Cincom

FROM 2.0 TO 3.0 AND ONWARD: DATABASE TO GLORP - WHAT'S NEW

Niall Ross, David Wallen and Yuwei Li


Cincom
SMALLTALK
esug
CONFERENCE
Whiskies: Talisker, Elbe and Ardbeg
and lavender shortbread (and Clan Ross single malt) – in exchange for FEEDBACK

EXDI 3.0 Drivers

EXDI is VisualWorks' External Database Interface

+ Gemstone ? !

- **ODBC:** SQLServer (and Oracle, MySQL, …, even PostgreSQL) ^
  - ODBC 2.0: old, still supported
  - ODBC 3.0: in preview (VW 8.1), supported (VW 8.2)

- **PostgreSQL:** GLORP's reference database
  - PostgreSQL 2.0: old API, goodie, sockets API+ EXDI facade
  - PostgreSQL 3.0: latest API, thread-robust, Cincom-supported
    - sockets API: PostgresSocketConnection (or PostgreSQLEXDIConnection)
    - the C API: PostgresLibpqConnection (or PostgresLibpqThapiConnection)

PostgreSQL 3.0 survives Inspection ☺

A socket is read in the order that results are added!

- **old Postgres driver**: forked sessions on same connection ☹
  - E.g. inspect while developing, or fork while running, etc.
    session readOneOf: StorePackage where: [:pkg| pkg id = 123456]
  - could return data for a StoreBlob, a StoreMethod, … anything! - and DNU
- **new Postgres driver**: forked sessions on same connection ☺
  - The C API: maintains session-specific internals => no problem
  - Sockets API: same server issue => thread-safe lazy buffering

```
[first1 := (a1 := session1 execute; answer) next] forkAt: … “high priority” .
[first2 := (a2 := session2 execute; answer) next] forkAt: … “low priority” .
…                          session1 buffers when session2 executes
a1 next.   “session1 is now reading from its buffer”
a2 next.   “session2 is still reading from the shared connection’s socket”
```

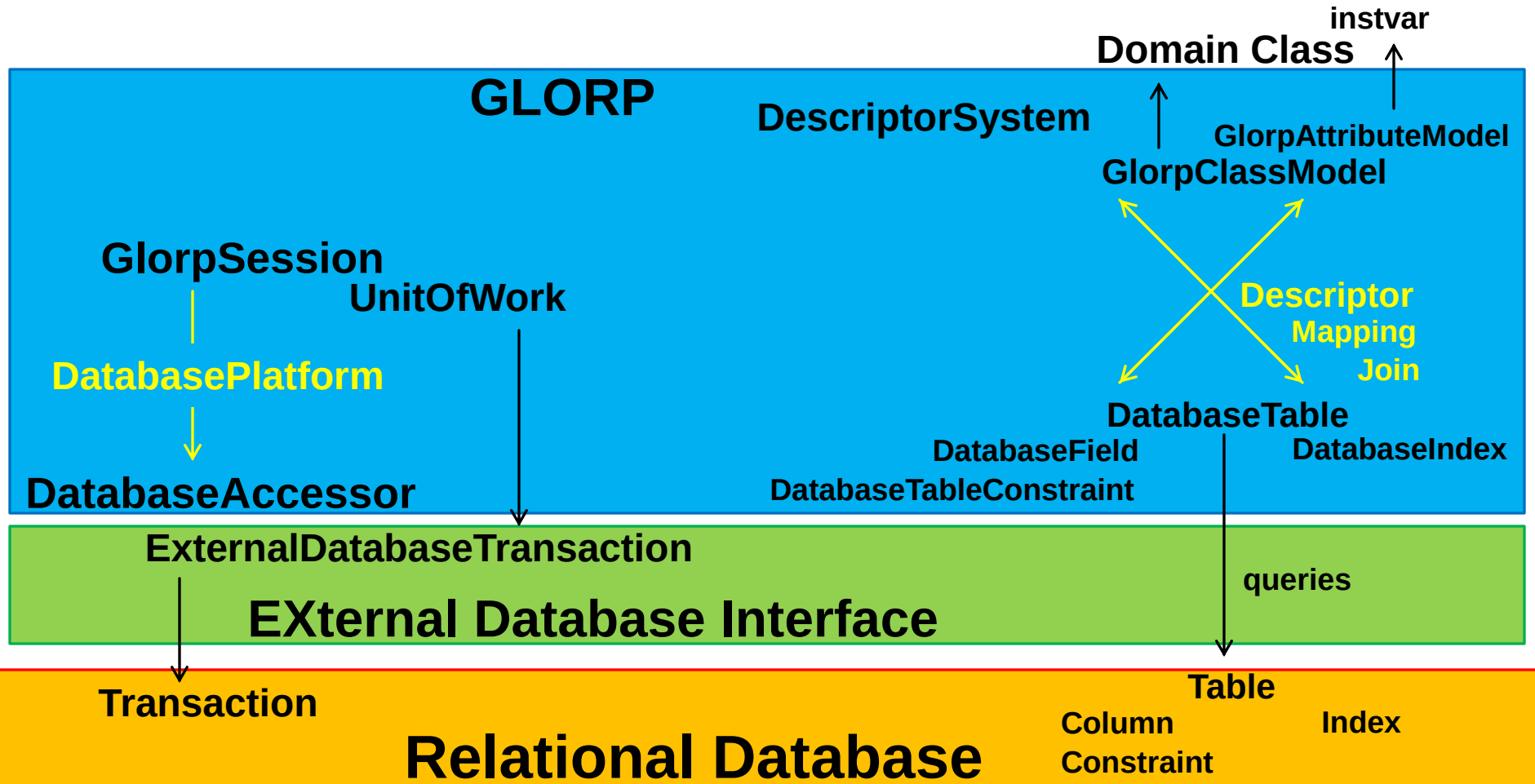
GLORP is…

- **Generic**: abstract, declarative, multi-platform framework
- **Lightweight**: looks like Smalltalk

```
session read: Customer where: [:each | each orders size > 0]
```

- **Object**: general hierarchic OO models
  - no ActiveRecord-like style restriction
  - remains flexible *throughout* the lifecycle
- **Relational**: embedded or bound SQL
- **Persistence**: transactions rollback in sync in DB and image

Database <-> EXDI <-> GLORP <-> AppeX



Mapping GLORP to EXDI

- **EXDI framework:** ExternalDatabaseConnection
  - the wire protocol: convert Smalltalk data to server i/o
  - class hierarchy: a subclass per database i/o protocol {version}

- **GLORP framework:** DatabasePlatform
  - the SQL protocol: convert generic SQL to DB-specific SQL
    - “SQL is a standard: that means it differs everywhere” (Alan Knight)
  - class hierarchy: a subclass per SQL dialect

- **A database login needs both:** was 1:n, will be m:n
  - (so, e.g. our Store login UI will evolve)

Mappings: GLORP deduces all it can

```
classModelForDriver: aClassModel
        …
        aClassModel newAttributeNamed: #vehicles collectionOf: Vehicle.

descriptorForDriver: aDescriptor
        …
        aDescriptor manyToManyMapping
                attributeName: #vehicles;
                referenceClass: Vehicle.
                usesLinkTable: true;
                join: (Join from: (driverTable fieldNamed: 'ID')
                                to: (driverVehicleTable fieldNamed: 'DRIVER_ID'));
                linkFields: (Array with: (driverVehicleTable fieldNamed: 'VEHICLE_ID'));
                reverseJoin: (Join from: (driverVehicleTable fieldNamed: 'VEHICLE_ID')
                                        to: (vehicleTable fieldNamed: 'ID')).
```

Mappings: GLORP deduces all it can

```
classModelForDriver: aClassModel
    ...
    aClassModel newAttributeNamed: #vehicles collectionOf: Vehicle.

descriptorForDriver: aDescriptor
    ...
    aDescriptor manyToManyMapping
        attributeName: #vehicles;
        referenceClass: Vehicle.
        usesLinkTable: true;
        join: (Join from: (driverTable fieldNamed: 'ID')
                    to: (driverVehicleTable fieldNamed: 'DRIVER_ID'));
        linkFields: (Array with: (driverVehicleTable fieldNamed: 'VEHICLE_ID'));
        reverseJoin: (Join from: (driverVehicleTable fieldNamed: 'VEHICLE_ID')
                        to: (vehicleTable fieldNamed: 'ID')).
```

Mappings: GLORP deduces all it can

```
classModelForDriver: aClassModel
    ...
    aClassModel newAttributeNamed: #vehicles collectionOf: Vehicle.

descriptorForDriver: aDescriptor
    ...
    aDescriptor manyToManyMapping
        attributeName: #vehicles;
        referenceClass: Vehicle.
        usesLinkTable: true;
        join: (Join from: (driverTable fieldNamed: 'ID')
                    to: (driverVehicleTable fieldNamed: 'DRIVER_ID'));
        linkFields: (Array with: (driverVehicleTable fieldNamed: 'VEHICLE_ID'));
        reverseJoin: (Join from: (driverVehicleTable fieldNamed: 'VEHICLE_ID')
                        to: (vehicleTable fieldNamed: 'ID')).
```

Mappings: GLORP deduces all it can

```
classModelForDriver: aClassModel
    ...
    aClassModel newAttributeNamed: #vehicles collectionOf: Vehicle.

descriptorForDriver: aDescriptor
    ...
    aDescriptor manyToManyMapping
        attributeName: #vehicles;
        referenceClass: Vehicle.
        usesLinkTable: true;
        join: (Join from: (driverTable fieldNamed: 'ID')
                    to: (driverVehicleTable fieldNamed: 'DRIVER_ID'));
        linkFields: (Array with: (driverVehicleTable fieldNamed: 'VEHICLE_ID'));
        reverseJoin: (Join from: (driverVehicleTable fieldNamed: 'VEHICLE_ID')
                        to: (vehicleTable fieldNamed: 'ID')).
```

Mappings: GLORP deduces all it can

```
classModelForDriver: aClassModel
    ...
    aClassModel newAttributeNamed: #vehicles collectionOf: Vehicle.

descriptorForDriver: aDescriptor
    ...
    aDescriptor manyToManyMapping
        attributeName: #vehicles;
        referenceClass: Vehicle.
        usesLinkTable: true;
        join: (Join from: (driverTable fieldNamed: 'ID')
                    to: (driverVehicleTable fieldNamed: 'DRIVER_ID'));
        linkFields: (Array with: (driverVehicleTable fieldNamed: 'VEHICLE_ID'));
        reverseJoin: (Join from: (driverVehicleTable fieldNamed: 'VEHICLE_ID')
                            to: (vehicleTable fieldNamed: 'ID')).
```

Mappings: GLORP deduces all it can

```
classModelForDriver: aClassModel
    ...
    aClassModel newAttributeNamed: #vehicles collectionOf: Vehicle.

descriptorForDriver: aDescriptor
    ...
    aDescriptor manyToManyMapping
        attributeName: #vehicles;
        referenceClass: Vehicle.
        usesLinkTable: true;
        join: (Join from: (driverTable fieldNamed: 'ID')
                    to: (driverVehicleTable fieldNamed: 'DRIVER_ID'));
        linkFields: (Array with: (driverVehicleTable fieldNamed: 'VEHICLE_ID'));
        reverseJoin: (Join from: (driverVehicleTable fieldNamed: 'VEHICLE_ID')
                            to: (vehicleTable fieldNamed: 'ID')).
```

Mappings: GLORP deduces all it can

```
classModelForDriver: aClassModel
    ...
    aClassModel newAttributeNamed: #vehicles collectionOf: Vehicle.

descriptorForDriver: aDescriptor
    ...
    aDescriptor manyToManyMapping
        attributeName: #vehicles;
        referenceClass: Vehicle.
        usesLinkTable: true;
        join: (Join from: (driverTable fieldNamed: 'ID')
                    to: (driverVehicleTable fieldNamed: 'DRIVER_ID'));
        linkFields: (Array with: (driverVehicleTable fieldNamed: 'VEHICLE_ID'));
        reverseJoin: (Join from: (driverVehicleTable fieldNamed: 'VEHICLE_ID')
                            to: (vehicleTable fieldNamed: 'ID')).
```

GLORP: (a little of) what's new, what's fixed

- **OK to compare composite primaryKeys:** = <> in:

```
session read: User where:
    [:each || query | query := Query read: User where: [:user | user … ].
    (each in: query) AND: … ].
```

- **OK to alsoFetch: a toMany relation:** safe to limit query, preserves order

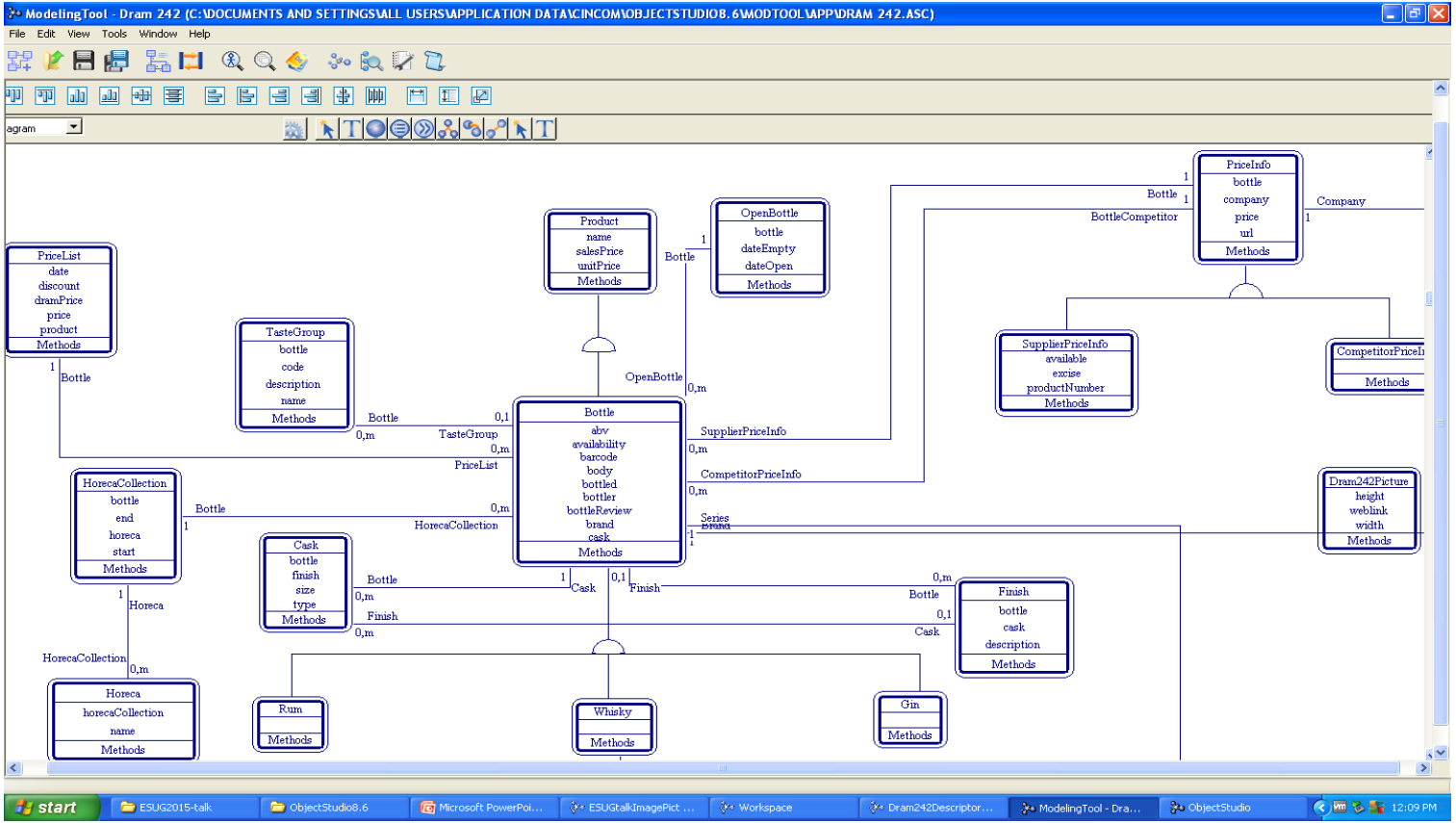
```
query := (Query readOneOf: Customer where: [ … ])
        alsoFetch: #accounts;
        …
```

- **OK to nest iterators:** in queries and in class hierarchy filters

```
session readOneOf: User where: [:user | user name = 'Niall' AND:
        [user orders anySatisfy: [:order | order lineItems anySatisfy:
                [:item | item bonuses anySatisfy: [:bonus | bonus credits > 50]]]]].
```

- **And more:**
  - dictionary mapping: Key can be complex, not in scope of Owner -> linkTable -> Value
  - rollbackAndContinue (and more literals are exempt)
  - …

Mapping: The Full Process (ObjectStudio)



Mapping: The Simple Quick-Start Process (VW)

- **[ObjectStudio Modeling and Mapping Tools:**
  - map DB schema to GLORP system, on to domain model
    - and vice-versa: model problem, map to GLORP system, create DB schema
  - understand domain, improve model, refine mapping to DB
  - can deploy on VisualWorks (e.g. for non-Windows target) **]**

- **The GlorpAtlasUI:** quick-started VisualWorks mapping tool
  - simple tabular UI, view schema or schema fragment
  - read database, generate first-cut GLORP system
  - generate domain classes (or map existing classes)

- **(Onward to the Web:** AppeX Scaffolding does the rest)

What Have We Seen

- **External Database Interface:** what's new
  - Postgres 3.0, ODBC 3.0 supported
    - ExternalDatabaseConnection many-many to GLORP's DatabasePlatform
- **Writing GLORP:** some fixes, additions
  - GLORP systems: minimalism
  - GLORP queries: comparisons
- **GlorpAtlasUI:** demo
  - from legacy database schema to GLORP system
    - quick-start, user-guided
- **Over Lunch, the GLORP doctor is in:** and so is the whisky!
  - New UI thoughts also welcome – or any other OS/VW feedback

Questions?

Contact Information

Technical Details

- nross, dwallen, yli@cincom.com **Glorp, EXDI**
- tkogan, jkott, vdegen@cincom.com **AppeX and SiouX**

Contact Information

Star Team (Smalltalk Strategic Resources)

- **Suzanne Fortman** (sfortman@cincom.com)
  Cincom Smalltalk Program Director
- **Arden Thomas** (athomas@cincom.com)
  Cincom Smalltalk Product Manager
- **Jeremy Jordan** (jjordan@cincom.com)
  Cincom Smalltalk Marketing Manager
- **Suzanne Fortman** (sfortman@cincom.com)
  Cincom Smalltalk Engineering Manager

Try Cincom Smalltalk

Evaluate Cincom Smalltalk:

- **try.cincomsmalltalk.com**

Join our Cincom Smalltalk Developer Program:

- **develop.cincomsmalltalk.com**