



Implementing Enterprise Applications with the Geodatabase

Gudmundur Hafberg

Tom Brown

Brent Pierce



Please!

turn OFF cell phones
and refrain from using
flash photography

Assumptions

- **Good working knowledge of the Geodatabase**
- **Experience in programming against the GDB API**
- **Code examples will use C#.NET**
 - Readily translatable to VB, VB.NET, VC++, etc.
 - Code samples available with the slides on EDN after the Summit
 - Code samples in the presentation might not compile
 - (shortened to improve readability)
- **Lots of content, little time**
 - Please hold all questions to the end
 - We'll be at the Tech Talk for questions
 - Also Island area for the remainder of the summit

What is covered in this session

- **Connecting to the geodatabase**
- **Working with Schemas**
- **Geodatabase cursor model**
- **Querying Data**
 - Spatial / Non-Spatial
 - Views and Joins
- **Transactions and Editing**
 - Transactions
 - Versioned / Non-Versioned
- **Difference Cursors**
- **Events**
- **Code Instrumentation and Tracing**



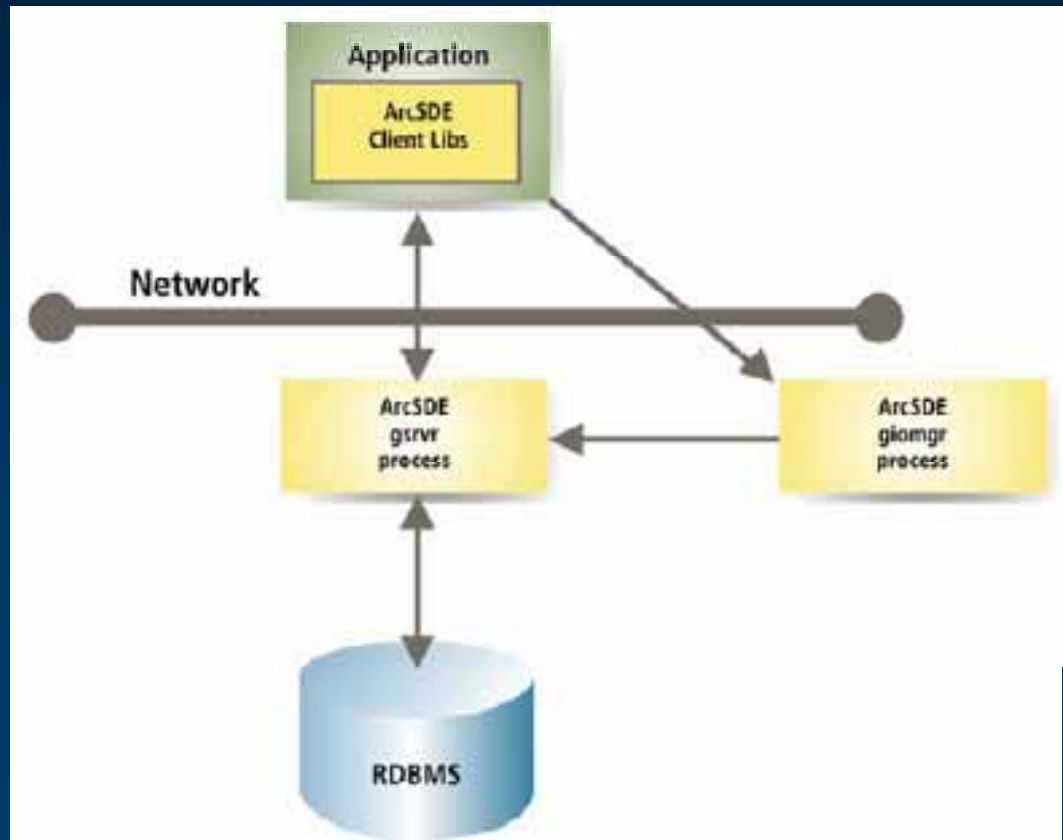
Workspaces

Connections and workspaces (Enterprise)

- **The workspace is the DBMS connection and gives access to the data**
- **Each connection/workspace creates an ArcSDE and DBMS process.**
 - Type of connection controls server/client load
- **ArcGIS maintains a workspace/connection pool to save resources**
 - Workspaces are uniquely instanced
- **Opening too many connections consumes server and DBMS resources**

Database Connection using Application Server

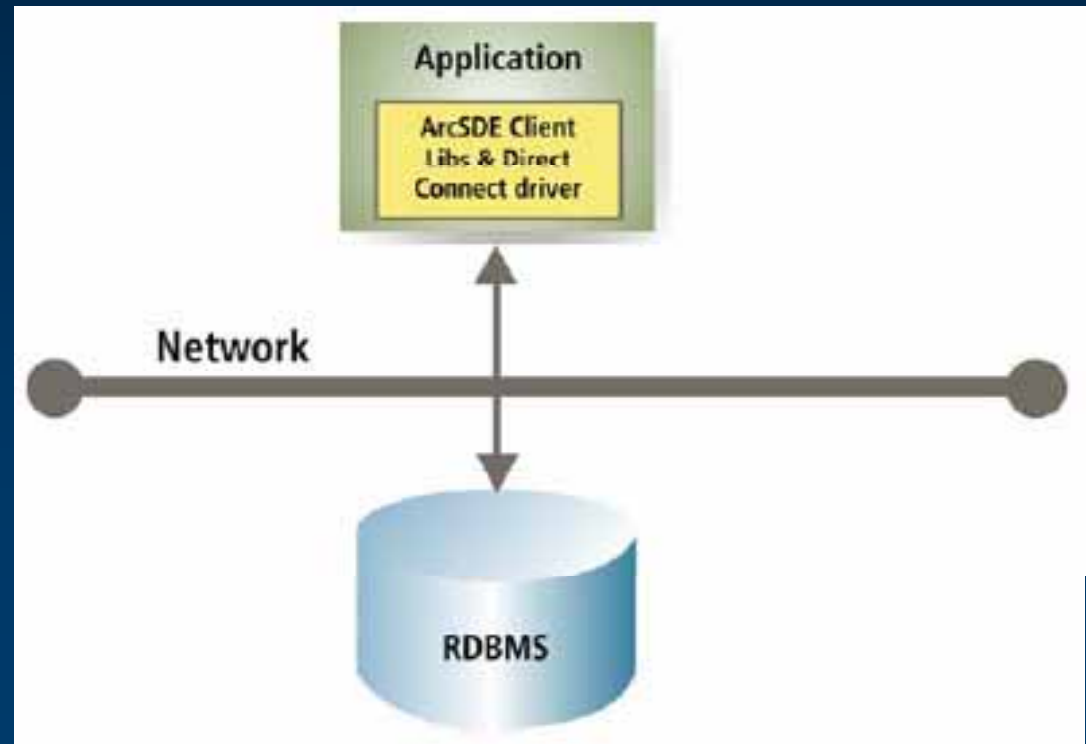
- Server machine must run ArcSDE service
 - **giomgr/gsrvr** processes
 - Can decrease client load by increasing server load



Database Connection

Direct Connect

- Connect directly to database
- No ArcSDE service required
 - No **giomgr/gsrvr** processes
 - Can decrease server load by increasing client load



Making a connection

- Connecting using a property set

```
IWorkspaceFactory sdeWkspFact = new SdeWorkspaceFactoryClass();
IPropertySet propset = new PropertySetClass();

propset.SetProperty("SERVER", "crimsontide");
propset.SetProperty("INSTANCE", "5151");
propset.SetProperty("USER", "brent");
propset.SetProperty("PASSWORD", "brent");
propset.SetProperty("DATABASE", "null");
propset.SetProperty("VERSION", "SDE.DEFAULT");
propset.SetProperty("AUTHENTICATION_MODE", "DBMS");

IWorkspace workspace = sdeWkspFact.Open(propset, 0);
```

- Connecting using a connection file

```
string nameOfFile = "D:\\data\\redarrow.sde";
IWorkspace workspace = workspaceFactory.OpenFromFile(nameOfFile, 0);
```

Personal ArcSDE geodatabases

- Rich API
 - Analyze statistics
 - Create a geodatabase

```
IDataServerManager DSManager = new DataServerManagerClass();  
DSManager.ServerName = "crimsontide\\sqlexpress";  
DSManager.Connect();  
IDataServerManagerAdmin DSMAdmin = (IDataServerManagerAdmin)DSManager;  
  
DSMAdmin.CreateGeodatabase("MyGDB", "C:\\MyGDB.mdf", 100, "C:\\MyGDB.ldf", 100);
```

- Detatch/Attach geodatabases

```
DSMAdmin.DetatchGeodatabase("MyGDB");  
  
DSMAdmin.AttachGeodatabase("MyGDB", "C:\\MyGDB.mdf", "C:\\MyGDB.ldf");
```

Once Connected...

Information about the connected workspace

- **IDatabaseConnectionInfo2**
 - **GeodatabaseServerClass**
 - **esriServerClassEnterprise**
 - **esriServerClassWorkgroup**
 - **esriServerClassPersonal**
 - **ConnectionDBMS**
 - **esriDBMS_DB2**
 - **esriDBMS_Informix**
 - **esriDBMS_Oracle**
 - **esriDBMS_SQLServer**
 - **ConnectionCurrentDateTime**
 - **ConnectedUser**
 - **ConnectedDatabase**



Working With Schemas

Schema Cache

What is the schema cache?

- **A cached snapshot of the GDB schema**
 - Used by ArcGIS (opening Map Documents, Reconcile)
 - Requires a static data model
 - GDB schema changes will not be reflected in the cache
- **Can improve performance when opening datasets**
 - Reduces database round trips by using the cached information
- **APIs to access the schema cache**
 - `IWorkspaceFactorySchemaCache`
 - `EnableSchemaCache`
 - `EnableSchemaCaching`
 - `RefreshSchemaCache`
 - `DisableSchemaCache`

Schema Cache

When to use a schema cache?

- **Beneficial when working with large static data models**
 - Tables, fields, domains, sub-types and relationships are well defined and will not change
- **If the application opens and uses many classes**
 - These should be opened at the start of the application and references maintained throughout the lifetime of the application

Schema Cache

How to leverage in enterprise applications

- **Enable schema cache before tables are opened**
 - Calls to `OpenFeatureClass`, `OpenTable`, `IName.Open` will be optimized
 - Can enable schema caching at the factory level
- **Cache needs to be “fresh”**
 - If in dynamic environments schema changes will not be visible until cache is refreshed.

```
//Spatial Cache Refresh if stale
if( sCache.IsSchemaCacheStale(workspace) )
{
    sCache.RefreshSchemaCache(workspace);
}
```

- **your responsibility to disable the cache**
 - Must be disabled before releasing the workspace object that is cached

Using Schema Caching

- 1 Cast to `IWorkspaceFactorySchemaCache` from the workspace factory and open the workspace

```
IWorkspaceFactorySchemaCache sCache =  
(IWorkspaceFactorySchemaCache)workspaceFactory;  
  
IWorkspace workspace = workspaceFactory.Open(propset, 0);  
  
sCache.EnableSchemaCache(workspace);  
  
IFeatureWorkspace featureWorkspace = (IFeatureWorkspace)workspace;  
IFeatureClass parcel = featureWorkspace.OpenFeatureClass("ParcelClass");  
//...open all feature classes used in application  
  
sCache.DisableSchemaCache(workspace);
```

Using Schema Cache

② Enable the schema cache on the workspace

Alternatively `EnableSchemaCaching` will enable caching on all workspace passed out by the factory

```
IWorkspaceFactorySchemaCache sCache =  
(IWorkspaceFactorySchemaCache)workspaceFactory;  
  
IWorkspace workspace = workspaceFactory.Open(propset, 0);  
  
sCache.EnableSchemaCache(workspace);  
  
IFeatureWorkspace featureWorkspace = (IFeatureWorkspace)workspace;  
IFeatureClass parcel = featureWorkspace.OpenFeatureClass("ParcelClass");  
//...open all feature classes used in application  
  
sCache.DisableSchemaCache(workspace);
```

Using Schema Cache

3 Open all feature classes used in application

Largest benefit for stand alone feature classes and tables

```
IWorkspaceFactorySchemaCache sCache =  
(IWorkspaceFactorySchemaCache)workspaceFactory;  
  
IWorkspace workspace = workspaceFactory.Open(propset, 0);  
  
sCache.EnableSchemaCache(workspace);  
  
IFeatureWorkspace featureWorkspace = (IFeatureWorkspace)workspace;  
IFeatureClass parcel = featureWorkspace.OpenFeatureClass("ParcelClass");  
//...open all feature classes used in application  
  
sCache.DisableSchemaCache(workspace);
```

Using Schema Cache

- 4 Disable the schema cache after the classes have been opened

```
IWorkspaceFactorySchemaCache sCache =  
(IWorkspaceFactorySchemaCache)workspaceFactory;  
  
IWorkspace workspace = workspaceFactory.Open(propset, 0);  
  
sCache.EnableSchemaCache(workspace);  
  
IFeatureWorkspace featureWorkspace = (IFeatureWorkspace)workspace;  
IFeatureClass parcel = featureWorkspace.OpenFeatureClass("ParcelClass");  
//...open the all feature classes used in application  
  
sCache.DisableSchemaCache(workspace);
```

Using Schema Cache

- Gotchas

- If the GDB schema changes cache will have to be refreshed
 - Geodatabase schema changes will not be visible until cache is refreshed

```
IWorkspaceFactorySchemaCache sCache =  
(IWorkspaceFactorySchemaCache)workspaceFactory;  
  
IWorkspace workspace = workspaceFactory.Open(propset, 0);  
  
sCache.EnableSchemaCache(workspace);  
  
IFeatureWorkspace featureWorkspace = (IFeatureWorkspace)workspace;  
IFeatureClass parcel = featureWorkspace.OpenFeatureClass("ParcelClass");  
//...open all feature classes used in application  
  
sCache.DisableSchemaCache(workspace);
```



Cursors

Cursors

What are they?

- A geodatabase object used for the iteration of records returned from a query
- **3 Class Cursors**
 - Search (general query cursor)
 - Update (positioned update cursor)
 - Insert (bulk inserts)
- **1 QueryDef Cursor**
 - Defined query (e.g. IQueryDef.Evaluate)
- **What's the difference?**
 - Rows created by Class cursors are bound to the class which created the cursor, rows created by a QueryDef cursor are not bound to a class

Cursors: Recycling Property

What is Recycling?

- A **recycling cursor** is a cursor that hands out the same row object on each call to `NextRow`
 - Internal data structures and objects will be reused

```
//example of a recycling cursor  
IFeatureCursor searchCursor = featureClass.Search(queryFilter, true)
```

- A **non-recycling cursor** is a cursor that hands out a new row object on each call to `NextRow`
 - New internal data structures and objects will be created for each row

```
//example of a non-recycling cursor  
IFeatureCursor searchCursor = featureClass.Search(queryFilter, false)
```

Cursors: Recycling

When to use a recycling cursor

- Use recycling cursors when references to the current row and its values **do not need** to be kept beyond the next call to `NextRow/NextFeature`
- do not pass the references around as some other method may decide to hold it

Cursors: Non-Recycling

When to use a non-recycling cursor

- Use non-recycling cursors when references to the current row and its values **are needed** beyond the next call to `NextRow/NextFeature`
- Commonly used to cache sets of rows (long lived references)
- Some Geodatabase APIs require sets of rows – should be retrieved as non-recycled rows



Querying and Joining Data

Querying and Joining Data

- **Spatial Caching**
 - Optimizing spatial queries
- **Query Filters / Spatial Filters**
 - Class based queries
- **Persisted Views**
 - ArcSDE Views (Multi Versioned Views)
- **On-The-Fly Joins**
 - Query Defs
 - Cursors
 - QueryTables and RelQueryTables
 - Tables

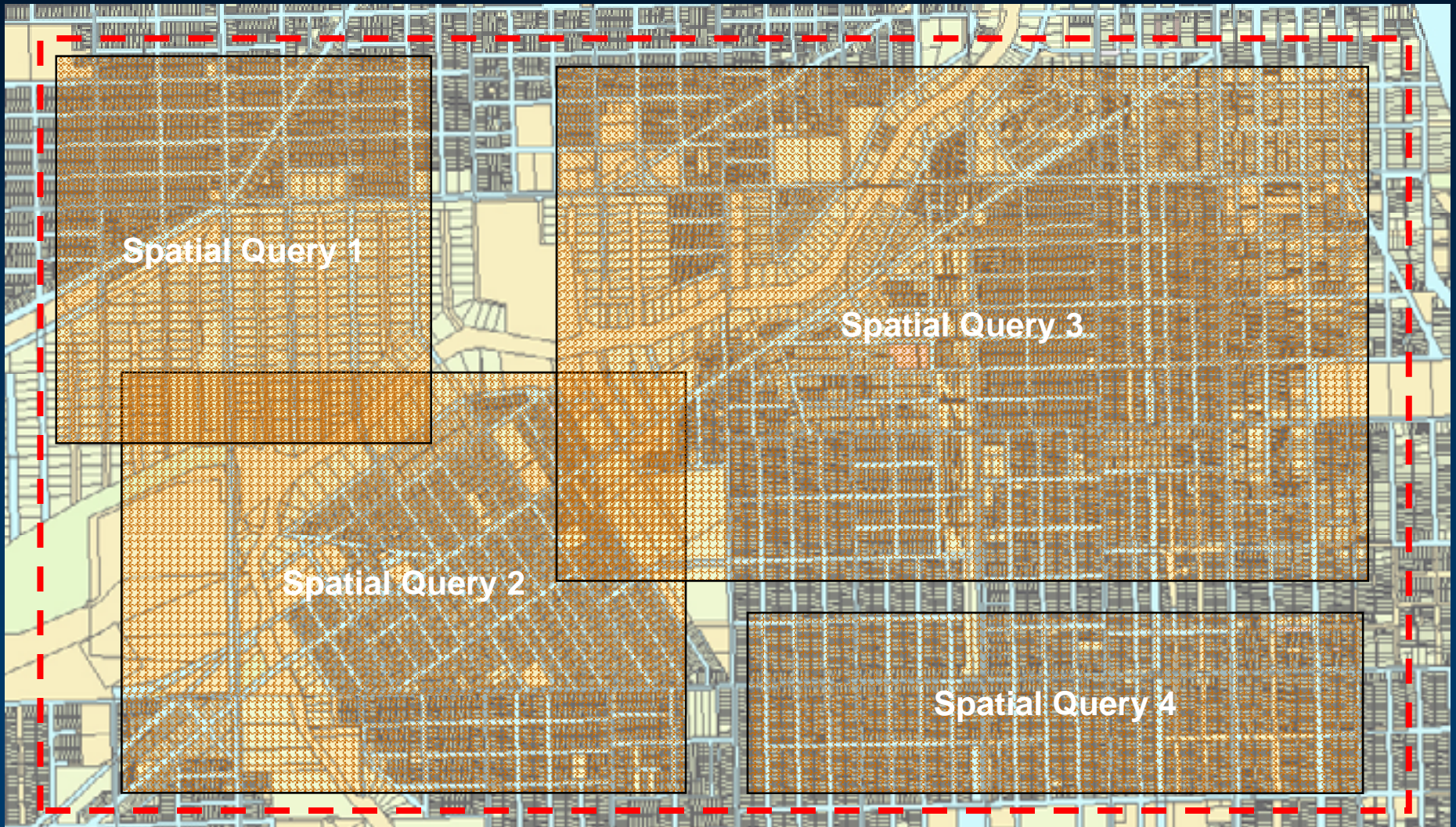
Spatial Caching

What is it?

- Client side caching of feature values over a given spatial extent (MapCache in ArcMap)
- Can speed up queries
 - Reduces roundtrips to the database
- `ISpatialCacheManager2`
 - `FillCache \ EmptyCache`
 - `CacheExtent`
- When to use?
 - If making many spatial queries within a common extent

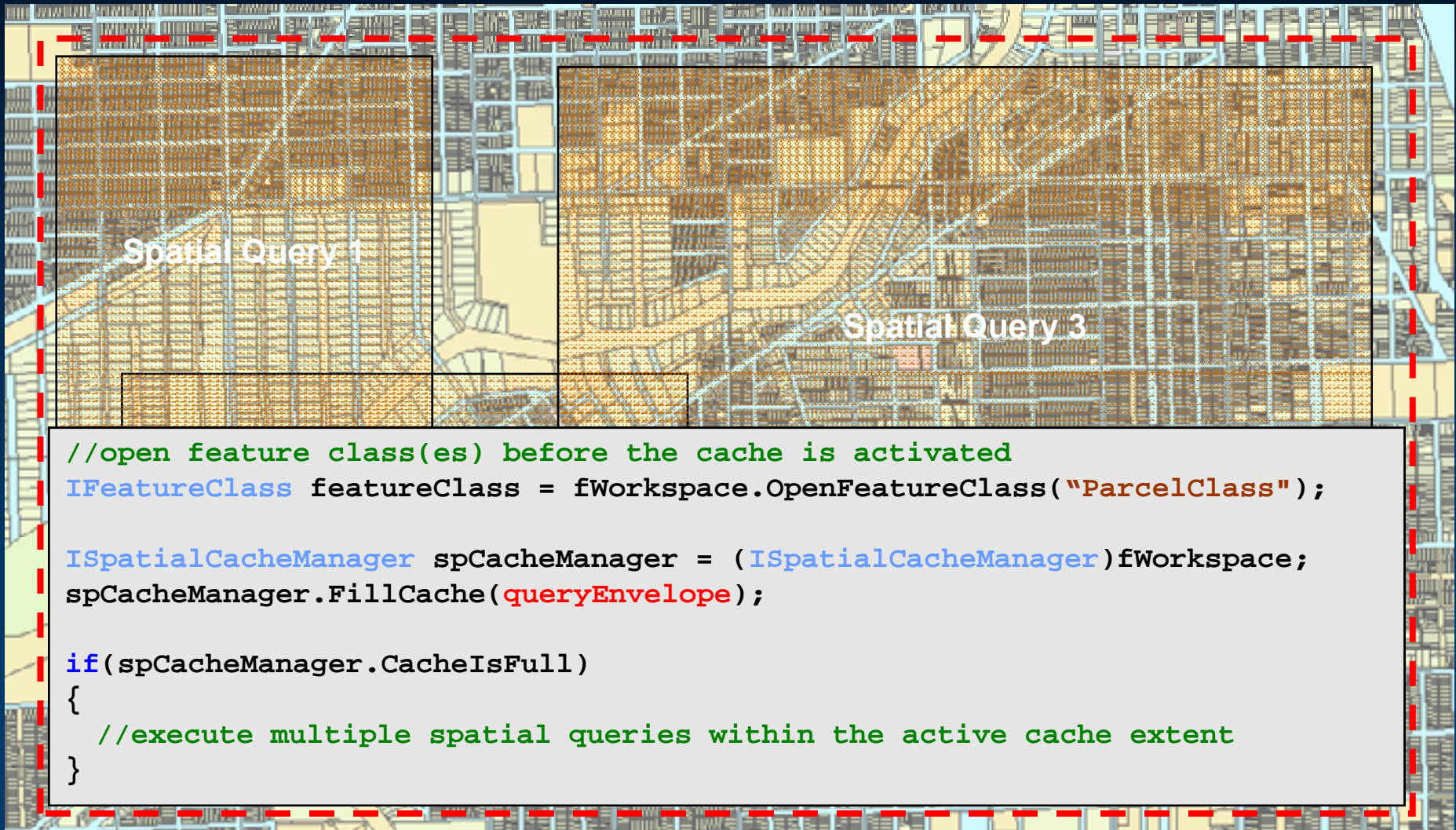
Spatial Caching

How to leverage spatial caching in enterprise applications



Spatial Caching

How to leverage spatial caching in enterprise applications

An aerial photograph of a city street grid is overlaid with a semi-transparent grid of small orange squares. Three rectangular areas are highlighted with solid black borders and labeled: 'Spatial Query 1' on the left, 'Spatial Query 2' in the middle, and 'Spatial Query 3' on the right. A larger dashed red border encompasses the entire grid area.

Spatial Query 1

Spatial Query 2

Spatial Query 3

```
//open feature class(es) before the cache is activated
IFeatureClass featureClass = fWorkspace.OpenFeatureClass("ParcelClass");

ISpatialCacheManager spCacheManager = (ISpatialCacheManager)fWorkspace;
spCacheManager.FillCache(queryEnvelope);

if(spCacheManager.CacheIsFull)
{
    //execute multiple spatial queries within the active cache extent
}
```

IQueryFilter

- Used when querying a single class
- `IQueryFilterDefinition::PostFixClause`
 - Supports aggregate functions (group by, order by)

```
IQueryFilter queryFilter = new QueryFilterClass();
queryFilter.SubFields = "OBJECTID,FULLNAME,ParcelID";
queryFilter.WhereClause = "FULLNAME like 'D%'";

IQueryFilterDefinition queryFilterDef =
(IQueryFilterDefinition)queryFilter;
queryFilterDef.PostfixClause = "ORDER BY FULLNAME";

IFeatureCursor featureCursor = featureClass.Search(queryFilter, true);
```

ISpatialFilter

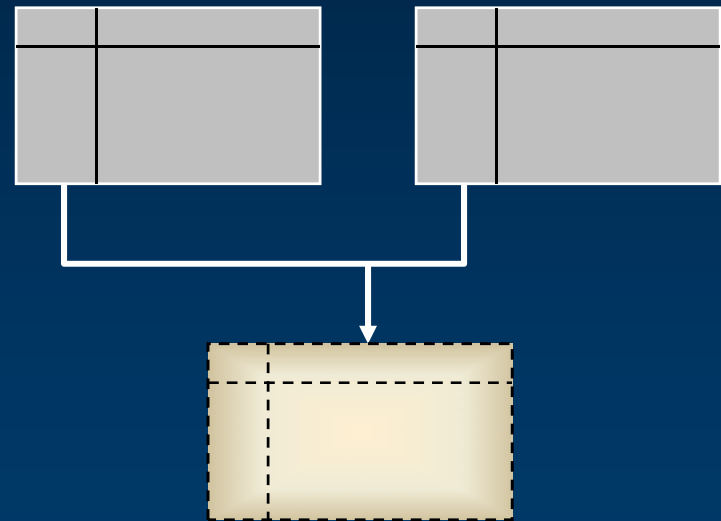
- Used to query spatial aspects of a feature class
 - Inherits from IQueryFilter

```
ISpatialFilter spatialFilter = new SpatialFilterClass();  
  
spatialFilter.SubFields = "OBJECTID,FULLNAME,ParcelID,SHAPE";  
spatialFilter.Geometry = envelope;  
spatialFilter.SpatialRel = within;  
spatialFilter.WhereClause = "FULLNAME like 'D%'";  
  
IFeatureCursor featureCursor = featureClass.Search(spatialFilter, true);
```

- Will be satisfied by spatial cache if within cache extent

Views, Table Joins and Complex Queries

- **Persisted**
 - ArcSDE Views (MultiVersion Views)
- **On-The-Fly**
 - Queries
 - QueryDef
 - Tables
 - QueryTables
 - RelQueryTables



Persisted Views

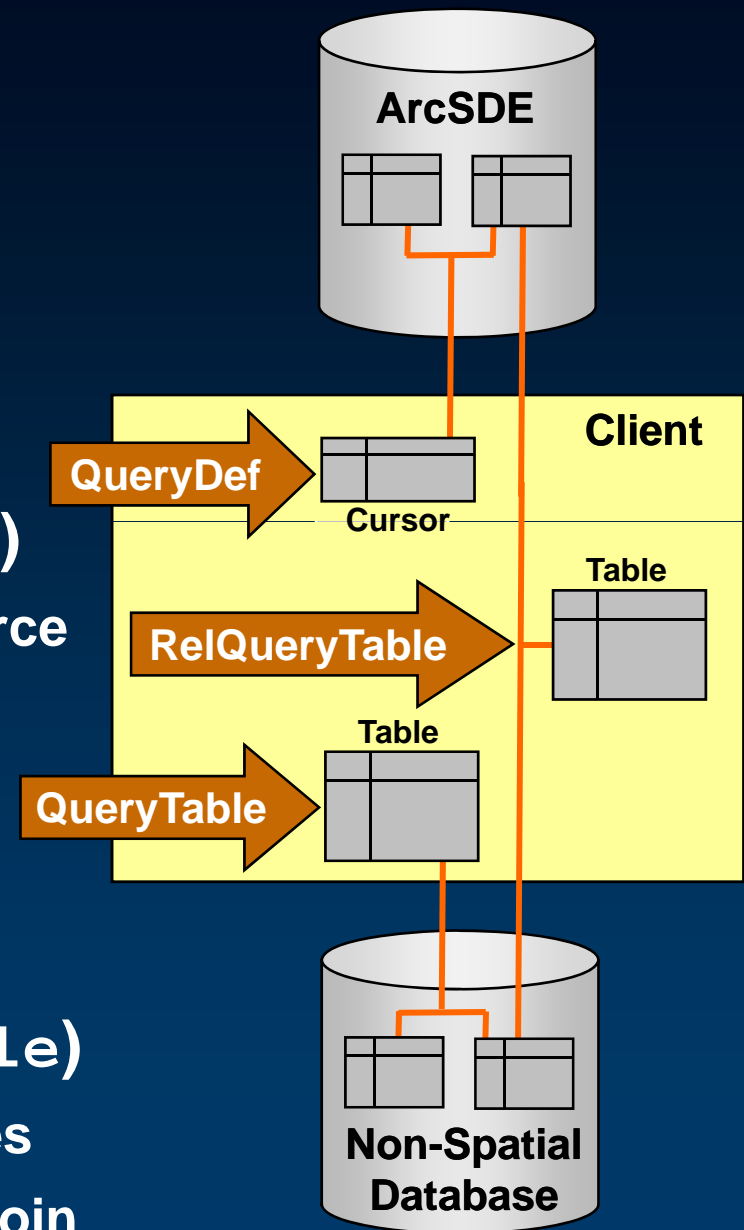
- Open views as tables (read only)

```
//open view through API as (read only) table  
ITable myJoinedTable = fWorkspace.OpenTable("databaseView");
```

- Must satisfy geodatabase rules for a valid table or feature class
 - Only one spatial column
 - Supported data types in returned fields

On-The-Fly Joins

- **QueryDefs (IQueryDef)**
 - Tables must be in same datasource
 - Represented as a cursor
 - Can't be added to ArcMap
- **QueryTables (ITableQueryName)**
 - Tables must be within same datasource
 - Matches all candidates
 - Uses QueryDef object
 - Can be used with non-spatial tables
- **RelQueryTables (IRelQueryTable)**
 - Tables can be in different datasources
 - Matches only first candidate on 1:M join
 - Uses in-memory or geodatabase relationship classes



Example: QueryDef Joins (IQueryDef)

- Simple 1:M table join

```
//Create the query definition
IQueryDef queryDef = featureWorkspace.CreateQueryDef();

//Provide a list of tables to join
queryDef.Tables = "PoleFeature, TransformerFeature";

//Retrieve the fields from all tables
queryDef.SubFields = "bob.PoleFeature.TAG, bob.PoleFeature.SHAPE,
bob.TransformerFeature.Tag_Val";

//Set up the join based on the owner_name attribute
queryDef.WhereClause = "PoleFeature.TAG = TransformerFeature.Tag_Val";

ICursor cursor = queryDef.Evaluate();
IRow row = cursor.NextRow();
```

Example: QueryDef Joins (IQueryDef)

- Complex 1:M table Join

```
IQueryDef queryDef = featureWorkspace.CreateQueryDef();
queryDef.Tables = "ElectricStation, CircuitSource, CableJunction";

queryDef.SubFields = "ElectricStation.StationName,
    ElectricStation.StationAbbreviation,
    CableJunction.ObjectID";
queryDef.WhereClause = "((ElectricStation.CircuitID LIKE 'A-235%' " +
    "AND SUBSTR(ElectricStation.CircuitID,5+1,1) not between '0' and '9'))" +
    "OR (ElectricStation.CircuitID = 'A-235'))" +
    "AND ElectricStation.StationTypeCode = 'GEN'" +
    "AND CircuitSource.ObjectID = ElectricStation.CircuitSourceObjectID" +
    "AND ElectricStation.ObjectID = CircuitSource.ElectricStationObjectID" +
    "AND CableJunction.DeviceObjectID = ElectricStation.ObjectID" +
    "AND CableJunction.DeviceType = 'ENG'" +
    "OR INSTR(UPPER(CircuitSource.PhaseDesignation),'123') > 0)";

ICursor cursor = queryDef.Evaluate();
IRow row = cursor.NextRow();
```

Example: QueryTables (ITableQueryName)

- Steps to create a join via ITableQueryName
 - ① Create a new TableQueryName object
 - ② Set the QueryDef and PrimaryKey property
 - ③ Cast to IDatasetName setting WorkspaceName and Name
 - ④ Open the name object as a table

```
// Make the new TableQueryName
IQueryName2 qn2 = (IQueryName2)new TableQueryName();
qn2.QueryDef = qdef;
qn2.PrimaryKey = "ObjectID";
qn2.CopyLocally = false;

// Set the workspace and name of the new QueryTable
IDatasetName pDSName = (IDatasetName)qn2;
pDSName.WorkspaceName = WSName;
pDSName.Name = TableName;

// Open and return the table
IName name = (IName)qn2;
ITable table = (ITable)name.Open();
```

Example: RelQueryTables (IRelQueryTable)

- Steps to create a join via IRelQueryTable
 - ① Create a new memory RelationshipClass
Passing in the ObjectClasses that need to be joined
An existing relationship class can be used
 - ② Call open on RelQueryTableFactory casting to ITable
Passing in the relationship class

```
// build a memoryrelationshipclass
IMemoryRelationshipClassFactory mRCfactory = new
MemoryRelationshipClassFactoryClass();

// open the memoryrelationshipclass
IRelationshipClass memRC = mRCfactory.Open("memrc", targetObjectClass,
fromField, joinObjectClass, toField, "forward", "backward",
esriRelCardinality.esriRelCardinalityOneToOne);

// Open the relquerytable as a table
IRelQueryTableFactory rqtfactory = new RelQueryTableFactoryClass();

ITable rqTable = (ITable)rqtfactory.Open(memRC, true, null, null, "",
false, true);
```



Editing and Transactions

Geodatabase & DBMS Transactions

Transaction Models

- **Non-versioned**
 - Utilizes database transactions
 - Transaction scope controlled through `startEditing` and `stopEditing` calls
- **Versioned**
 - Provides read consistency and isolation
 - Geodatabase transaction is defined by the scope of the edit session
 - Changes are only viewable by the editor until the edit session is explicitly saved
 - Provides undo and redo
 - Transaction scope controlled through `startOperation` and `stopOperation` calls

Geodatabase & DBMS Transactions

Outside of an Edit Session (i.e. Updating)

- **ITransactions interface** – `StartTransaction`, `CommitTransaction`, `AbortTransaction`, `InTransaction`
- **ITransactionsOptions interface** – `AutoCommitInterval`
 - Number of modifications before a commit is automatically executed
 - Default is 1000 (server config)
 - 0 == no automatic commits (developers should set to 0 if server is configured with appropriate rollback resources)
- **DBMS rules apply** – **DDL may commit DML**

Geodatabase & DBMS Transactions

Within a Non-Versioned Edit Session (i.e. Editing)

- The logical transaction

- `StartEditing == StartTransaction`
- `StopEditing(true) == CommitTransaction`
- `StopEditing(false) == AbortTransaction`

- `StartMultiUserEditing` turns off auto commit (set to 0)

```
muWEdit.StartMultiuserEditing(esriMultiuserEditMode.esriMESMNonVersioned);
```

- Editing errors need to be handled around the API committing the transaction – if error need to call `StopEditing(false)`
- Don't mix use of `ITransaction` and `IMultiUserWorkspaceEdit`

Geodatabase & DBMS Transactions

Within a Versioned Edit Session (i.e. Editing)

- The logical transaction

- `StartEditOperation == StartTransaction`
- `StopEditOperation == CommitTransaction`
- `AbortEditOperation == AbortTransaction`

- `StartMultiUserEditing` turns off auto commit (set to 0)

```
muWEdit.StartMultiuserEditing(esriMultiuserEditMode.esriMESMVersioned);
```

- Editing errors need to be handled around the API committing the transaction – if error need to call `AbortEditOperation`

Geodatabase & DBMS Transactions

Developer Considerations

- **Geodatabase “DDL” and other objects which invoke it may cause a commit**
 - CreateFeatureClass
 - Geoprocessing Tools
 - Etc...
- **Transactions are per “connection” not per Version**

Geodatabase & DBMS Transactions

Transaction Scope

- **Non-Versioned Edit Session**

- Make many calls to `stopEditing(true)`
- Edit Operations are no ops

```
muWEdit.StartMultiuserEditing(esriMESMNonVersioned);  
    // Edit Code  
workspaceEdit.StopEditing(true);  
    //...  
muWEdit.StartMultiuserEditing(esriMESMNonVersioned);  
    // Edit Code  
workspaceEdit.StopEditing(true);
```

} Database Transaction

} Database Transaction

- **Versioned Edit Session**

- Avoid excessive calls to `stopEditing(True)`

```
muWEdit.StartMultiuserEditing(esriMESMVersioned);  
workspaceEdit.StartEditOperation();  
    // Edit Code  
workspaceEdit.StopEditOperation();  
    //...  
workspaceEdit.StartEditOperation();  
    // Edit Code  
workspaceEdit.StopEditOperation();  
workspaceEdit.StopEditing(true);
```

} Database Transaction

} Database Transaction

Example: Editing a Versioned Geodatabase

- Steps to making a version edit
 - ① Get a reference to the `version\workspace` that is to be edited
 - ② Use `IMultiUserWorkspaceEdit` to start the appropriate edit session
 - ③ Use `IWorkspaceEdit.StartOperation` to start an edit operation
 - ④ Make edit
 - ⑤ Stop operation and stop edit session saving edits

Versioned Editing

- 1 Get a reference to the `Version` that is to be edited

Open the `FeatureClass` to be edited from this version

```
IFeatureWorkspace fWorkspace =  
(IFeatureWorkspace)versionedWorkspace.FindVersion("SDE.DEFAULT");  
  
IFeatureClass parcelClass = fWorkspace.OpenFeatureClass("ParcelClass");  
  
IWorkspaceEdit workspaceEdit = (IWorkspaceEdit)featureWorkspace;  
IMultiuserWorkspaceEdit muWEdit = (IMultiuserWorkspaceEdit)featureWorkspace;  
  
if(muWEdit.SupportsMultiuserEditSessionMode(esriMESMVersioned))  
{  
muWEdit.StartMultiuserEditing(esriMESMVersioned);  
workspaceEdit.StartEditOperation();  
IFeature feature = parcelClass.GetFeature(2);  
feature.set_Value(8, "TEG");  
feature.Store();  
workspaceEdit.StopEditOperation();  
workspaceEdit.StopEditing(true);  
}
```

Versioned Editing

- 2 Use `IMultiUserWorkspaceEdit` to start the appropriate edit session

```
IFeatureWorkspace fWorkspace =  
(IFeatureWorkspace)versionedWorkspace.FindVersion("SDE.DEFAULT");  
  
IFeatureClass parcelClass = fWorkspace.OpenFeatureClass("ParcelClass");  
  
IWorkspaceEdit workspaceEdit = (IWorkspaceEdit)featureWorkspace;  
IMultiuserWorkspaceEdit muWEdit = (IMultiuserWorkspaceEdit)featureWorkspace;  
  
if(muWEdit.SupportsMultiuserEditSessionMode(esriMESMVersioned))  
{  
muWEdit.StartMultiuserEditing(esriMESMVersioned);  
workspaceEdit.StartEditOperation();  
IFeature feature = parcelClass.GetFeature(2);  
feature.set_Value(8, "TEG");  
feature.Store();  
workspaceEdit.StopEditOperation();  
workspaceEdit.StopEditing(true);  
}
```

Versioned Editing

- 3 Use `IWorkspaceEdit.StartOperation` to start an edit operation

```
IFeatureWorkspace fWorkspace =
(IFeatureWorkspace)versionedWorkspace.FindVersion("SDE.DEFAULT");

IFeatureClass parcelClass = fWorkspace.OpenFeatureClass("ParcelClass");

IWorkspaceEdit workspaceEdit = (IWorkspaceEdit)featureWorkspace;
IMultiuserWorkspaceEdit muWEdit = (IMultiuserWorkspaceEdit)featureWorkspace;

if(muWEdit.SupportsMultiuserEditSessionMode(esriMESMVersioned))
{
muWEdit.StartMultiuserEditing(esriMESMVersioned);
    workspaceEdit.StartEditOperation();
        IFeature feature = parcelClass.GetFeature(2);
        feature.set_Value(8, "TEG");
        feature.Store();
        workspaceEdit.StopEditOperation();
workspaceEdit.StopEditing(true);
}
```

Versioned Editing

4 Make edit(s) to the open feature class

```
IFeatureWorkspace fWorkspace =  
(IFeatureWorkspace)versionedWorkspace.FindVersion("SDE.DEFAULT");  
  
IFeatureClass parcelClass = fWorkspace.OpenFeatureClass("ParcelClass");  
  
IWorkspaceEdit workspaceEdit = (IWorkspaceEdit)featureWorkspace;  
IMultiuserWorkspaceEdit muWEdit = (IMultiuserWorkspaceEdit)featureWorkspace;  
  
if(muWEdit.SupportsMultiuserEditSessionMode(esriMESMVersioned))  
{  
muWEdit.StartMultiuserEditing(esriMESMVersioned);  
workspaceEdit.StartEditOperation();  
    IFeature feature = parcelClass.GetFeature(2);  
    feature.set_Value(8, "TEG");  
    feature.Store();  
workspaceEdit.StopEditOperation();  
workspaceEdit.StopEditing(true);  
}
```

Versioned Editing

5 Stop operation and stop edit session saving edits

```
IFeatureWorkspace fWorkspace =  
(IFeatureWorkspace)versionedWorkspace.FindVersion("SDE.DEFAULT");  
  
IFeatureClass parcelClass = fWorkspace.OpenFeatureClass("ParcelClass");  
  
IWorkspaceEdit workspaceEdit = (IWorkspaceEdit)featureWorkspace;  
IMultiuserWorkspaceEdit muWEdit = (IMultiuserWorkspaceEdit)featureWorkspace;  
  
if(muWEdit.SupportsMultiuserEditSessionMode(esriMESMVersioned))  
{  
muWEdit.StartMultiuserEditing(esriMESMVersioned);  
workspaceEdit.StartEditOperation();  
IFeature feature = parcelClass.GetFeature(2);  
feature.set_Value(8, "TEG");  
feature.Store();  
workspaceEdit.StopEditOperation();  
workspaceEdit.StopEditing(true);  
}
```

Versioned Editing

- Gotchas

- Feature class needs to be open in the version before edit session is started

```
IFeatureWorkspace fWorkspace =  
(IFeatureWorkspace)versionedWorkspace.FindVersion("SDE.DEFAULT");  
  
IFeatureClass parcelClass = fWorkspace.OpenFeatureClass("ParcelClass");  
  
IWorkspaceEdit workspaceEdit = (IWorkspaceEdit)featureWorkspace;  
IMultiuserWorkspaceEdit muWEdit = (IMultiuserWorkspaceEdit)featureWorkspace;  
  
if(muWEdit.SupportsMultiuserEditSessionMode(esriMESMVersioned))  
{  
muWEdit.StartMultiuserEditing(esriMESMVersioned);  
workspaceEdit.StartEditOperation();  
    IFeature feature = parcelClass.GetFeature(2);  
    feature.set_Value(8, "TEG");  
    feature.Store();  
workspaceEdit.StopEditOperation();  
workspaceEdit.StopEditing(true);  
}
```



Archiving

Archiving with the Geodatabase

- **The ability to archive edits made to versioned feature classes**
- **The archives will support historical queries**
 - Display the data at a particular time or over a time span
- **Historical Version another type of version**
 - Classes can be opened from a historical version
- **Queries go against the archive table**
 - May be faster than querying Default

Archiving API

- Querying a class open in a historical version

```
IHistoricalWorkspace hWorkspace = (IHistoricalWorkspace)pWksp;  
IHistoricalVersion hVersion = hWorkspace.FindHistoricalVersionByTimeStamp(date);  
  
IFeatureWorkspace featureWorkspace = (IFeatureWorkspace)hVersion;  
IFeatureClass parcel = featureWorkspace.OpenFeatureClass("ParcelClass");
```

- Querying the archive class directly

```
IArchivableClass parcel =  
(IArchivableClass)featureWorkspace.OpenFeatureClass("ParcelClass");  
ITable parcelArchive = parcel.Archive;  
  
IQueryFilter queryFilter = new QueryFilterClass();  
queryFilter.SubFields = "APN";  
queryFilter.WhereClause = "APN = 12336 AND GDB_FROM_DATE > '2007-03-10  
20:01:48.000'";
```



Difference Cursors

Difference Cursors

What are they?

- A cursor of object differences between versions
- Common APIs which create difference cursors
 - `IVersionTable.Differences`
- Returns a `IDifferenceCursor` Object
- Can be used for Historical or Transactional versions
- Difference type must be specified

Difference Cursors

Transactional Versions

- Returns differences between any two versioned tables
 - IRow object not returned for DeleteNoChange difference case
 - To get deleted row use IVersion2.GetCommonAncestor

```
// get delete no change differences between the two versions
IDifferenceCursor differenceCursor = childTable.Differences(parentTable,
    esriDifferenceTypeDeleteNoChange,
    null);

differenceCursor.Next(out oid, out differenceRow);
featureWorkspace =
    (IFeatureWorkspace)childVersion.GetCommonAncestor(parentVersion);
caFeatureClass = featureWorkspace.OpenFeatureClass("parcelClass");

while (oid != -1)
{
    differenceRow = caFeatureClass.GetFeature(oid);
    Console.WriteLine("The difference row has an OID of {0}", oid);

    differenceCursor.Next(out oid, out differenceRow);
}
```

Difference Cursors

Historical Versions

- Does not support all differences
 - Supported for Insert, UpdateNoChange, DeleteNoChange
 - IRow object not returned for DeleteNoChange difference case

```
// get insert differences between the two historical versions
IDifferenceCursor differenceCursor = hvYearEnd2006.Differences(hvQ12007,
                                                             esriDifferenceTypeInsert,
                                                             null);

differenceCursor.Next(out oid, out differenceRow);

while (oid != -1)
{
    Console.WriteLine("The difference row has an OID of {0}", oid);
    differenceCursor.Next(out oid, out differenceRow);
}
```



Events

Version Events

- `IVersionEvents` / `IVersionEvents2`
 - Used for executing specific code blocks when events fire
 - Reconcile, Post, Version Redefine, Archive Update events

```
//event handler method
private void versionEvent_OnReconcile(string targetVersionName, bool
    HasConflicts)
{
    //filter false conflicts from conflict classes
}

//event handler method
private void versionEvent_OnArchiveUpdated(string targetVersionName, bool
    HasConflicts)
{
    //modify user maintained date fields in the archive
}
```

Code Instrumentation and Tracing

Code instrumentation

- **Why do you need it?**
 - Time certain operations in your application
 - Check how efficient the application works with the enterprise (database), e.g. number of calls and the performance of individual operations
- **Symptoms to look out for**
 - Repeated calls to the database
 - Slow running operations

Code instrumentation and tracing for performance

- **Built-in and already available**
 - ArcSDE client intercept and trace logs
 - ArcIMS logs
 - DBMS trace files
- **Custom applications**
 - Add code to time critical operations
 - Timers around an edit operation(s)
 - Data fetching
 - Enable dbms tracing

Built in ArcSDE instrumentation

- **SDETRACE and SDEINTERCEPT**
- **Both instrument client calls to the ArcSDE server**
- **Useful to profile how many and what type of calls are made**

```
//enable SDEINTERCEPT
```

```
C:\TEMP>set SDEINTERCEPTLOC=c:\temp\client
```

```
C:\TEMP>\arcgis\bin\arcmap
```

```
// "tail" the commands being executed
```

```
C:\TEMP>tail -f client.001 | grep Command
```

```
[W] Command: ExecuteSpatialQuery
```

```
[W] Command: NextBuffer
```

```
[W] Command: NextBuffer
```

```
[W] Command: CloseStream
```

```
[W] Command: StreamSetState
```

```
[W] Command: QueryWithInfo
```

```
[W] Command: SetSpatialConstraints
```

```
... .
```

Instrument data fetch and display performance

- Map display includes fetching and rendering the data
- Instrument fetching the data by timing the Feature Cursor
- Instrument the map display by timing the refresh on the data view
- The delta time difference between the Feature cursor time and the ActiveView refresh is the local rendering time
- Use to determine where the bottleneck is when having display performance on Feature Layers

Code example: Feature Cursor Performance

- Create a search cursor
- Wrap a timer around the fetch operation

```
// Create a spatial filter (Envelope)
ISpatialFilter spFil = PrepareQueryFilter(env, fLyr, null);

//Create the Fetch Cursor
IFeatureClass fCls = fLyr.FeatureClass;
start = DateTime.Now;
rCur = (ICursor)fCls.Search(spFil, true);

// Fetch the rows
while ((r = rCur.NextRow()) != null)
    featCnt++;

TimeSpan result = DateTime.Now.Subtract(start);

// Release the cursor object
Marshal.ReleaseComObject(rCur);
rCur = null;;
```

Code example: Map Display Performance

- Trigger a map re-display from your code
- Put a timer around the `ActiveView.Refresh()` method.

```
//Refresh the Map Display
start = DateTime.Now;
m_ActiveView = (IActiveView) (IMap)m_currentMap;
m_ActiveView.Refresh();

// Stop the timer and show results
TimeSpan result = DateTime.Now.Subtract(start);
```

Coding the application to do DBMS traces

- Putting code that enables operations to be traced in the database
- Allows you to figure out how much time is being spent inside the DBMS and how individual SQL statements perform, e.g. Definition Queries

Code example: enable DBMS trace

Use the `IWorkspace.ExecuteSQL` to alter the database session to begin tracing

```
//Find the dbms flavor you are connected to construct a SQL Trace Statement
String sSQLStm;

if (pDBMSInfo.ConnectionDBMS == esriConnectionDBMS.esriDBMS_Oracle)
    sSQLStm = "alter session set events '10046 trace name context forever,
    level 12'";
else if (pDBMSInfo.ConnectionDBMS == esriConnectionDBMS.esriDBMS_SQLServer)
    sSQLStm = "master..sp_trace_setstatus 1";
else if (pDBMSInfo.ConnectionDBMS == esriConnectionDBMS.esriDBMS_Informix)
    sSQLStm = "set explain on";

//Execute the SQL statement on the Workspace

pWksp.ExecuteSQL(sSQLStm);
```

DBMS trace output example: Oracle

- The output contains SQL execution paths, row counts and wait events

```
OVERALL TOTALS FOR ALL NON-RECURSIVE STATEMENTS
```

call	count	cpu	elapsed	disk	query	current	rows
Parse	6	0.00	0.00	0	0	0	0
Execute	54	0.14	0.15	0	207	649	113
Fetch	1025	3.34	3.54	50	677710	0	15054
total	1085	3.48	3.70	50	677917	649	15167

Misses in library cache during parse: 4
Misses in library cache during execute: 2

Elapsed times include waiting on following events:

Event waited on	Times Waited	Max. Wait	Total Waited
SQL*Net message to client	1052	0.00	0.00
SQL*Net message from client	1052	29.07	64.27
SQL*Net more data to client	1579	0.00	0.02
direct path read	44	0.05	0.15
log file sync	12	0.00	0.00

Other Sessions to Attend

All sessions are in the Catalina/Madera rooms

- **Using the SQL API**
 - Wednesday, 21st , 10:30-11:45 am
- **Turbocharged Geodatabase Programming**
 - Wednesday, 21st , 1-2:15 pm
- **Distributed Geodatabase for Developers**
 - Wednesday, 21st, 4:30-5:45 pm
- **Raster Data Management in ArcGIS 9.2**
 - Thursday, 22nd , 8:30-9:45 am

Further Questions

- **TECH-TALK**

- **What:** Opportunity to ask questions and discuss concerns with presenters and other GDB team members
- **Where:** One of the five Tech Rooms
- **When:** During the next 30 minutes

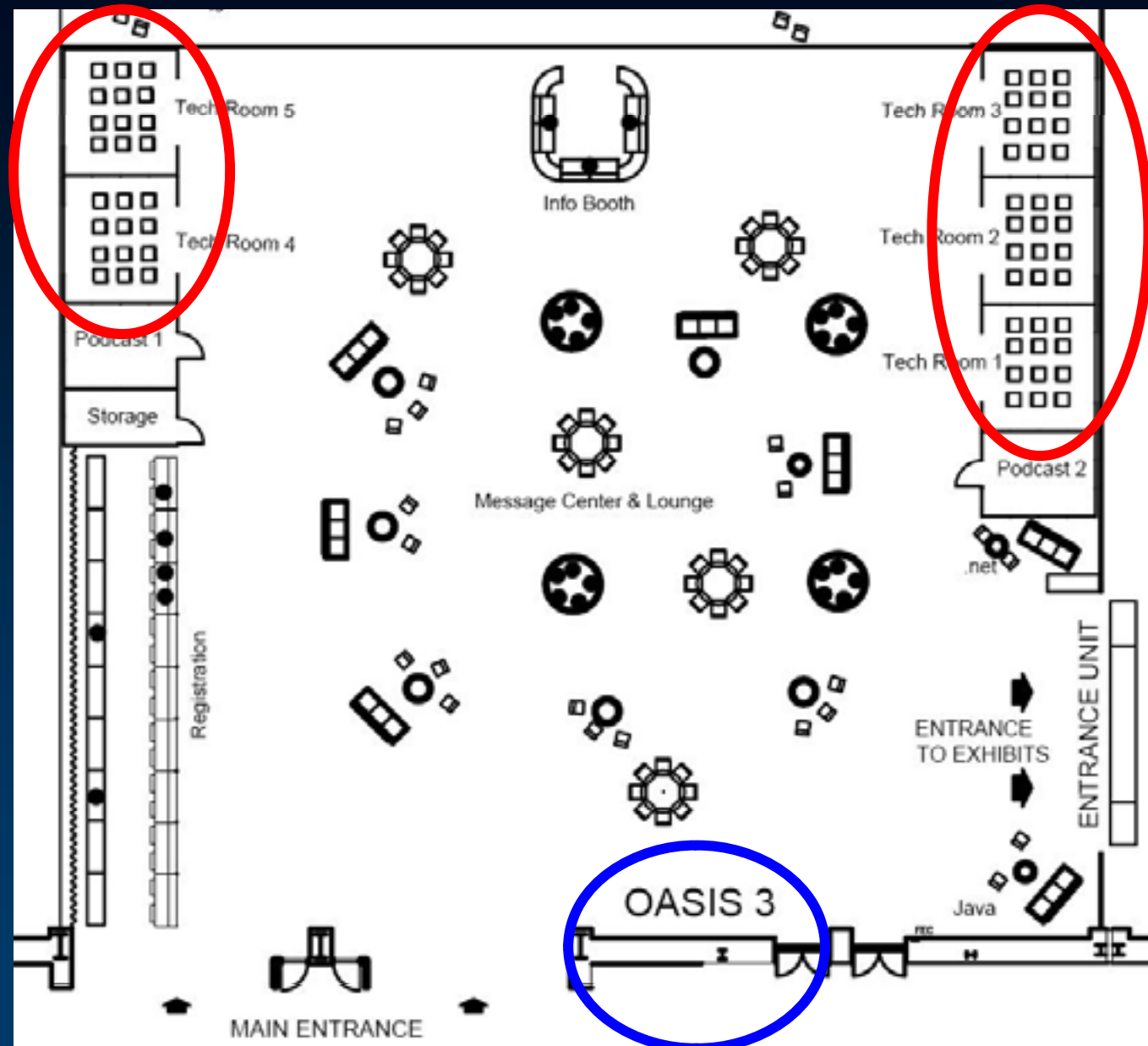
- **Meet the Teams**

- **When:** Wednesday, 11:00am, Message Center and Lounge, Oasis 3

- **ESRI Developers Network (EDN) website**

- <http://edn.esri.com>

Tech Talk Map



Showcase Organization

