Implementing Enterprise Applications with the Geodatabase

Brent Pierce
Forrest Jones
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 and Server Island Area for questions
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
  – Reconcile and Post
• Difference Cursors
• Archiving
• Version 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

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

```csharp
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

```csharp
string nameOfFile = "D:\data\redarrow.sde";
IWorkspace workspace = workspaceFactory.OpenFromFile(nameOfFile, 0);
```
ArcSDE for SQL Server Express Geodatabases

• Detailed API
  – Analyze statistics
  – Create a geodatabase

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

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

– Detach/Attach geodatabases
Once Connected…
Information about the connected workspace

- **IDatabaseConnectionInfo2**
  - **GeodatabaseServerClass**
    - esriServerClassEnterprise
    - esriServerClassWorkgroup
    - esriServerClassPersonal

- **ConnectionDBMS**
  - esriDBMS_DB2
  - esriDBMS_Informix
  - esriDBMS_Oracle
  - esriDBMS_SQLServer
  - esriDBMS_PostgreSQL

- **ConnectionCurrentDateTime**
- **ConnectedUser**
- **ConnectedDatabase**

- **IFeatureClassStorage**
Working With Schemas
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.

```csharp
// 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

```csharp
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 Caching

2 Enable the schema cache on the workspace
   • Alternatively EnableSchemaCaching will enable caching on all workspace passed out by the factory

```csharp
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 Caching

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 Caching

4 Disable the schema cache after the classes have been opened

```csharp
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 Caching

• 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
Class Cursors

Types

• **Search cursors**
  – Returns rows specified by a Query or Spatial Filter

• **Update cursors**
  – Update and delete rows specified by the filter
  – Specify the ObjectID field

• **Insert cursors**
  – Used for inserting rows into a table

• **Accessed by**
  – Corresponding methods on table or feature class
Class Cursors

Types …

• Insert cursors are used to bulk insert rows
  – Faster for loading simple data than IFeature.Store
    • Bypasses events
    • IObjectClassInfo2 and IWorkspaceEditControl to override
  – Not Faster for non-simple data
    • Behavior, composite relationships, and notification
    • Need CreateRow and Store methods, so no performance gain
  – Use of Buffering is key
    • Pre-define attribute values
    • Buffers inserts on client, sends to database on Flush

• Flush – Call or not
  – Interval flushing: Check for room or handle errors
  – Careful: Insert cursors flush on destruction
    • No chance to detect errors
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

```csharp
//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

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

- Query based on one or more tables
  - Analogous to an SQL Query
  - Get a cursor back
  - Not bound to a class

- Tables must be in database
  - Do not have to be registered with the geodatabase

- Can result in a feature layer
  - Need to use TableQueryName::IQueryName2 if no ObjectIDs in input tables

- Are also used to establish joins
Example: QueryDef Cursors

- Simple Query (One Table)

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

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

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

//Set up the join based on the owner_name attribute
queryDef.WhereClause = "PoleFeature.TAG > 1000";

ICursor cursor = queryDef.Evaluate();
IRow row = cursor.NextRow();
```
Example: QueryDef Joins

- Simple 1:M table join

```csharp
//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

- Complex 1:M table Join

```csharp
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();
```
Querying and Joining Data
Querying and Joining Data

• Defining the query
  – Query Filters
  – Spatial Filters

• Optimizing queries
  – Spatial Caching

• Persisted Views
  – ArcSDE Views (Multi Versioned Views)

• On-The-Fly Joins
  – QueryTables
  – RelQueryTables
Defining the query

**IQueryFilter**

- Used when querying a single class
- `IQueryFilterDefinition::PostFixClause`
  - Supports aggregate functions (i.e. order by)

```csharp
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);
```
Defining the query

- Used to query spatial aspects of a feature class
  - Inherits from IQueryFilter
- Will be satisfied by spatial cache if within cache extent

```csharp
ISpatialFilter spatialFilter = new SpatialFilterClass();
spatialFilter.SubFields = "OBJECTID,FULLNAME,ParcelID,SHAPE";
spatialFilter.Geometry = searchShape;
spatialFilter.SpatialRel = within;
spatialFilter.WhereClause = "FULLNAME like 'D%'";

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

```csharp
//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
}
```
Views, Table Joins and Complex Queries

• Persisted
  – ArcSDE Views (MultiVersion Views)

• On-The-Fly
  – QueryTables
    • Same Database
      – Spatial and Non-Spatial
  – RelQueryTables
    • Cross Database
      – Spatial and Non-Spatial
Persisted Views

• Open views as tables (read only)

```csharp
//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

- **QueryTables (`ITableQueryName`)**
  - Tables must be within same datasource
  - Matches all candidates
  - Uses QueryDef object
  - Can be used with non-spatial tables
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

```csharp
// 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();
```
On-The-Fly Joins

- **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 relationship classes
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

```csharp
// 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 `StartEditOperation` and `StopEditOperation` 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)

```csharp
muWEdit.StartMultiuserEditing(esriMultiuserEditSessionMode.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
The logical transaction

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

StartMultiUserEditing turns off auto commit (set to 0)

```csharp
muWEdition.StartMultiuserEditing(esriMultiuserEditSessionMode.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

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

• Versioned Edit Session
  – Avoid excessive calls to StopEditing(True)

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

```csharp
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

- Use IMultiUserWorkspaceEdit to start the appropriate edit session

```csharp
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

```csharp
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

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

```csharp
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

```csharp
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

Reconcile and Post

• Ability to merge changes from one version to another

```csharp
// To reconcile and post the changes, we need to start a
// new edit session.
muWorkspaceEdit.StartMultiuserEditing(esriMultiuserEditSessionMode.esriMESMVersioned);

// Cast the workspace to the IVersionEdit4 interface and reconcile the changes
// with the QA version.
IVersionEdit4 versionEdit4 = (IVersionEdit4)workspace;
Boolean conflictsDetected = versionEdit4.Reconcile4("QA", true, true, false, true);

// CanPost indicates that posting can occur, post the changes to the QA version.
if (versionEdit4.CanPost())
{
    versionEdit4.Post("QA");
    workspaceEdit.StopEditing(true);
}
else
{
    // Stop the edit session, without saving any edits.
    workspaceEdit.StopEditing(false);
}
```
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

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

• Querying the archive class directly

```csharp
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

```csharp
// 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

```csharp
// 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);
}
```
Differences Within An Edit Session

EditDataChanges

- Ability to determine what’s changed
  - Within a Edit Session or Edit Operation
  - Returns difference cursor
- Works on all types of geodatabases

```csharp
// get insert differences in the edit session

IDataChangesEx editDataChanges;
editDataChanges = workspaceEdit2.get_EditDataChanges(WithinSession);

ESRI.ArcGIS.esriSystem.IEnumBSTR enumBSTR = editDataChanges.ModifiedClasses;
string changedClass = enumBSTR.Next();

IDifferenceCursorEx differenceCursor;
differenceCursor = editDataChanges.ExtractEx(changedClass, TypeInsert);
```
Version Events

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

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

//event handler method
dprivate void versionEvent_OnArchiveUpdated(string targetVersionName, bool HasConflicts)
{
    //modify user maintained date fields in the archive
}
```
Code Instrumentation and Tracing
Code instrumentation and tracing

What is it?

• Code Instrumentation
  – Adding code to application for debug
  – Time certain operations in your application

• Tracing
  – More information about what applications are doing
  – Using existing tools to trace
  – Instrument code to produce own trace files

• Allows a look at how application is performing
Code instrumentation and tracing

• Why do you need it?
  – Check how efficient the application works with the database
    • e.g. number of calls and the performance of individual operations
  – Determine application transaction times
  – Identify bottlenecks
  – Testing overall performance

• Symptoms to look out for
  – Repeated calls to the database
  – Slow running operations
Code instrumentation and tracing

**Performance**

- **Custom applications**
  - Add code to time critical operations
    - Timers around an edit operation(s)
    - Data fetching
  - Enable DBMS tracing

- **Built-in and already available**
  - ArcSDE intercept and trace
  - ArcIMS/ArcGIS Server logs
  - DBMS trace files
Code instrumentation and tracing

Application

ArcSDE Technology
Client Libs

Network

ArcSDE Technology
gsrvr

RDBMS
Code instrumentation and tracing

Application Wrap Timer: data fetch and display

- Map display includes fetching and rendering 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
**Example: Feature Cursor Performance**

**Application Wrap Timer**

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

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

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

//Start timer and fetch the rows
start = DateTime.Now;
while ((r = rCur.NextRow()) != null)
    featCnt++;
TimeSpan result = DateTime.Now.Subtract(start);

//Release the cursor object
Marshal.ReleaseComObject(rCur);
rCur = null;
```
Example: Map Display Performance

Application Wrap Timer

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

```csharp
//m_currentMap is an iMap
//getting the active view from the current map
m_ActiveView = (IActiveView)m_currentMap;

//Refresh the Map Display
start = DateTime.Now;
    m_ActiveView.Refresh();
TimeSpan result = DateTime.Now.Subtract(start);
```

1

2
Code instrumentation and tracing

SDEINTERCEPT
Code instrumentation and tracing

SDEINTERCEPT

Application

ArcSDE Technology
Client Libs

Network

ArcSDE Technology
gsrvr

RDBMS

SDEINTERCEPT

ESRI Developer Summit 2008
Code instrumentation and tracing

SDEINTERCEPT

• Built-in ArcSDE functionality
• Instrument client calls to the ArcSDE server
• Client side - only client session
• Server side - all sessions
• Based on application’s workspace connection
• Useful to profile how many and what type of calls are made

• set SDEINTERCEPTLOC=<file location>
• set SDEINTERCEPT=<flags>
Code instrumentation and tracing

SDEINTERCEPT

• Intercept flags
  – set SDEINTERCEPT=<flags>
  – Flags to intercept network broadcasts
  – Sets type and amount of information written to intercept file
  – SDEINTERCEPT flags not set, defaults to “crwf”

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>Intercept the API command name</td>
</tr>
<tr>
<td>r</td>
<td>Intercept the Channel broadcasts read-only</td>
</tr>
<tr>
<td>w</td>
<td>Intercept the Channel broadcasts write-only</td>
</tr>
<tr>
<td>t</td>
<td>Intercept log time (minute:second)</td>
</tr>
<tr>
<td>T</td>
<td>Intercept log time (hour:minute:second)</td>
</tr>
<tr>
<td>f</td>
<td>Intercept flush immediate</td>
</tr>
</tbody>
</table>
Example: SDEINTERCEPT setup

SDEINTERCEPT

- set SDEINTERCEPTLOC
- set SDEINTERCEPT
- Start application

REM enabling SDEINTERCEPT
REM Batch File to enable SDEINTERCEPT and Start ArcMap

set SDEINTERCEPTLOC=c:\temp\client_sdeintercept
set SDEINTERCEPT=cwrfT
C:\arcgis\bin\arcmap.exe
Example: SDEINTERCEPT Output

SDEINTERCEPT

// “tail” the commands being executed
C:\temp>tail -f client_sdeintercept.001 | grep Command

Mode: Command Name, Read, Write, Time, Flush
[W 09:53:09] Command: GetDbmsInfo
[W 09:53:09] Command: GetAutoCommit
[W 09:53:09] Command: GetDbUserName
[W 09:53:09] Command: GetSdedbaName
[W 09:53:09] Command: GetConnectionServerClass
[W 09:53:09] Command: DbmsQualifyRules
[W 09:53:09] Command: CreateStream
[W 09:53:09] Command: ExecuteSpatialQuery
[W 09:53:09] Command: NextBuffer
[W 09:53:09] Command: SetConnectionProperty
[W 09:53:09] Command: GetDbmsFunction
[W 09:53:09] Command: ExecuteSpatialQuery
...

Code instrumentation and tracing

SDETRACE

Application
ArcSDE Technology
Client Libs

Network

ArcSDE Technology
gsrvr

RDBMS
Code instrumentation and tracing

SDETRACE

Application

ArcSDE Technology
Client Libs

Network

ArcSDE Technology
gsrvr

RDBMS

SDETRACE

ESRI Developer Summit 2008
Code instrumentation and tracing

**SDETRACE**

- Built-in ArcSDE functionality
- Client-side only
- Instrument client calls to the ArcSDE server
- Actual ArcSDE CAPI function names
- Based on application’s workspace connection
- Useful to profile how many and what type of calls are made

- `set SDETRACELOC=<file location>`
- `set SDETRACEMODE=<codes>`
Code instrumentation and tracing

**SDETRACE**

- **Trace modes**
  - set SDETRACEMODE=<codes>
  - Sets type and amount of information written to trace file.
  - SDETRACEMODE not set, defaults to “vhf”

<table>
<thead>
<tr>
<th>Code</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Brief mode</td>
<td>Prints function names only</td>
</tr>
<tr>
<td>v</td>
<td>Verbose mode</td>
<td>Prints names, input, output and return values</td>
</tr>
<tr>
<td>m</td>
<td>Minute Mode</td>
<td>Prints the time in [min:sec] format between function calls</td>
</tr>
<tr>
<td>h</td>
<td>Hour Mode</td>
<td>Prints the time in [hour:minute:second] format</td>
</tr>
<tr>
<td>f</td>
<td>Force Mode</td>
<td>Forces data to be written to the trace file</td>
</tr>
</tbody>
</table>
Example: SDETRACE setup

SDETRACE

• set SDETRACELOC
• set SDETRACEMODE
• Start application

REM enabling SDETRACE
REM Batch File to enable SDETRACE and Start ArcMap

set SDETRACELOC=c:\temp\client_sdetrace
set SDETRACEMODE=vhf
C:\arcgis\bin\arcmap.exe
Example: SDETRACE Output

SDETRACE

// “tail” the commands being executed
C:\temp>tail -f client_sdetrace.001 | grep ~~

~~Function: SE_connection_get_dbms_info
~~Time in Sec:1205772789.708
~~Time in Sec:1205772789.708

~~Function: SE_connection_get_transaction_autocommit
~~Time in Sec:1205772789.708
~~Time in Sec:1205772789.708

~~Function: SE_connection_get_database
~~Time in Sec:1205772789.708
~~Time in Sec:1205772789.708

~~Function: SE_connection_get_user_name
~~Time in Sec:1205772789.708
~~Time in Sec:1205772789.708

~~Function: SE_connection_get_admin_database
~~Time in Sec:1205772789.708
~~Time in Sec:1205772789.708

~~Function: SE_connection_get_sdedba_name
~~Time in Sec:1205772789.708
~~Time in Sec:1205772789.708

~~Function: SE_connection_get_server_class
~~Time in Sec:1205772789.708

....
Code instrumentation and tracing

Database Trace

Application

ArcSDE Technology
Client Libs

Network

ArcSDE Technology
gsrvr

RDBMS
Code instrumentation and tracing

**Database Trace**

- **Application**
  - ArcSDE Technology
  - Client Libs

- **Network**

- **ArcSDE Technology**
  - gsrvr

- **Database Trace**
  - RDBMS
Code instrumentation and tracing

Database Trace

- Coding the application to perform DBMS traces
  - enables operations to be profiled 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
Example: enable DBMS trace

**Database Trace**

- Determine DBMS flavor with `IDatabaseConnectionInfo2.ConnectionDBMS`
- Use the `IWorkspace.ExecuteSQL` to alter the database session to begin tracing

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

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

//Execute the SQL statement on the Workspace
workspace.ExecuteSQL(sqlStatement);
```
Example: DBMS trace output on Oracle

Database Trace: COMPLEX symbology

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

<table>
<thead>
<tr>
<th>OVERALL TOTALS FOR ALL NON-RECURSIVE STATEMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>call</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>Parse</td>
</tr>
<tr>
<td>Execute</td>
</tr>
<tr>
<td>Fetch</td>
</tr>
<tr>
<td>total</td>
</tr>
</tbody>
</table>

Misses in library cache during parse: 0

Elapsed times include waiting on following events:

<table>
<thead>
<tr>
<th>Event waited on</th>
<th>Times Waited</th>
<th>Max. Wait</th>
<th>Total Waited</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL*Net message to client</td>
<td>200</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>SQL*Net message from client</td>
<td>199</td>
<td>0.52</td>
<td>2.50</td>
</tr>
<tr>
<td>SQL*Net more data to client</td>
<td>923</td>
<td>0.00</td>
<td>0.09</td>
</tr>
<tr>
<td>direct path read</td>
<td>44</td>
<td>0.05</td>
<td>0.15</td>
</tr>
<tr>
<td>log file sync</td>
<td>12</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Example: DBMS trace output on Oracle

Database Trace: COMPLEX symbology

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

OVERALL TOTALS FOR ALL NON-RECURSIVE STATEMENTS

<table>
<thead>
<tr>
<th>call</th>
<th>count</th>
<th>cpu</th>
<th>elapsed</th>
<th>disk</th>
<th>query</th>
<th>current</th>
<th>rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Execute</td>
<td>16</td>
<td>0.17</td>
<td>0.19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Fetch</td>
<td>189</td>
<td>2.96</td>
<td>3.53</td>
<td>0</td>
<td>1605</td>
<td>0</td>
<td>17820</td>
</tr>
<tr>
<td>total</td>
<td>205</td>
<td>3.14</td>
<td>3.72</td>
<td>0</td>
<td>1605</td>
<td>0</td>
<td>17820</td>
</tr>
</tbody>
</table>

Misses in library cache during parse: 0

Elapsed times include waiting on following events:

<table>
<thead>
<tr>
<th>Event waited on</th>
<th>Times Waited</th>
<th>Max. Wait</th>
<th>Total Waited</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL*Net message to client</td>
<td>200</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>SQL*Net message from client</td>
<td>199</td>
<td>0.52</td>
<td>2.50</td>
</tr>
<tr>
<td>SQL*Net more data to client</td>
<td>923</td>
<td>0.00</td>
<td>0.09</td>
</tr>
<tr>
<td>direct path read</td>
<td>44</td>
<td>0.05</td>
<td>0.15</td>
</tr>
<tr>
<td>log file sync</td>
<td>12</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Example: DBMS trace output on Oracle

Database Trace: COMPLEX symbology

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

<table>
<thead>
<tr>
<th>call</th>
<th>count</th>
<th>cpu</th>
<th>elapsed</th>
<th>disk</th>
<th>query</th>
<th>current</th>
<th>rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Execute</td>
<td>16</td>
<td>0.17</td>
<td>0.19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Fetch</td>
<td>189</td>
<td>2.96</td>
<td>3.53</td>
<td>0</td>
<td>1605</td>
<td>0</td>
<td>17820</td>
</tr>
<tr>
<td>total</td>
<td>205</td>
<td>3.14</td>
<td>3.72</td>
<td>0</td>
<td>1605</td>
<td>0</td>
<td>17820</td>
</tr>
</tbody>
</table>

Misses in library cache during parse: 0

Elapsed times include waiting on following events:

<table>
<thead>
<tr>
<th>Event waited on</th>
<th>Times Waited</th>
<th>Max. Wait</th>
<th>Total Waited</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL*Net message to client</td>
<td>200</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>SQL*Net message from client</td>
<td>199</td>
<td>0.52</td>
<td>2.50</td>
</tr>
<tr>
<td>SQL*Net more data to client</td>
<td>923</td>
<td>0.00</td>
<td>0.09</td>
</tr>
<tr>
<td>direct path read</td>
<td>44</td>
<td>0.05</td>
<td>0.15</td>
</tr>
<tr>
<td>log file sync</td>
<td>12</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Example: DBMS trace output on Oracle

Database Trace: COMPLEX symbology

• Many layers in ArcMap document

• Parcels layer has complex symbology
  – line fill
  – two colors

• Refresh map view
Example: DBMS trace output on Oracle

Database Trace: COMPLEX symbology

//Versioned Query with the longest time
//Using more COMPLEX symbology
//Actual query shortened for readability in presentation

```
SELECT  V__3799.st_SHAPE$, V__3799.OBJECTID,  V__3799.st_points,
        V__3799.st_numpts,V__3799.st_entity,V__3799.st_minx,V__3799.st_miny,
        V__3799.st_maxx,V__3799.st_maxy,V__3799.st_area$,V__3799.st_len$,
        V__3799.st_rowid
FROM
    (Select …)
```
Example: DBMS trace output on Oracle

Database Trace: COMPLEX symbology

```
//Versioned Query with the longest time
//Using more COMPLEX symbology
//Actual query shortened for readability in presentation

SELECT  V__3799.st_SHAPE$, V__3799.OBJECTID,  V__3799.st_points,
        V__3799.st_numpts,V__3799.st_entity,V__3799.st_minx,V__3799.st_miny,
        V__3799.st_maxx,V__3799.st_maxy,V__3799.st_area$,V__3799.st_len$,
        V__3799.st_rowid
FROM
  (Select ...
```

<table>
<thead>
<tr>
<th>call</th>
<th>count</th>
<th>cpu</th>
<th>elapsed</th>
<th>disk</th>
<th>query</th>
<th>current</th>
<th>rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Execute</td>
<td>1</td>
<td>0.15</td>
<td>0.15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Fetch</td>
<td>34</td>
<td>1.06</td>
<td>1.08</td>
<td>0</td>
<td>457</td>
<td>0</td>
<td>3348</td>
</tr>
</tbody>
</table>

---

Misses in library cache during parse: 0
Optimizer mode: ALL_ROWS
Parsing user id: 15678

Elapsed times include waiting on following events:

<table>
<thead>
<tr>
<th>Event waited on</th>
<th>Times Waited</th>
<th>Max. Wait</th>
<th>Total Waited</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL*Net message to client</td>
<td>34</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>SQL*Net more data to client</td>
<td>226</td>
<td>0.00</td>
<td>0.03</td>
</tr>
<tr>
<td>SQL*Net message from client</td>
<td>34</td>
<td>0.48</td>
<td>1.19</td>
</tr>
</tbody>
</table>
```
Example: DBMS trace output on Oracle

Database Trace: COMPLEX symbology

```sql
//Versioned Query with the longest time
//Using more COMPLEX symbology
//Actual query shortened for readability in presentation

SELECT V__3799.st_SHAPE$, V__3799.OBJECTID, V__3799.st_points,
       V__3799.st_numpts,V__3799.st_entity,V__3799.st_minx,V__3799.st_miny,
       V__3799.st_maxx,V__3799.st_maxy,V__3799.st_area$,V__3799.st_len$,
       V__3799.st_rowid
FROM (Select …)
```

<table>
<thead>
<tr>
<th>call</th>
<th>count</th>
<th>cpu</th>
<th>elapsed</th>
<th>disk</th>
<th>query</th>
<th>current</th>
<th>rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Execute</td>
<td>1</td>
<td>0.15</td>
<td>0.15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Fetch</td>
<td>34</td>
<td>1.06</td>
<td>1.08</td>
<td>0</td>
<td>457</td>
<td>0</td>
<td>3348</td>
</tr>
</tbody>
</table>

**Total**

<table>
<thead>
<tr>
<th>call</th>
<th>count</th>
<th>cpu</th>
<th>elapsed</th>
<th>disk</th>
<th>query</th>
<th>current</th>
<th>rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Execute</td>
<td>1</td>
<td>0.15</td>
<td>0.15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Fetch</td>
<td>34</td>
<td>1.06</td>
<td>1.08</td>
<td>0</td>
<td>457</td>
<td>0</td>
<td>3348</td>
</tr>
</tbody>
</table>

**Total**

---

Misses in library cache during parse: 0
Optimizer mode: ALL_ROWS
Parsing user id: 15678

Elapsed times include waiting on following events:

<table>
<thead>
<tr>
<th>Event waited on</th>
<th>Times</th>
<th>Max. Wait</th>
<th>Total Waited</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL*Net message to client</td>
<td>34</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>SQL*Net more data to client</td>
<td>226</td>
<td>0.00</td>
<td>0.03</td>
</tr>
<tr>
<td>SQL*Net message from client</td>
<td>34</td>
<td>0.48</td>
<td><strong>1.19</strong></td>
</tr>
</tbody>
</table>
**Example:** DBMS trace output on Oracle

Database Trace: BASIC symbology

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

### OVERALL TOTALS FOR ALL NON-RECURSIVE STATEMENTS

<table>
<thead>
<tr>
<th>call</th>
<th>count</th>
<th>cpu</th>
<th>elapsed</th>
<th>disk</th>
<th>query</th>
<th>current</th>
<th>rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Execute</td>
<td>16</td>
<td>0.18</td>
<td>0.19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Fetch</td>
<td>189</td>
<td>2.51</td>
<td>2.62</td>
<td>0</td>
<td>1605</td>
<td>0</td>
<td>17820</td>
</tr>
<tr>
<td><strong>total</strong></td>
<td><strong>205</strong></td>
<td><strong>2.70</strong></td>
<td><strong>2.81</strong></td>
<td>0</td>
<td>1605</td>
<td>0</td>
<td>17820</td>
</tr>
</tbody>
</table>

Misses in library cache during parse: 0

Elapsed times include waiting on following events:

<table>
<thead>
<tr>
<th>Event waited on</th>
<th>Times Waited</th>
<th>Max. Wait</th>
<th>Total Waited</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL*Net message to client</td>
<td>200</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>SQL*Net message from client</td>
<td>199</td>
<td>0.50</td>
<td>1.08</td>
</tr>
<tr>
<td>SQL*Net more data to client</td>
<td>923</td>
<td>0.00</td>
<td>0.08</td>
</tr>
<tr>
<td>direct path read</td>
<td>44</td>
<td>0.05</td>
<td>0.15</td>
</tr>
<tr>
<td>log file sync</td>
<td>12</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Example: DBMS trace output on Oracle

Database Trace: BASIC symbology

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

<table>
<thead>
<tr>
<th>call</th>
<th>count</th>
<th>cpu</th>
<th>elapsed</th>
<th>disk</th>
<th>query</th>
<th>current</th>
<th>rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Execute</td>
<td>16</td>
<td>0.18</td>
<td>0.19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Fetch</td>
<td>189</td>
<td>2.51</td>
<td>2.62</td>
<td>0</td>
<td>1605</td>
<td>0</td>
<td>17820</td>
</tr>
<tr>
<td>total</td>
<td>205</td>
<td>2.70</td>
<td>2.81</td>
<td>0</td>
<td>1605</td>
<td>0</td>
<td>17820</td>
</tr>
</tbody>
</table>

Misses in library cache during parse: 0

Elapsed times include waiting on following events:

<table>
<thead>
<tr>
<th>Event waited on</th>
<th>Times</th>
<th>Max. Wait</th>
<th>Total Waited</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL*Net message to client</td>
<td>200</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>SQL*Net message from client</td>
<td>199</td>
<td>0.50</td>
<td>1.08</td>
</tr>
<tr>
<td>SQL*Net more data to client</td>
<td>923</td>
<td>0.00</td>
<td>0.08</td>
</tr>
<tr>
<td>direct path read</td>
<td>44</td>
<td>0.05</td>
<td>0.15</td>
</tr>
<tr>
<td>log file sync</td>
<td>12</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Example: DBMS trace output on Oracle

Database Trace: BASIC symbology

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

<table>
<thead>
<tr>
<th>call</th>
<th>count</th>
<th>cpu</th>
<th>elapsed</th>
<th>disk</th>
<th>query</th>
<th>current</th>
<th>rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Execute</td>
<td>16</td>
<td>0.18</td>
<td>0.19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Fetch</td>
<td>189</td>
<td>2.51</td>
<td>2.62</td>
<td>0</td>
<td>1605</td>
<td>0</td>
<td>17820</td>
</tr>
</tbody>
</table>

| total        | 205   | 2.70  | 2.81    | 0    | 1605  | 0       | 17820|

Misses in library cache during parse: 0

Elapsed times include waiting on following events:

<table>
<thead>
<tr>
<th>Event waited on</th>
<th>Times Waited</th>
<th>Max. Wait</th>
<th>Total Waited</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL*Net message to client</td>
<td>200</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>SQL*Net message from client</td>
<td>199</td>
<td>0.50</td>
<td>1.08</td>
</tr>
<tr>
<td>SQL*Net more data to client</td>
<td>923</td>
<td>0.00</td>
<td>0.08</td>
</tr>
<tr>
<td>direct path read</td>
<td>44</td>
<td>0.05</td>
<td>0.15</td>
</tr>
<tr>
<td>log file sync</td>
<td>12</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Example: DBMS trace output on Oracle

Database Trace: BASIC symbology

- Many layers in ArcMap document
- Parcels layer has basic symbology
  - simple fill
  - one color
- Refresh map view
Example: DBMS trace output on Oracle

Database Trace: BASIC symbology

//Versioned Query with the longest time
//Using more BASIC symbology
//Actual query shortened for readability in presentation

SELECT  V__3799.st_SHAPE$, V__3799.OBJECTID,  V__3799.st_points,
    V__3799.st_numpts,V__3799.st_entity,V__3799.st_minx,V__3799.st_miny,
    V__3799.st_maxx,V__3799.st_maxy,V__3799.st_area$,V__3799.st_len$,
    V__3799.st_rowid
FROM
(Select …)

<table>
<thead>
<tr>
<th>call</th>
<th>count</th>
<th>cpu</th>
<th>elapsed</th>
<th>disk</th>
<th>query</th>
<th>current</th>
<th>rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Execute</td>
<td>1</td>
<td>0.15</td>
<td>0.15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Fetch</td>
<td>34</td>
<td>0.70</td>
<td>0.71</td>
<td>0</td>
<td>457</td>
<td>0</td>
<td>3348</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>call</th>
<th>count</th>
<th>cpu</th>
<th>elapsed</th>
<th>disk</th>
<th>query</th>
<th>current</th>
<th>rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>total</td>
<td>35</td>
<td>0.85</td>
<td>0.86</td>
<td>0</td>
<td>457</td>
<td>0</td>
<td>3348</td>
</tr>
</tbody>
</table>

Misses in library cache during parse: 0
Optimizer mode: ALL ROWS
Parsing user id: 15678

Elapsed times include waiting on following events:

<table>
<thead>
<tr>
<th>Event waited on</th>
<th>Times Waited</th>
<th>Max. Wait</th>
<th>Total Waited</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL*Net message to client</td>
<td>34</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>SQL*Net more data to client</td>
<td>226</td>
<td>0.00</td>
<td>0.02</td>
</tr>
<tr>
<td>SQL*Net message from client</td>
<td>34</td>
<td>0.00</td>
<td>0.03</td>
</tr>
</tbody>
</table>
Example: DBMS trace output on Oracle

Database Trace: BASIC symbology

//Versioned Query with the longest time
//Using more BASIC symbology
//Actual query shortened for readability in presentation

SELECT  V__3799.st_SHAPE$, V__3799.OBJECTID,  V__3799.st_points,
        V__3799.st_numpts,V__3799.st_entity,V__3799.st_minx,V__3799.st_miny,
        V__3799.st_maxx,V__3799.st_maxy,V__3799.st_area$,V__3799.st_len$,
        V__3799.st_rowid
FROM
(Select …)

<table>
<thead>
<tr>
<th>call</th>
<th>count</th>
<th>cpu</th>
<th>elapsed</th>
<th>disk</th>
<th>query</th>
<th>current</th>
<th>rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Execute</td>
<td>1</td>
<td>0.15</td>
<td>0.15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Fetch</td>
<td>34</td>
<td>0.70</td>
<td>0.71</td>
<td>0</td>
<td>457</td>
<td>0</td>
<td>3348</td>
</tr>
<tr>
<td>total</td>
<td>35</td>
<td>0.85</td>
<td>0.86</td>
<td>0</td>
<td>457</td>
<td>0</td>
<td>3348</td>
</tr>
</tbody>
</table>

Misses in library cache during parse: 0
Optimizer mode: ALL_ROWS
Parsing user id: 15678

Elapsed times include waiting on following events:

<table>
<thead>
<tr>
<th>Event waited on</th>
<th>Times Waited</th>
<th>Max. Wait</th>
<th>Total Waited</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL*Net message to client</td>
<td>34</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>SQL*Net more data to client</td>
<td>226</td>
<td>0.00</td>
<td>0.02</td>
</tr>
<tr>
<td>SQL*Net message from client</td>
<td>34</td>
<td>0.00</td>
<td>0.03</td>
</tr>
</tbody>
</table>
**Example:** DBMS trace output on Oracle

**Database Trace: BASIC symbology**

//Versioned Query with the longest time
//Using more BASIC symbology
//Actual query shortened for readability in presentation

```sql
SELECT  V__3799.st_SHAPE$, V__3799.OBJECTID,  V__3799.st_points,
        V__3799.st_numpts,V__3799.st_entity,V__3799.st_minx,V__3799.st_miny,
        V__3799.st_maxx,V__3799.st_maxy,V__3799.st_area$,V__3799.st_len$,
        V__3799.st_rowid
FROM
  (Select …)
```

<table>
<thead>
<tr>
<th>call</th>
<th>count</th>
<th>cpu</th>
<th>elapsed</th>
<th>disk</th>
<th>query</th>
<th>current</th>
<th>rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Execute</td>
<td>1</td>
<td>0.15</td>
<td>0.15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Fetch</td>
<td>34</td>
<td>0.70</td>
<td>0.71</td>
<td>0</td>
<td>457</td>
<td>0</td>
<td>3348</td>
</tr>
</tbody>
</table>

**total** | 35 | 0.85 | **0.86** | 0 | 457 | 0 | 3348 |

Misses in library cache during parse: 0
Optimizer mode: ALL_ROWS
Parsing user id: 15678

Elapsed times include waiting on following events:

<table>
<thead>
<tr>
<th>Event waited on</th>
<th>Times Waited</th>
<th>Max. Wait</th>
<th>Total Waited</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL*Net message to client</td>
<td>34</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>SQL*Net more data to client</td>
<td>226</td>
<td>0.00</td>
<td>0.02</td>
</tr>
<tr>
<td>SQL*Net message from client</td>
<td>34</td>
<td>0.00</td>
<td><strong>0.03</strong></td>
</tr>
</tbody>
</table>
Example: DBMS trace output on Oracle

Database Trace: symbology rendering summary

<table>
<thead>
<tr>
<th>Time</th>
<th>COMPLEX Symbology</th>
<th>BASIC Symbology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Versioned Query</td>
<td>1.23</td>
<td>0.86</td>
</tr>
<tr>
<td>Versioned Query SQL Net Message from client</td>
<td>1.19</td>
<td>0.03</td>
</tr>
<tr>
<td>Total SQL</td>
<td>3.72</td>
<td>2.81</td>
</tr>
<tr>
<td>Total SQL Net Message from client</td>
<td>2.50</td>
<td>1.08</td>
</tr>
</tbody>
</table>
Example: DBMS trace output on Oracle

Database Trace: Performance Issue

• Receive DBMS trace

• What is the problem?
Example: DBMS trace output on Oracle

Database Trace: Performance Issue

• 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:

<table>
<thead>
<tr>
<th>Event waited on</th>
<th>Times Waited</th>
<th>Max. Wait</th>
<th>Total Waited</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL*Net message to client</td>
<td>1052</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>SQL*Net message from client</td>
<td>1052</td>
<td>29.07</td>
<td>64.27</td>
</tr>
<tr>
<td>SQL*Net more data to client</td>
<td>1579</td>
<td>0.00</td>
<td>0.02</td>
</tr>
<tr>
<td>direct path read</td>
<td>44</td>
<td>0.05</td>
<td>0.15</td>
</tr>
<tr>
<td>log file sync</td>
<td>12</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Example: DBMS trace output on Oracle

Database Trace: Performance Issue

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

### OVERALL TOTALS FOR ALL NON-RECURSIVE STATEMENTS

<table>
<thead>
<tr>
<th>call</th>
<th>count</th>
<th>cpu</th>
<th>elapsed</th>
<th>disk</th>
<th>query</th>
<th>current</th>
<th>rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse</td>
<td>6</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Execute</td>
<td>54</td>
<td>0.14</td>
<td>0.15</td>
<td>0</td>
<td>207</td>
<td>649</td>
<td>113</td>
</tr>
<tr>
<td>Fetch</td>
<td>1025</td>
<td>3.34</td>
<td>3.54</td>
<td>50</td>
<td>677710</td>
<td>0</td>
<td>15054</td>
</tr>
<tr>
<td><strong>total</strong></td>
<td>1085</td>
<td>3.48</td>
<td>3.70</td>
<td>50</td>
<td>677917</td>
<td>649</td>
<td>15167</td>
</tr>
</tbody>
</table>

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

Elapsed times include waiting on following events:

<table>
<thead>
<tr>
<th>Event waited on</th>
<th>Times Waited</th>
<th>Max. Wait</th>
<th>Total Waited</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL*Net message to client</td>
<td>1052</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>SQL*Net message from client</td>
<td>1052</td>
<td>29.07</td>
<td>64.27</td>
</tr>
<tr>
<td>SQL*Net more data to client</td>
<td>1579</td>
<td>0.00</td>
<td>0.02</td>
</tr>
<tr>
<td>direct path read</td>
<td>44</td>
<td>0.05</td>
<td>0.15</td>
</tr>
<tr>
<td>log file sync</td>
<td>12</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

ESRI Developer Summit 2008
Example: DBMS trace output on Oracle

Database Trace: Performance Issue

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

OVERALL TOTALS FOR ALL NON-RECURSIVE STATEMENTS

<table>
<thead>
<tr>
<th>call</th>
<th>count</th>
<th>cpu</th>
<th>elapsed</th>
<th>disk</th>
<th>query</th>
<th>current</th>
<th>rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse</td>
<td>6</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Execute</td>
<td>54</td>
<td>0.14</td>
<td>0.15</td>
<td>0</td>
<td>207</td>
<td>649</td>
<td>113</td>
</tr>
<tr>
<td>Fetch</td>
<td>1025</td>
<td>3.34</td>
<td>3.54</td>
<td>50</td>
<td>677710</td>
<td>0</td>
<td>15054</td>
</tr>
<tr>
<td>total</td>
<td>1085</td>
<td>3.48</td>
<td>3.70</td>
<td>50</td>
<td>677917</td>
<td>649</td>
<td>15167</td>
</tr>
</tbody>
</table>

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

Elapsed times include waiting on following events:

<table>
<thead>
<tr>
<th>Event waited on</th>
<th>Times Waited</th>
<th>Max. Wait</th>
<th>Total Waited</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL*Net message to client</td>
<td>1052</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>SQL*Net message from client</td>
<td>1052</td>
<td>29.07</td>
<td>64.27</td>
</tr>
<tr>
<td>SQL*Net more data to client</td>
<td>1579</td>
<td>0.00</td>
<td>0.02</td>
</tr>
<tr>
<td>direct path read</td>
<td>44</td>
<td>0.05</td>
<td>0.15</td>
</tr>
<tr>
<td>log file sync</td>
<td>12</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Example: DBMS trace output on Oracle

Database Trace: Performance Issue

• Receive DBMS trace

• What is the problem?

• Look at the entire application/system
  – Determine where bottlenecks are

• Iterative process
Code instrumentation and tracing

Summary

Application

ArcSDE Technology
Client Libs

Network

ArcSDE Technology
gsrvr

RDBMS
Code instrumentation and tracing

Summary

Application

ArcSDE Technology
Client Libs

Network

ArcSDE Technology
gsrvr

RDBMS

Application Wrap Timer
Code instrumentation and tracing

Summary

Application

ArcSDE Technology
Client Libs

Network

Application Wrap Timer

SDEINTERCEPT

ArcSDE Technology
gsrvr

RDBMS

ESRI Developer Summit 2008
Code instrumentation and tracing

Summary

Application
ArcSDE Technology
Client Libs

Network

ArcSDE Technology
gsrvr

SDEINTERCEPT

RDBMS

SDETRACE

Application Wrap Timer
Code instrumentation and tracing

Summary

Application

ArcSDE Technology
Client Libs

Network

SDEINTERCEPT

ArcSDE Technology
gsrvr

Application Wrap Timer

SDETRACE

Database Trace

RDBMS
Code instrumentation and tracing

Summary and Best Practices

• Several instrumentation and tracing methods
• Look at entire application stack
• Application running slow
  – Iterative process throughout development and release
  – Tools to determine where performance problems are
  – Determine where to look next
  – i.e. start at application level, or maybe the initial information is a DBMS trace, use built-in tools
• Geodatabase Toolset (GDBT)
In Conclusion...

• **Other recommended sessions**
  
  Effective Geodatabase Programming
  – Thursday 1:30 – 2:45
  
  Distributed Geodatabase Development
  – Wednesday 4:30 – 5:45
  – Thursday 8:30 – 9:45
  
  Developing with Rasters in ArcGIS
  – Thursday 8:30 – 9:45
In Conclusion...

• All sessions are recorded and will be available on EDN
  – Slides and code will also be available

• Please fill out session surveys!

• Still have questions?
  1. Tech talk, Demo Theatres, Meet the Team
  2. “Ask a Developer” link on web page
     • www.esri.com/devsummit/techquestions
Tech Talk
Map