epcc

The OGSA-DAI Client Toolkit

Amy Krause
a.krause@epcc.ed.ac.uk

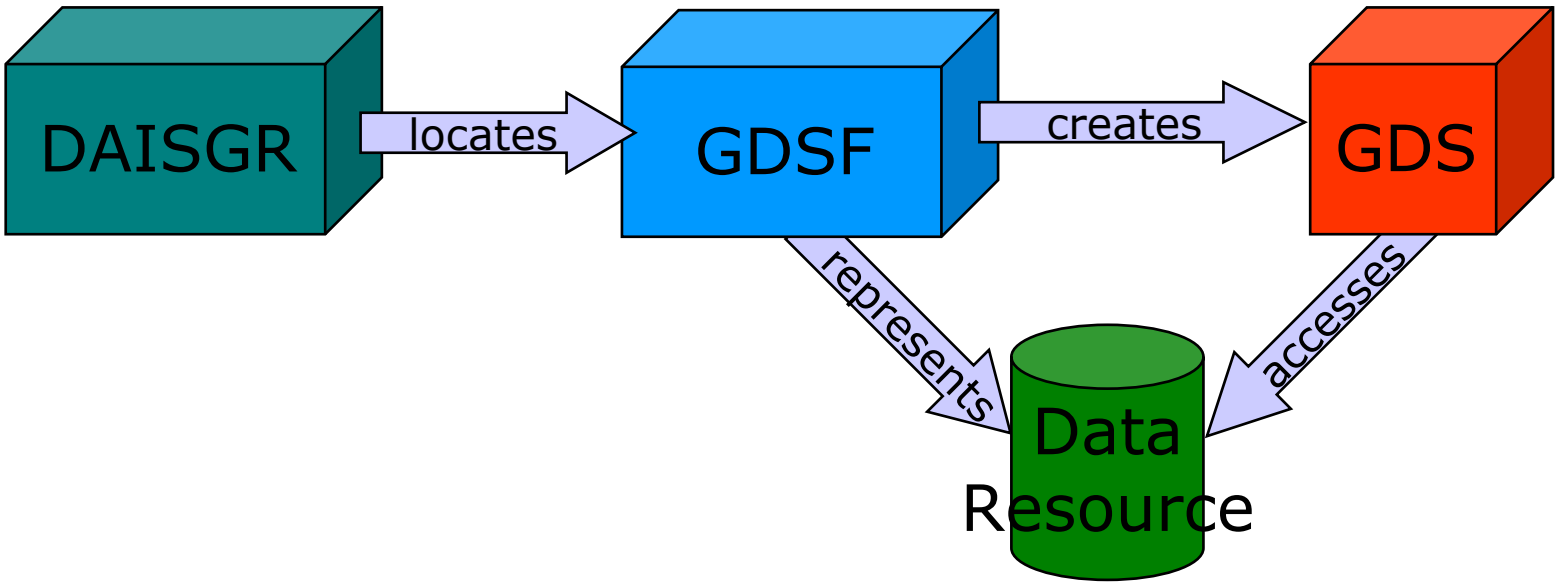
Overview

- The Client Toolkit
- OGSA-DAI Service Types
- Locating and Creating Data Services
- Requests and Results
- Data Integration

Why use a Client Toolkit?

- Nobody wants to write XML!
- Users aren't concerned about the connection mechanism
- Client API hides changes in the service implementation

- OGSA-DAI uses three main service types
  - DAISGR (registry) for discovery
  - GDSF (factory) to represent a data resource
  - GDS (data service) to access a data resource



- The ServiceFetcher class creates service objects from a URL

```
ServiceGroupRegistry registry =
      ServiceFetcher.getRegistry( registryHandle );
GridDataServiceFactory factory =
      ServiceFetcher.getFactory( factoryHandle );
GridDataService service =
      ServiceFetcher.getGridDataService( handle );
```

Registry

- A registry holds a list of service handles and associated metadata
- For example, clients can query a registry for all registered Grid Data Factory Services

```
GridServiceMetaData[] services =
    registry.listServices(
          OGSADAIConstants.GDSF_PORT_TYPE );
```

- The *GridServiceMetaData* object contains the handle and the port types that the factory implements

```
String handle = data.getHandle();
QName[] portTypes = data.getPortTypes();
```

- A factory object can create a new Grid Data Service.

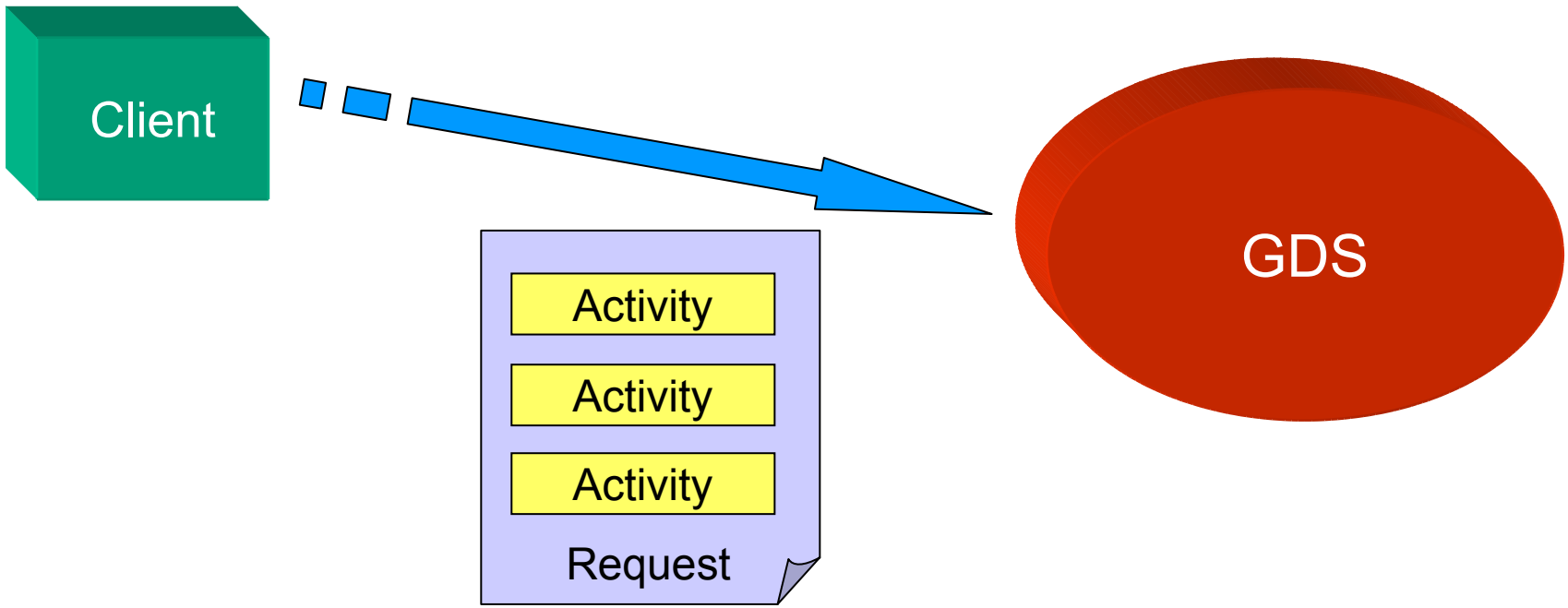
```
GridDataService service =
        factory.createGridDataService();
```

- Grid Data Services are transient (i.e. have finite lifetime) so they can be destroyed by the user.

```
service.destroy();
```

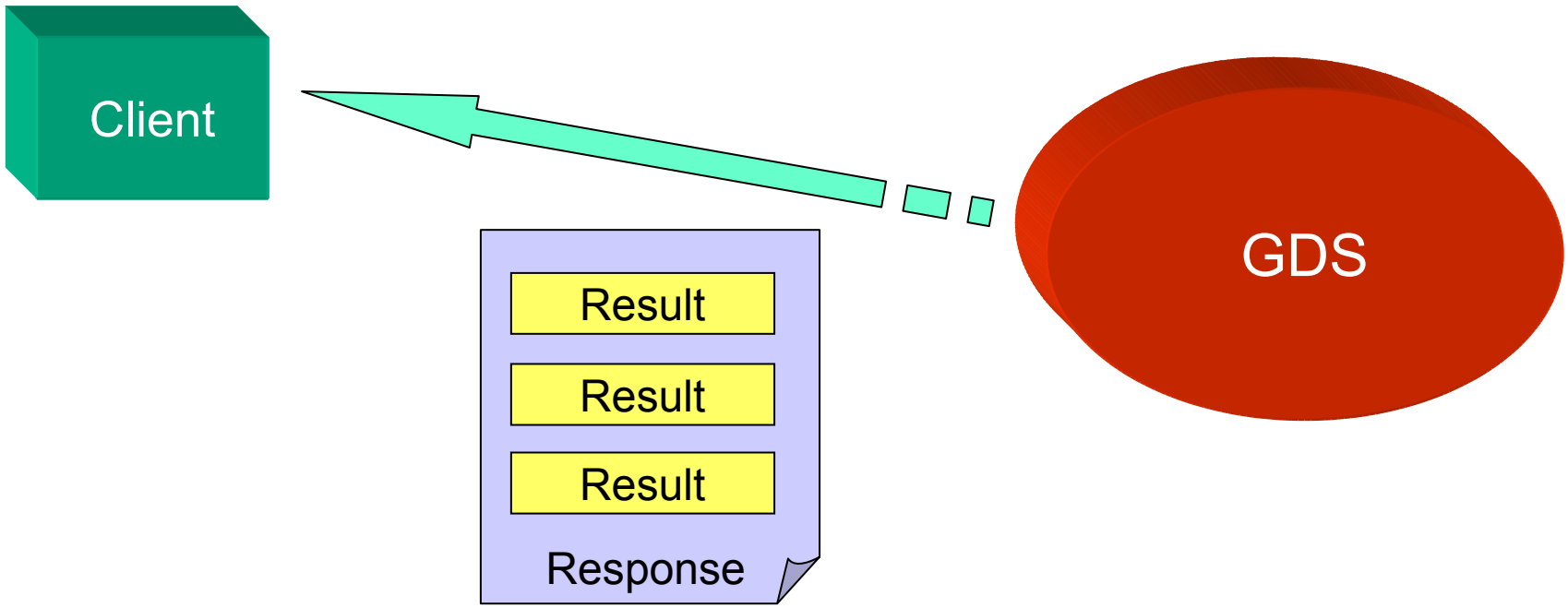
Interaction with a GDS

- Client sends a request to a data service
- A request contains a set of activities



Interaction with a GDS

- The Data service processes the request
- Returns a response document with a result for each activity



Activities and Requests

- A request contains a set of activities
- An activity dictates an action to be performed
  - Query a data resource
  - Transform data
  - Deliver results
- Data can flow between activities

Examples of Activities

- SQLQuery

```
SQLQuery query = new SQLQuery(
   "select * from littleblackbook where id='3475'");
```

- XPathQuery

```
XPathQuery query = new XPathQuery( "/entry[@id<10]" );
```

- XSLTransform

```
XSLTransform transform = new XSLTransform();
```

- DeliverToGFTP

```
DeliverToGFTP deliver = new DeliverToGFTP(
      "ogsadai.org.uk", 8080, "myresults.txt" );
```

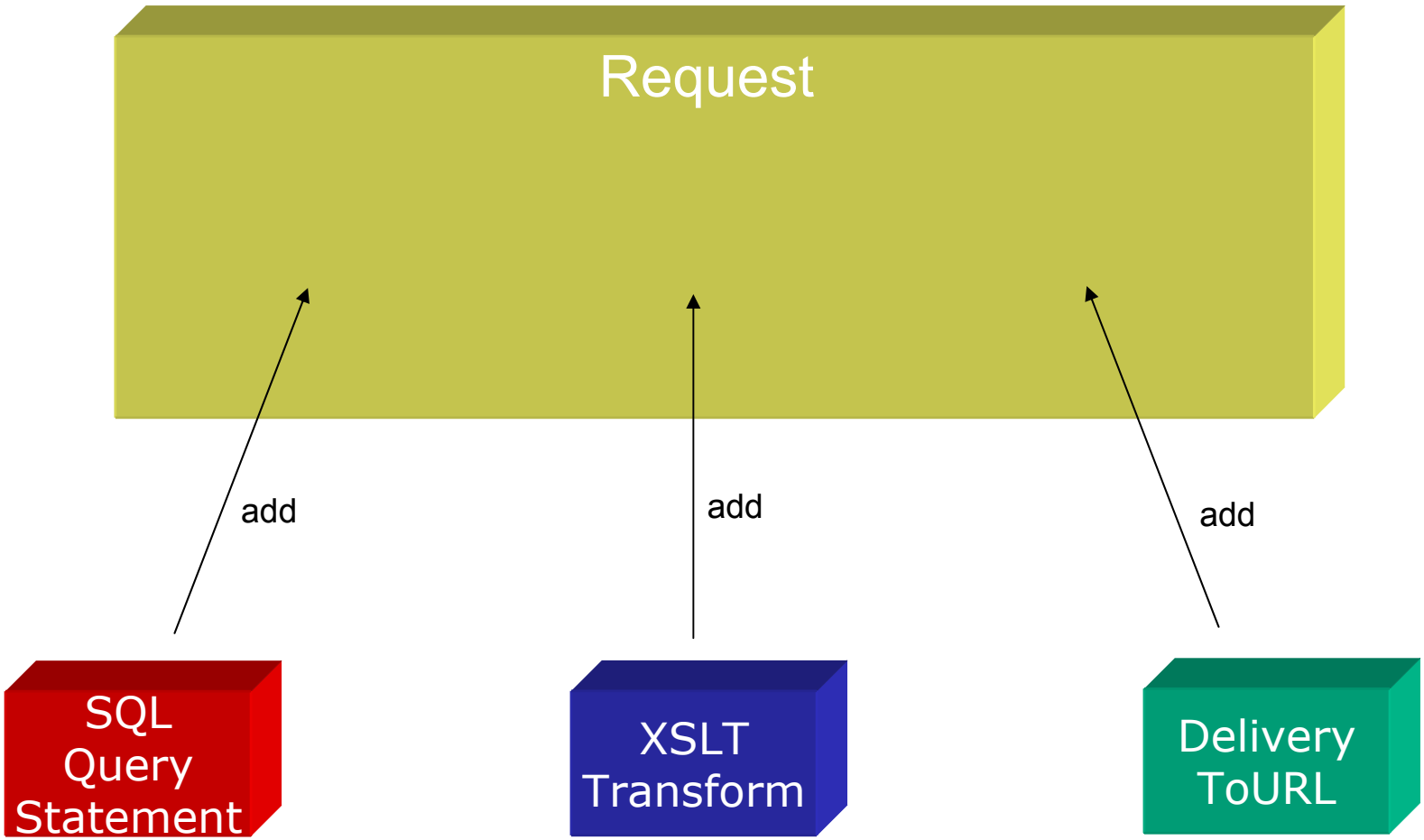
Simple Requests

- Simple requests consist of only one activity
- Send the activity directly to the perform method

```
SQLQuery query = new SQLQuery(
   "select * from littleblackbook where id='3475'");

Response response = service.perform( query );
```

Constructing a Request



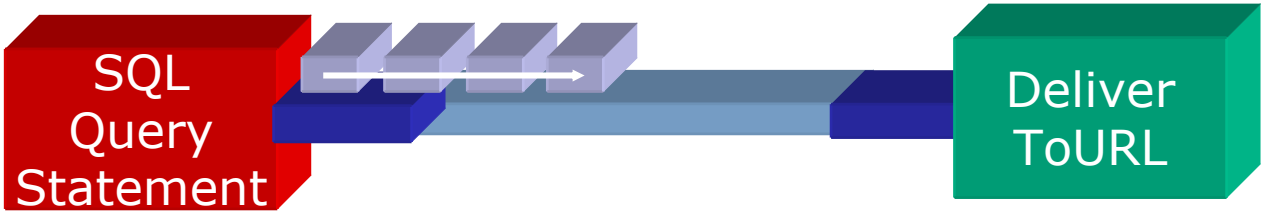
Constructing a Request cont.


```
ActivityRequest request = new ActivityRequest;
request.add( query );
request.add( transform );
request.add( delivery );
```

Connecting activities

```
SQLQuery query = new SQLQuery(
   "select * from littleblackbook where id<=1000");
DeliverToURL deliver = new DeliverToURL( url );

deliver.setInput( query.getOutput() );
```



Performing Requests

- Finally… perform the request!

```
Response response = service.perform( Request );
```

- The response contains status and results of each activity in the request.

```
System.out.println( response.getAsString() );
```

- Varying formats of output data
  - SQLQuery
    - JDBC ResultSet:

```
ResultSet rs = query.getResultSet();
```

  - SQLUpdate
    - Integer:

```
int rows = update.getModifiedRows();
```

  - XPathQuery
    - XML:DB ResourceSet:

```
ResourceSet results = query.getResourceSet();
```

- Output can always be retrieved as a String

```
String output = myactivity.getOutput().getData();
```

- Data can be pulled from or pushed to a remote location.
- OGSA-DAI supports third-party transfer using *FTP*, *HTTP*, or *GridFTP* protocols.

```
DeliverToURL deliver = new DeliverToURL( url );
deliver.setInput( myactivity.getOutput() );
```

```
DeliverToGFTP deliver = new DeliverToGFTP(
    "ogsadai.org.uk", 8080, "tmp/data.out" );
deliver.setInput( myactivity.getOutput() );
```

Delivery Activities

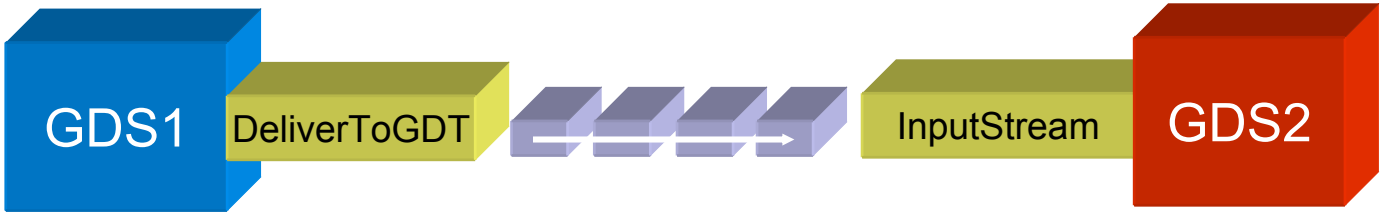
- The *DeliverFromURL* and *DeliverToURL* activities transfer data to/from a remote location.

```
DeliverFromURL deliver = new DeliverFromURL( url );
myactivity.setInput( deliver.getOutput() );
```

- Supported protocols are *http*, *ftp*, and *file*.
- Other delivery activities:
  - DeliverFromGFTP/DeliverToGFTP
  - DeliverToStream

Delivering data to another GDS

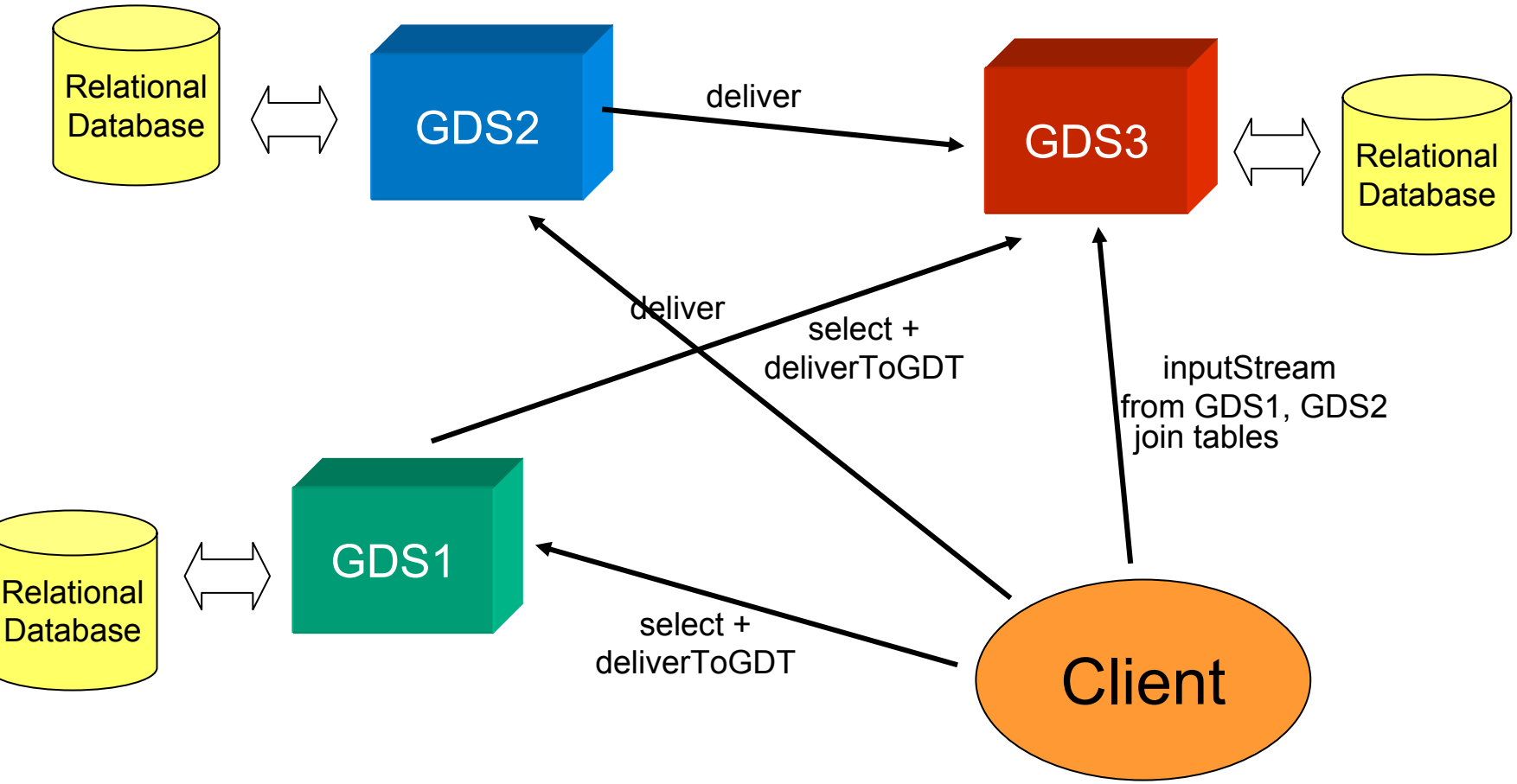
- The GDT port type allows to transfer data from one data service to another.
- An *InputStream* activity of GDS1 connects to a *DeliverToGDT* activity of GDS2
- Alternatively, an *OutputStream* activity can be connected to a *DeliverFromGDT* activity



Delivering Data

- Transfer in blocks or in full
- *InputStream* activities wait for data to arrive at their input
- Therefore, the *InputStream* activity at the sink has to be started before the *DeliverToGDT* activity at the source
- Same for *OutputStream* and *DeliverFromGDT*

Data Integration Scenario



Conclusion

- Easy to use
- No XML !
- Service implementation is hidden from the user