FDO: Street Vendors in the Cathedral

Frank Warmerdam
Mateusz Loskot

FDO GDAL Provider
FDO PostGIS Provider

OSGeo.org
The Open Source Geospatial Foundation

Feature Data Objects History

Initial technology requirements

- Object-Oriented and Component-based architecture
- Provider-based data source access
- Capabilities API support
- OGC Simple Features geometry model
- Logical model definition and mapping to physical model

FDO API 1.0 – spring 2004

FDO API 3.0 released as Open Source in 2006

- together with MapGuide Open Source

Feature Data Objects Releases - 1

FDO 1.0 - spring 2004

- First providers: Oracle, SDF
- Products:
  - Autodesk Map 3D 2005

FDO 2.0

- New providers: ArcSDE
- Products:
  - Autodesk Map 3D 2006

Feature Data Objects Releases - 2

FDO 3.0

- New providers:
  - OGC: WFS, WMS
  - RDBMS: MySQL, SQL Server, ODBC
  - other: SHP, Raster
- Products:
  - Autodesk Map 3D 2007
  - MapGuide Open Source 1.0.0

FDO 3.1

- New providers: ?
- Products:
  - Autodesk MapGuide Enterprise 2007
  - MapGuide Open Source 1.0.1

Feature Data Objects Availability

FDO versions:

- Enterprise
  - contains Open Source + proprietary components
  - released as a part of Autodesk Map 3D and Autodesk MapGuide Enterprise products
- Open Source
  - contains only Open Source components
  - released as source code available to download from OSGeo website

Supported platforms:

- Windows
- Linux

Why FDO Open Source?

FDO has been moved to Open Source for the same reasons as MapGuide:

- Fulfill developers and customers needs
  - Faster development of innovative web solutions
  - More frequent releases
  - Lower costs
- Map serving technologies popularization
- Join and support Open Source Geospatial community

FDO Open Source License

GNU Lesser General Public License (LGPL)

- Use and distribute without any royalties or license fees
- Source code modifications remain open
- Copyleft license
- Business-firiendly license
- Clear models of usage:
  - work that uses the library
  - work that is based on the library
- Release under the LGPL once and forever

What is the FDO API?

FDO API is a generic interface for accessing and manipulating GIS data regardless where it is stored.

- Defines extensible interface
- Uses provider-based model of data source access
- Providers are extensible with new capabilities and commands
- Supports capabilities requests
- Defines hierarchical feature data storage
- Defines object-based logical schema model describing data with classes and properties
- Implemented using C++ language

FDO Core – Data concept - 1

- **Data concept** – constructs of data mapping to the FDO API
  - Feature – represents real world object, with or without geographical location
  - Schema – a metadata, defines data types used to model objects. Technically, schema is constituted by a set of classes and class' properties.
  - Schema Element – defines a type of data, such as a class or property of a feature, or an association between schema elements
  - Schema Override – defines rules to override default schema mappings
  - Schema Mapping – defines correspondence between a Schema Element and its physical representation in a data store

FDO Core – Data concept - 2

- **Data concept**:
  - Class Type – represents any type of data within a feature schema
  - Feature Class – describes a type of real world object, includes class name and properties definition
  - Property – 5 kinds of properties:
    - association - used to model a peer-to-peer
    - relationship between two classes
    - geometric – represents geometry
    - object - a complex property type used within a class
    - raster - information needed to process a raster image
    - data - a non-spatial property: Boolean, Byte, DateTime, Decimal, Double, Int16, Int32, Int64, Single (float), String, BLOB, CLOB

FDO Core – Operational concept

- **Operational concept** – operations that are used to manipulate the data
  - Command – used to perform operations on features
  - Expression – a basic component used to build up a filters or larger expressions
  - Filter – build up with expressions to identify a subset of objects in a data store.
  - Lock – used to gain exclusive control on update operations
    - Transaction lock – a series of commands is treat as a single atomic unit. Long Transactions support.
    - Persisent lock – applied and removed manually by a user

FDO Feature Data Model - Vectors

Geometry API:

- Formats:
  - AGF: little-endian only; 2D, 3D and 4D geometries supported with additional flag; additional geometry types
  - WKB: defines byte order for every geometry; 2D only
- FdoGeometryType enum:
  - Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, MultiGeometry
  - CurveString, CurvePolygon, MultiCurveString, MultiCurvePolygon

FDO Feature Data Model - Rasters

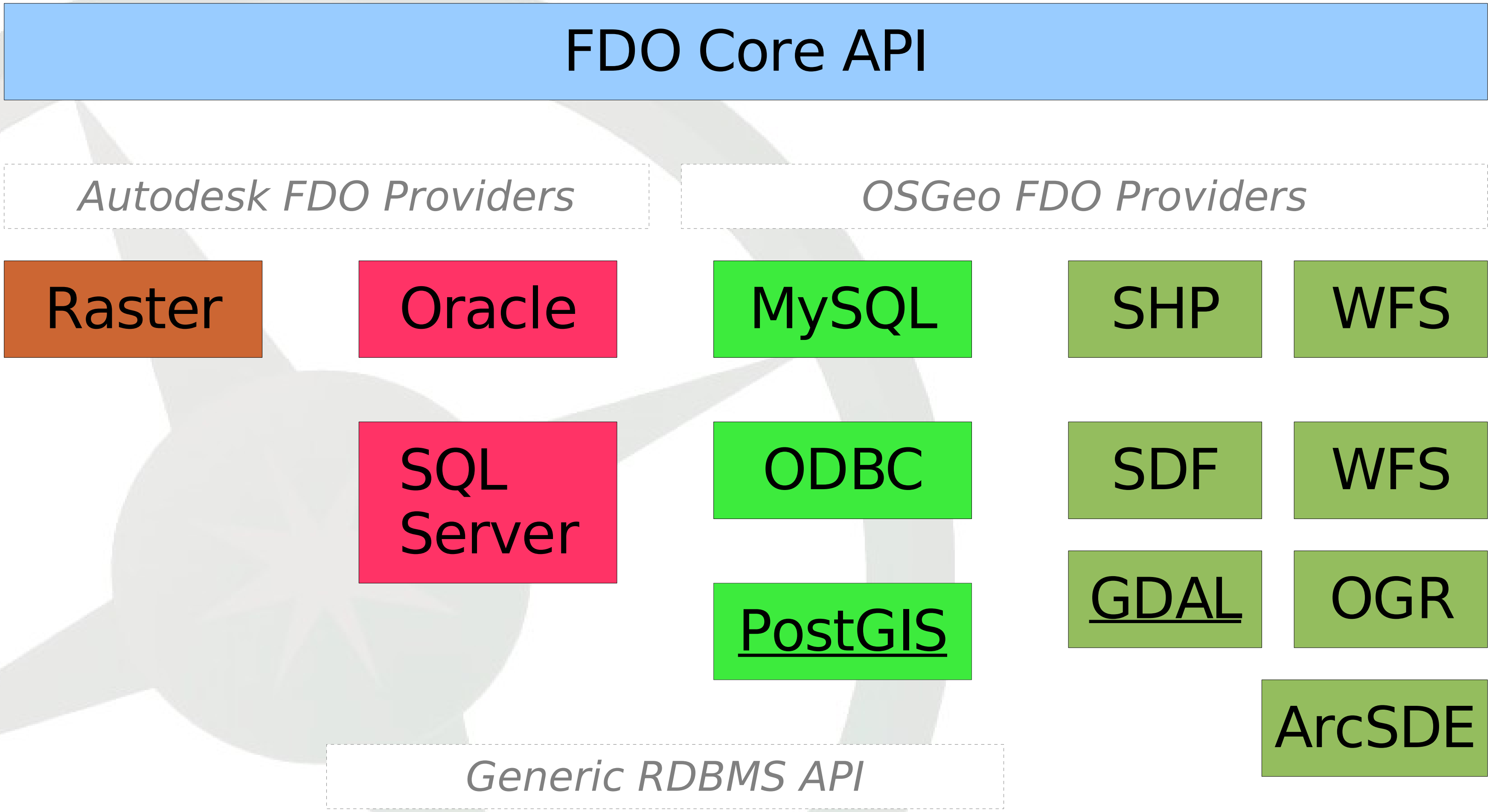
- *FdoDataType_Raster is a table column type*
- *Tables of rasters can be initialized from xml config files or a directory of files*
- *Pixel samples can be:*
  - *UnsignedInteger, SignedInteger, Float*
  - *SizeInBits: 1-64*
- *“Color” model: Data, Bitonal, Grey, RGB, RGBA, Palette*
- *A band selection mechanism for multispectral exists, but I don't understand it.*
- *Returned binary streams can be tiled, and interleaved in various ways*
- *Geolocation is based on an insertion point, pixel size and rotation (ie. GDAL geotransform / ESRI World file)*

What is the FDO Provider?

FDO Provider:

- Is a specific implementation of the FDO API to provide access to particular data store
- Provides functionality to directly access and manipulate data in a data store (file or database)
- Is called indirectly through the common FDO API
- Separate software component distributed as a shared loadable library

Feature Data Objects – base components



FDO: Impressions from an Outsider

- It is very RDBMS oriented (schemas, SelectCommand, connection strings, transactions)
- It feels big! FDO and GDAL/OGR each roughly 300KLOC, but FDO feels big (>1000 classes in core+fdordbms!)
- It uses many modern C++ idioms (templates, smart pointers, etc)
- It aims to be “enterprise ready”. RDBMS support, thread safety, wide strings pretty much everywhere
- It tries to be comprehensive in potential support for features of different data sources (ie. Broad set of field types, filtering features)

Fdogdal : Development Experience

- I found FDO to be intimidating, but really once I spent a few days buried in it, it becomes quite managable.
- Windows build problems were at times hard to trace
- Debugging problems in unittests was hard, especially on linux
- It was very important to have a provider to clone most of the “machinery” from for all the schema stuff, config file parsing, etc.
- Generally it was hard to navigate the many classes – perhaps I cloned too much?
- Unit test framework very helpful
- Lack of commandline tools for interactive testing.
- Too scared to test in mapguide yet. :-)

Fdopostgis : Development Experience

- The FDO uses a generic driver for RDBMS access which is implemented in a kind of “legacy“ C language.
- PostGIS installation requires a few manual steps, which are hard to implement as FDO API operation
- FDO consists of a big number of abstractions that I found hard to understand but they're required and powerful at the end.
- I found the FDO build system a little static and not configurable
- Huge number of classes and relations between them
- Code is NOT grouped into namespaces
- Very long names of C++ types and functions
- Very long functions call chains
- So, it's hard to familiarize with the the FDO source code
- Debugging is hard

Fdopostgis : Development Experience

- You have to be brave and don't stop digging into the code until you understand its all details well ;-)

FDO: Comparisons with OGR

FDO Advantages:

- Has a fuller set of field types than OGR.
- Supports much more sophisticated capabilities checks
- Supports multiple geometry columns
- Supports fuller set of spatial and attribute queries
- Supports transactions

FDO Disadvantages:

- No commandline user tools (like ogrinfo, ogr2ogr)
- Arguable more complicated to write an application
- Arguable more complicated to write a provider
- Less portable (only 32bit linux and win32 currently)
- Seemingly harder to build, especially flexibly
- Arguably “less approachable and understandable”

FDO: Comparisons with GDAL

FDO Advantages:

- Nice model for RDBM stored rasters – no analog in GDAL
- Convenient SQLish access to select rasters to operate on
- Convenient way to convert Grey/RGB/RGBA.

FDO Disadvantages:

- More complicated to write an application
- Much more complicated to write a provider
- No support for complex pixel types (ie. CInt16)
- Odd “band” model for multispectral data.
- Lacks GCP, RPC and geolocation array models for image geometry
- Difficult to write a general client because so many provider capabilities are optional

Questions?