etc. settings in client and server

Who needs this???

But only in “input-output” mode!!!

Who cares? ☺

An OERA compliant framework

ProDataSet Definition Include (.pds)

- Use the Temp-DB Maintenance Tool with option „use include“ to define your Temp-Tables needed in the dataset

```
/* ***Included Temp-Table & Buffer definitions*** */

{adm/repos/incl/dr_ord00.tdf}
{adm/repos/incl/dr_orl00.tdf}
```

ProDataSet Definition Include (.pds)

- Then define your dataset

```
define dataset dsOrder
  for ttOrder, ttOrderLine

  data-relation drOrderLine for ttOrder,ttOrderLine
     relation-fields (OrderNum,OrderNum)
 .
```

ProDataSet Definition Include (.pds)

- Startup Business Entity Object persistent and store handle.
- Do not start BEO as a super procedure, you will fail if you need to include more than one .pds!!!
- Initialize Dataset now

```
define variable ghdsOrderBE as handle no-undo.
run orderbeo.p persistent set ghdsOrderBE.
run initializeObject in ghdsOrderBE.
```

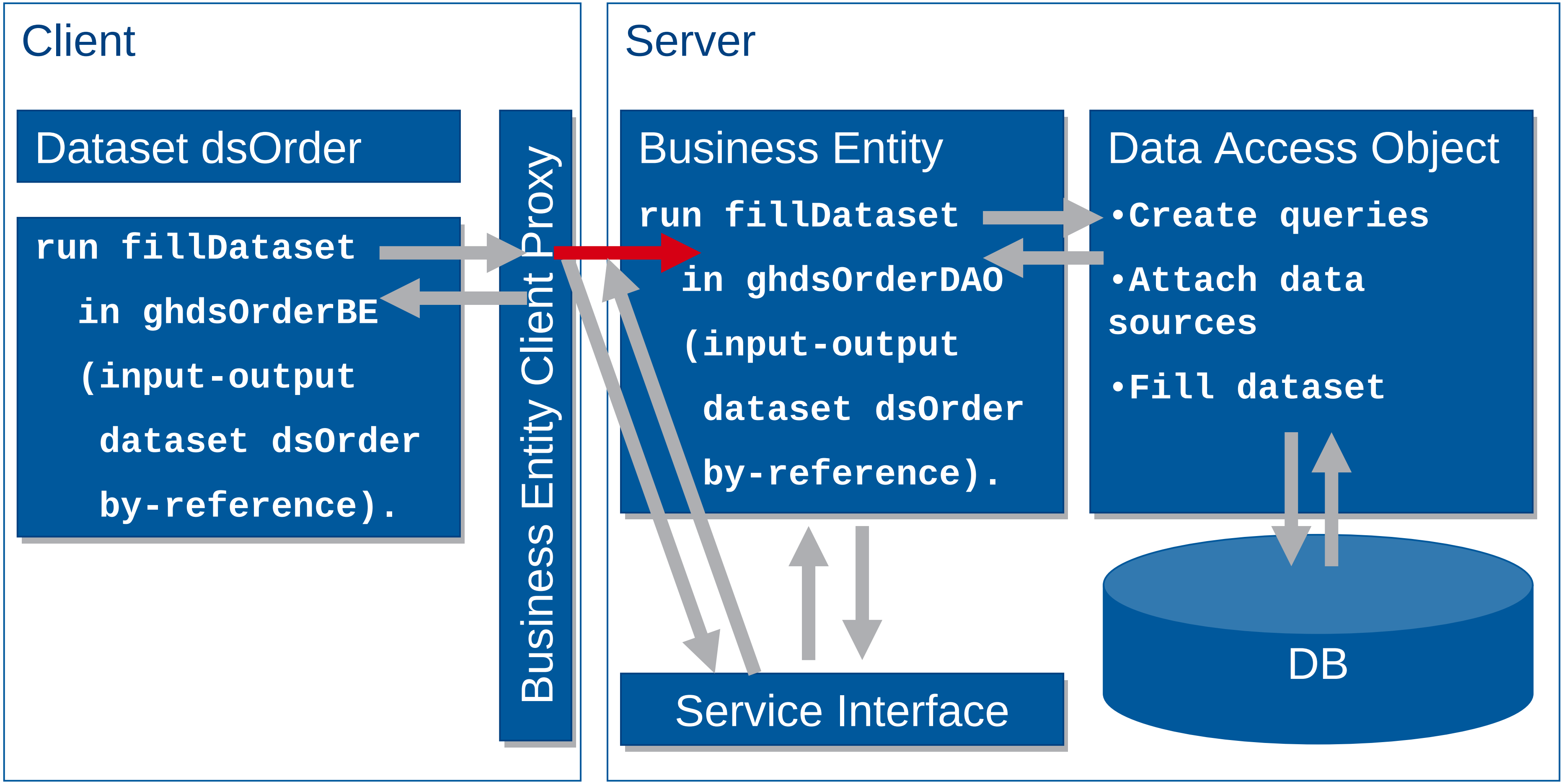
The Business Entity Class - Agenda

- Service Interface Layer
- Event handlers
- Fill Dataset
- Save Changes
- Batching

Service Interface Layer

- Reduce network traffic (pass only needed data to server)
  → Client processing is sometimes needed (e.g. „saveChanges“)
- Common environment used in all examples you‘ve ever seen is Client / Server
- The „user“ (e.g. a .p) of a ProDataSet might be an AppServer / WebSpeed Agent / etc. → „Client“ processing is STILL needed, but don‘t call another AppServer!
- “stateless” or “state-free”

Service Interface Layer



Client

Dataset dsOrder

```
run fillDataset
   in ghdsOrderBE
   (input-output
    dataset dsOrder
    by-reference).
```

Business Entity Client Proxy

Server

Business Entity

```
run fillDataset
   in ghdsOrderDAO
   (input-output
    dataset dsOrder
    by-reference).
```

Data Access Object

- Create queries
- Attach data sources
- Fill dataset

DB

Service Interface

The Business Entity Class - Agenda

- Service Interface Layer
- Event Handlers
- Fill Dataset
- Save Changes
- Batching

Client Event handlers

- Subscribe to „row-create“ event to
  - Assign Object ID
  - Copy primary fields from parent buffer
- Subscribe to „row-delete“ event to
  - Cascade deletion OR
  - Prevent deletion if dependent record exists

  Not to be used to ensure referential integrity of the database
- Within the generic event handlers, you can access the afore created or deleted buffer by using the „SELF“ handle

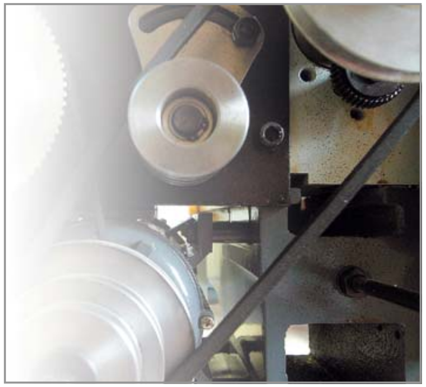
„row-create“ Event handler

Fill relation child fields by copying them from the parent record

```
if     valid-handle(self:parent-relation)
   and self:parent-relation:parent-buffer:available then
do:
  cRelationFields = self:parent-relation:relation-fields.
  do i = 1 to num-entries(cRelationFields) by 2:
    self:buffer-field(entry(i + 1,cRelationFields)):buffer-value
      = self:parent-relation:parent-buffer:buffer-field(entry(i,cRelationFields)):buffer-value.
  end.
end.
```

The Business Entity Class - Agenda

- Service Interface Layer
- Event handlers
- Fill Dataset
- Save Changes
- Batching



Fill Dataset Client Processing

- “native” approach “output append” will produce runtime errors in case of duplicate records
- → Create an empty dataset and merge results

```
define variable hPDS as
create dataset hPDS.
hPDS:create-like(iophDataset
run fillDatasetSV on gshAppS    er
  (input-output dataset-hand  e hPDS by-reference).
iophDataset:copy-dataset(hPDS,yes).
```

Online Help calls it “append” flag, but it is a “merge” flag

The Business Entity Class - Agenda

- Service Interface Layer
- Event handlers
- Fill Dataset
- Save Changes
- Batching

Save Changes Client Processing

- Create a new identical d ta
- Pass this dataset to the

Always use name prefix

```
create dataset hPDS.
hPDS:create-like(iophDataset,’pacd’:U).
hPDS:get-changes(iophDataset,yes).
<save Changes on Server>
```

Also retrieve unchanged “master” records for validation

Save Changes Client Processing (contd.)

- Everything ok in server processing? → Merge Changes at client side

Same value as get-parent-mode in get-changes

```
if hPDS:error = no then
  hPDS:merge-changes(iophDataset,yes).
else
  …
```

Save Changes Client Processing (contd.)

- Error occurred? → Show messages

```
create query hQuery.
do i = 1 to hPDS:num-buffers:
  hBuffer = hPDS:get-buffer-handle(i):before-buffer.
  hQuery:set-buffers(hBuffer).
  hQuery:query-prepare('for each ':U + hBuffer:name).
  hQuery:query-open().
  hQuery:get-first().
  do while hBuffer:available:
    if    hBuffer:rejected
      and hBuffer:error-string > '':U then
      message hBuffer:error-string view-as alert-box.
    hQuery:get-next().
  end.
end.
```

The Business Entity Class - Agenda

- Service Interface Layer
- Event handlers
- Fill Dataset
- Save Changes
- Batching

Batching

- Sounds “easy”, but which records should be retrieved in the “next batch”???
  - Next 50 orders
  - Next 50 order lines of current order
  - Next 50 orders including all order lines
  - Next 50 orders including first 50 order lines
  - …
- We decided: “nextBatch” should have 1 parameter “batchtable”
  → return next batch of *n* records for that table

Batching

- **ProDataSet support for Batching**
  - **“batch-size” stops fill process after *n* records**
  - **“next-rowid” rowid of *n*th record**
  - **“restart-rowid” starting point for fill query**

Simply use “next-rowid” of last batch

Batching

- What's the capacity of your client's memory – or how many properties do you want to store?
  - Store the last rowid only (table and record independent)
  - Store the last rowid per table
  - Store the last rowid per record
- We decided to store the last rowid per table

The Data Access Class - Agenda

- Calculated Fields
- Generic Queries and Data Sources
- Save Changes

Calculated Fields

- Define them in your Temp-Table
- Fill them in an Even
- Naming convention

  events

Find internal entries in super procedures as well

```
cInternalEntries = <internal entries including super
  procedure>
if can-do(cInternalEntries,iophDataset:name +
  'BeforeFill':U) then
  iophDataset:set-callback-procedure
    ('Before-Fill':U,
     iophDataset:name + 'BeforeFill':U,
     target-procedure).
...
```

Calculated Fields - Example

```
procedure ttOrderAfterRowFill :
  define input parameter dataset for dsOrder.
  for each Order
    where Orderline.OrderNum = ttOrder.OrderNum
    no-lock:
    ttOrder.OrderTotal
      = ttOrder.OrderTotal
        + Orderline.ExtendedPrice.
  end.
end procedure. /* ttOrderAfterRowFill */
```

Ease your life with naming conventions

Company & Product – OERA Basics – **Framework** – Summary

The Data Access Class - Agenda

- Calculated Fields
- Generic Queries and Data Sources
- Save Changes

Generic Queries and Data Sources

Tasks to fill a Dataset

- Attach Data Sources
- Prepare Queries
- Fill
- Detach Data Sources

Generic creation and attachment of data sources

- For each dataset member buffer that should be filled (I.e. fill-mode <> “no-fill”)
  - Create a query
  - Create a data source for the query
  - Attach data source to member buffer
- Activate Callback procedures

Generic creation and attachment of data sources (contd.)

```
do i = 1 to iophDataset:num-buffers:
 if not valid-handle(iophDataset:get-buffer-handle(i):data-source)
  and iophDataset:get-buffer-handle(i):fill-mode <> 'no-fill':U
 then
 do:
  cBufferName = substring(iophDataset:get-buffer-
 handle(i):name,3).
  create buffer hBuffer for table cBufferName no-error.
  if valid-handle(hBuffer) then
  do:
   create query hQuery.
   hQuery:set-buffers(hBuffer).
   create data-source hDataSource.
   hDataSource:query = hQuery.
   iophDataset:get-buffer-handle(i):attach-data-source
    (hDataSource,<fieldmapping>,…).
  end.
 end.
end.
…
```

Generic Query Preparation

- **Use “fill-where-string” for a complete fill and add your individual constraints**

```
do i = 1 to iophDataset:num-buffers:
  hBuffer = iophDataset:get-buffer-handle(i).
  if hBuffer:fill-mode <> 'no-fill':U then
    hBuffer:data-source:query:query-prepare
      ('for each ':U
       + hBuffer:data-source:query:get-buffer-
  handle(1):name
       + hBuffer:data-source:fill-where-string
       + <table specific constraints>).
end.
```

Generic Data Source Detachment

- Detach Data Sources
- Garbage Collection

```
do i = 1 to iophDataset:num-buffers:
  hDataSource
    = iophDataset:get-buffer-handle(i):data-source
  no-error.
  if valid-handle(hDataSource) then
    hQuery = hDataSource:query no-error.
  if valid-handle(hQuery) then
    hBuffer = hQuery:get-buffer-handle(1) no-error.
  iophDataset:get-buffer-handle(i):detach-data-source().
  delete object hBuffer     no-error.
  delete object hQuery      no-error.
  delete object hDataSource no-error.
end.
```

The Data Access Class - Agenda

- **Calculated Fields**
- **Generic Queries and Data Sources**
- Save Changes

```
ine E-Mail!</p>
PHP_SELF ?>" method="post">
<input type="text" name="Ma
len?<br>
aft" cols="50" rows="5" w

Mail abschicken">

il != "") {
"Sie haben Post!",
schaft wurde weiterg
```

Save Changes

- Save all changes in 1 transaction
- Attach data sources just like during the fill process
- Start at top level buffers and use child relations for a recursive storage
- In case of any error
  - Undo the whole transaction
  - Set buffer “rejected” = “yes”
  - Use buffer “error-string” to return appropriate message to the client
- See ProDataSet manual for generic code

Summary

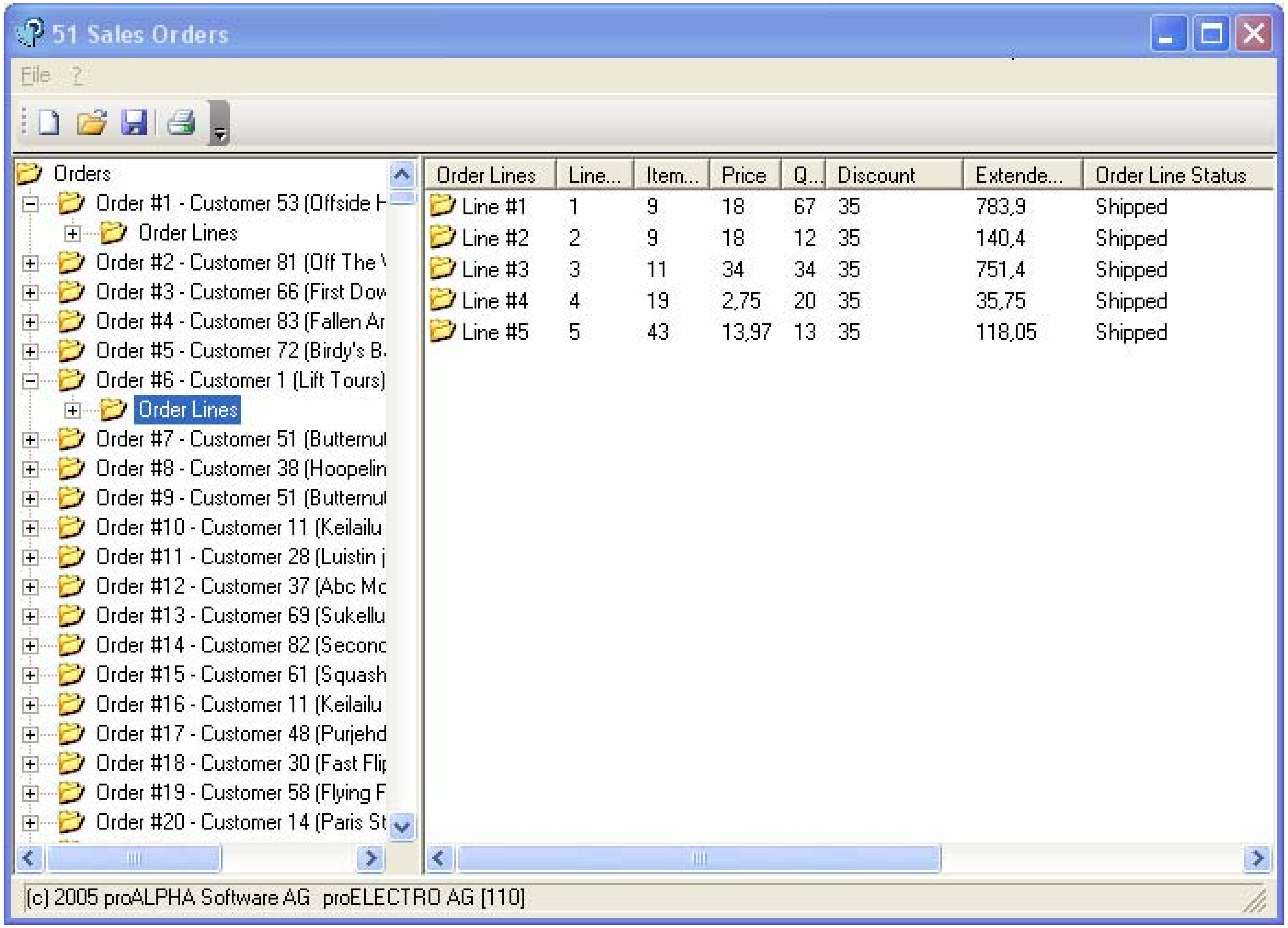
adulation

- ProDataSets…
  - …are very fast
  - …are easy to maintain – if you have an OERA compliant framework
  - …support very generic programming
  - …are one of the greatest features Progress ever introduced

but

- Still missing
  - “recursive” relations, e.g. parts lists etc.

Examples



51 Sales Orders

- Orders
  - Order #1 - Customer 53 (Offside H
    - Order Lines
  - Order #2 - Customer 81 (Off The \
  - Order #3 - Customer 66 (First Dow
  - Order #4 - Customer 83 (Fallen Ar
  - Order #5 - Customer 72 (Birdy's B.
  - Order #6 - Customer 1 (Lift Tours)
    - Order Lines
  - Order #7 - Customer 51 (Butternut
  - Order #8 - Customer 38 (Hoopelin
  - Order #9 - Customer 51 (Butternut
  - Order #10 - Customer 11 (Keilailu
  - Order #11 - Customer 28 (Luistin j
  - Order #12 - Customer 37 (Abc Mc
  - Order #13 - Customer 69 (Sukellu
  - Order #14 - Customer 82 (Seconc
  - Order #15 - Customer 61 (Squash
  - Order #16 - Customer 11 (Keilailu
  - Order #17 - Customer 48 (Purjehd
  - Order #18 - Customer 30 (Fast Flip
  - Order #19 - Customer 58 (Flying F
  - Order #20 - Customer 14 (Paris St

Order Lines	Line...	Item...	Price	Q...	Discount	Extende...	Order Line Status
Line #1	1	9	18	67	35	783,9	Shipped
Line #2	2	9	18	12	35	140,4	Shipped
Line #3	3	11	34	34	35	751,4	Shipped
Line #4	4	19	2,75	20	35	35,75	Shipped
Line #5	5	43	13,97	13	35	118,05	Shipped

(c) 2005 proALPHA Software AG proELECTRO AG [110]

Questions ?