



Improved CSDK for Cheetah and beyond...

Sheshnarayan Agrawal
IBM, India

Session Code: B11

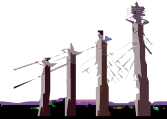
Day, April 29, 2008 • 03:30 p.m. – 04:30 p.m.

2008 IIUG Inform*i*x Conference



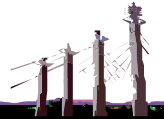
Agenda

- Quick Introduction – CSDK
- Tracing / Problem diagnostic
- What's new in CSDK Cheetah
- Improvements - Performance/Functionality
- API / Application performance tips
- Applications support in CSDK
- Going Forward – In Future



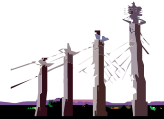
Introduction - CSDK

- CSDK is bundle of various connectivity tools for all the Informix database offerings (i.e. SE, 5x, 7x, XPS, 9x, 10x, 11x ...)
 - ESQL/C – Native, C based
 - ODBC – Open standard, C based
 - ADO.NET – Microsoft Technology, C# based
 - OLEDB – Microsoft Technology, COM+ based
 - LibC++ – Object Oriented, C++ based
- Runtime version of CSDK called I-Connect
- Compatible with multi-threaded applications



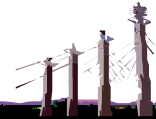
Introduction - CSDK Cont ...

- Industry standard specification compatibility
 - ODBC – 3.5 Spec, core level 3 APIs
 - OLEDB – 2.1
 - ADO .NET – Framework 2.0

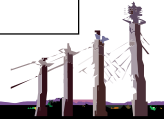
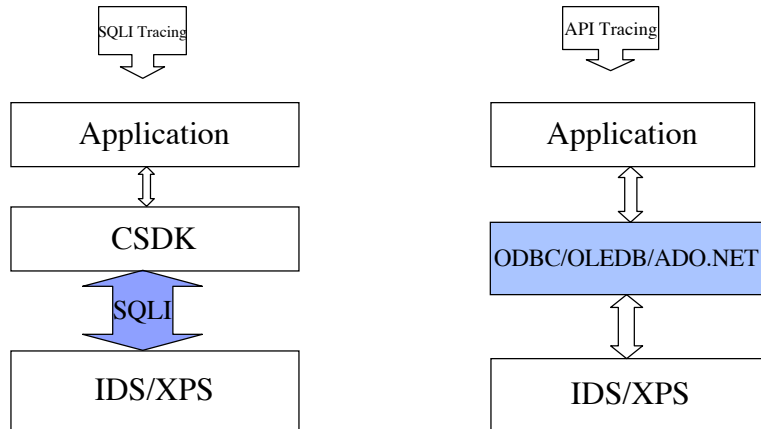


Introduction - CSDK Cont ...

- Availability on various platforms
 - CSDK is available across most of the platforms out there in the market i.e.
 - AIX
 - Solaris
 - HP
 - Linux
 - Windows
 - Mac OS
- CSDK / I-Connect is available both in 32 & 64 bit variations



CSDK : Tracing / Problem diagnostics



SQLI : Tracing / Problem diagnostics

SQLIPRINT

This utility can be used to trace the flow of information between client and server. This can be used for any CSDK components.

Location

Unix -> \$INFORMIXDIR/bin/[sqliprint](#)

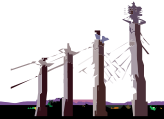
Windows -> %INFORMIXDIR%\bin\[sqliprt.exe](#)



SQLI : Tracing / Problem diagnostics

SQLIPRINT – How does it work?

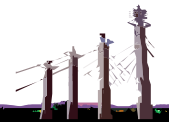
1. Set environment variable SQLIDEBUG i.e.
“setenv SQLIDEBUG 2:sqli”
2. When you run your application, file called
sqli_<alphanumeric value> will be created
3. \$sqliprint sqli_<alphanumeric value> -o
sqli_out.txt
\$sqliprint -help



SQLI : Tracing / Problem diagnostics

SQLIPRINT – Features / Benefits

- It's available across all the platforms
- It's multithreaded
- Easy problem determination
- This works even if client/server are on different machines



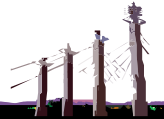
OLEDB/ODBC : Tracing / Problem diagnostics

OLEDB Provider

- Set environment variable, IFXOLEDBTRACE, i.e. set
IFXOLEDBTRACE=oledb_trace.txt

ODBC Driver

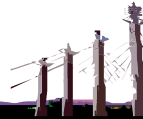
- On Windows, use trace provided by Driver Manager
- On Unix, in odbc.ini file, TRACE=1, TRACEFILE=<file
name>, TRACEDLL=idmrs09a.so



ADO.NET : Tracing / Problem diagnostics

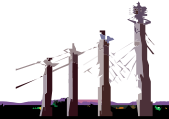
ADO.NET Provider

- Two environment variables, IFXDOTNETTRACE and IFXDOTNETTRACEFILE
- IFXDOTNETTRACE values
 - 0 No tracing
 - 1 Tracing of API entry and exit, with return code
 - 2 Tracing of API entry and exit, with return code, plus tracing of parameters to the methods



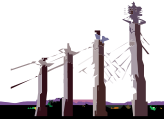
Cheetah: New Platforms

- Windows 64-bit
- Mac OS x Leopard 10.5.2
- Certified on Windows Vista



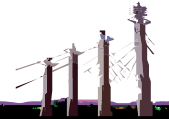
Cheetah: Label-Based Access Control (LBAC)

- LBAC is based on the design in DB2
 - Essentially the same
 - Control who can see a particular column
 - Control who can see a particular row
- CDSK components are enabled to handle new SQL queries
 - i.e.
 - CREATE SECURITY LABEL ...
 - CREATE SECURITY POLICY ...
 - GRANT SECURITY LABEL ...
 - ALTER SECURITY LABEL ...
 - etc



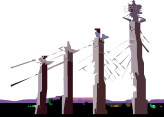
Cheetah: BIGINT/BIGSERIAL data type support

- Support from IDS 11.50.xC1, CSDK 3.50.xC1 onwards
- Similar to INT8/SERIAL8
- Benefits over INT8/SERIAL8
 - INT8/SERIAL8 is 10 byte structure
 - BIGINT/BIGSERIAL is 8 byte
 - With BIGINT/BIGSERIAL - no internal conversion
 - BIGINT/BIGSERIAL maps to:
 - Unix 32-bit: `long long`
 - Unix 64-bit: `long`
 - Windows 32/64 bits: `__int64`



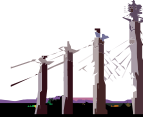
BIGINT/BIGSERIAL Usage - ESQL/C

- ANSI Strict C Compiler on Unix 32-bit
 - Define NOBIGINT
- EXEC SQL BEGIN DECLARE SECTION;
 - **bigint** big_var=123456789;
- EXEC SQL END DECLARE SECTION;
- EXEC SQL CREATE TABLE bigint_tab (bigint_col1 **bigint**, int_col1 int);
- EXEC SQL INSERT INTO bigint_tab VALUES(999999999999999999,12);
- EXEC SQL SELECT bigint_col, int_col1 into :big_var, :int_var FROM bigint_tab where int_col1=12;
- Various new functions
 - biginttoint2()
 - bigintcvasc()
 - biginttoasc()
 - bigintcvifx_int8()
 - biginttoifx_int8()
 - Etc..



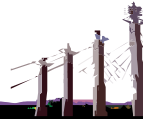
BIGINT/BIGSERIAL Usage - ODBC

- **SQLINFX_BIGINT** `bigint_var;`
- SQLINFX_BIGINT is defined in \$INFORMIXDIR/incl/cli/infxcli.h
- SQLExecDirect(hstmt, "CREATE TABLE biginttab(bigint_col **bigint**)", SQL_NTS);
- SQLExecDirect(hstmt, "INSERT INTO biginttab VALUES (4294967295000)", SQL_NTS);
- SQLBindParameter(hstmt, 1, SQL_PARAM_INPUT, SQL_C_BINARY, **SQL_INFX_BIGINT**, 0, 0, **&bigint_var**, 0, &dLen);
- SQLBindCol(hstmt, 1, SQL_C_BINARY, **&bigint_var**, sizeof(**SQLINFX_BIGINT**), &dind);
- Catalog APIs i.e. SQLColumns, SQLColumnAttributes etc will return appropriate metadata information for BIGINT/BIGSERIAL data types
- Supports all data type conversions as supported by existing INT8/SERIAL8
 - i.e. BIGINT to CHAR, CHAR to BIGINT, NUMERIC TO BIGINT etc



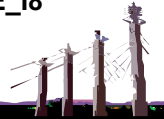
BIGINT/BIGSERIAL Usage – ADO.NET

- **Bigint** and **BigSerial** two new members in **IfxType** class
- New method **GetBigInt()** added in **IfxDataReader** class
- ExecNonQuery("create table biginttable(int_col integer,bigint_col **bigint**)");
- CommandText = "insert into biginttable values(?,?,?);";
- param1 = new IfxParameter("intparam", IfxType.Integer);
- param2 = new IfxParameter("bigintparam", **IfxType.BigInt**);
- param1.Value = 1234;
- param2.Value = **9223372036854775807**; // Max value of BIGINT
- Parameters.Add(param1);
- Parameters.Add(param2);
- ExecuteNonQuery();
- CommandText = "select * from biginttable";
- IfxDR = ExecuteReader();
- int64Val = IfxDR.**GetBigInt**(col_num);



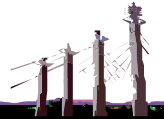
BIGINT/BIGSERIAL Usage – OLEDB

- Similar to INT8/SERIAL8
- As such nothing is exposed to be used directly by applications
- Internally OLEDB added support to handle these new data types
- All required conversions are supported
- BIGINT/BIGSERIAL is mapped to OLEDB type DBTYPE_I8



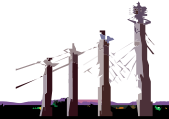
Cheetah: IDS ADO.NET provider

- .NET provider for 2.0 Framework Support
- Visual Studio 2005 Add-ins for .NET



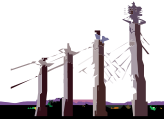
Cheetah: ADO.NET 2.0 FW

- Both 1.1 and 2.0 ADO.NET Providers shipped
 - %INFORMIXDIR%/bin/netf11
 - %INFORMIXDIR%/bin/netf20
- Multiple Active Result Set (MARS) support



Cheetah: ADO.NET 2.0 FW Support

- .NET provider for 2.0 framework
 - Compiled with 2.0 .NET Framework
 - Complies with ADO.NET 2.0 specification
 - Facilitate generic Coding with the ADO.NET 2.0
 - The base-class based provider model
 - Provider Factory
- New classes
 - Connection String Builder
 - Enumerating Data Sources



Cheetah: ADO.NET 2.0 FW Support

- .NET provider for 2.0 framework
 - Interface-based approach in ADO.NET 1.1 makes writing generic applications difficult
 - One must create a concrete instance of a specific class

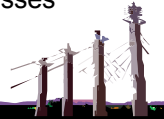
```
lfxConnection conn = new lfxConnection(  
    "Host=myhost;Service=1541;database=mydb....." );
```

- Provider factories in ADO.NET 2.0 solve this

```
DbProviderFactory fact =  
    DbProviderFactories.GetFactory("IBM.Data.Informix");  
DbConnection conn = fact.CreateConnection();  
conn.ConnectionString = "Server=danon_ids10;User" +  
    "ID=informix;password=inbluemix;Database=perf;";  
conn.Open();
```

- Provider factory have methods to create other classes

```
fact.CreateCommand();  
fact.CreateDataAdapter();  
fact.CreateCommandBuilder();  
fact.CreateDataSourceEnumerator();  
...
```



Cheetah: ADO.NET 2.0 FW Support

```
DbProviderFactory factory =  
DbProviderFactories.GetFactory("IBM.Data.Informix");
```

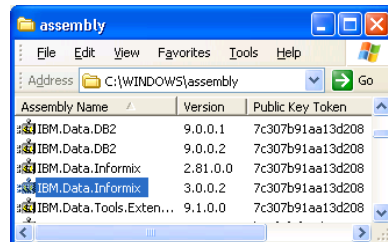
Application

```
<add name="IBM Informix .NET Data Provider"  
invariant="IBM.Data.Informix"  
description="IBM Informix Data Provider for .NET Framework 2.0"  
type="IBM.Data.Informix.IfxFactory, IBM.Data.Informix, Version=3.0.0.2,  
Culture=neutral, PublicKeyToken=7c307b91aa13d208" />
```

machine.config

C:\WINDOWS\Microsoft.NET\Framework
v2.0.50727\CONFIG\machine.config

Machine.config
entries are added by
installer by running
ifxnmpcfg.exe



Assembly Name	Version	Public Key Token
IBM.Data.DB2	9.0.0.1	7c307b91aa13d208
IBM.Data.DB2	9.0.0.2	7c307b91aa13d208
IBM.Data.Informix	2.81.0.0	7c307b91aa13d208
IBM.Data.Informix	3.0.0.2	7c307b91aa13d208
IBM.Data.Tools.Extens...	9.1.0.0	7c307b91aa13d208

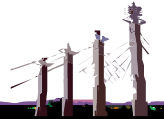
GAC

GAC entries are
added by CSDK
installer

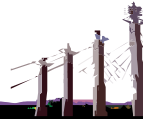
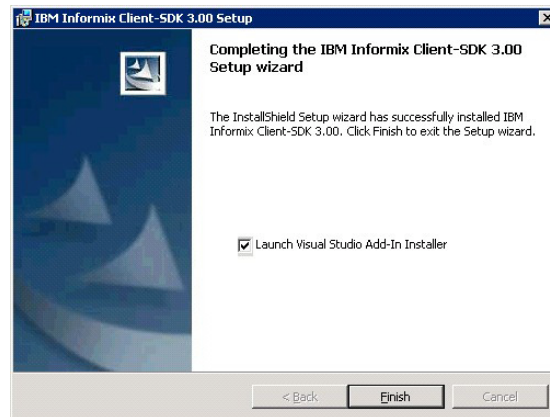


Cheetah: Visual Studio Add-Ins Support - Highlights

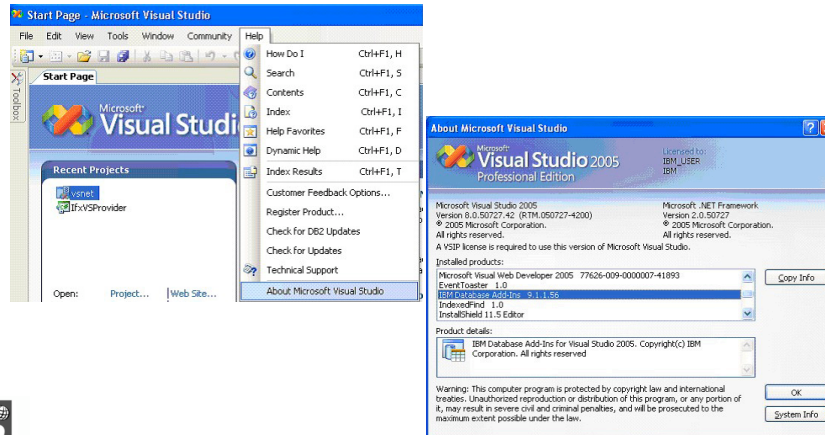
- **Integrated installation:**
 - IBM Database Add-ins for Visual Studio 2005 installation is integrated with CSDK 300TC1 installation
- **Server Explorer:**
 - Seamless integration with Microsoft Server Explorer
- **Windows application development:**
 - Build Windows .NET applications for Informix without writing any code



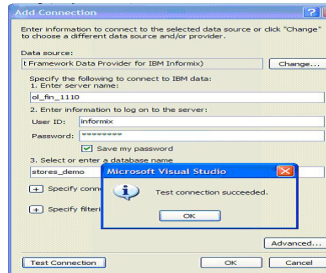
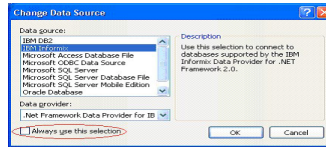
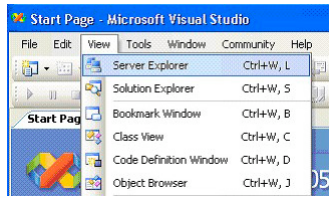
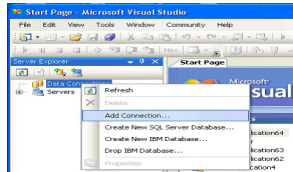
Visual Studio Add-Ins support - Installation



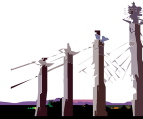
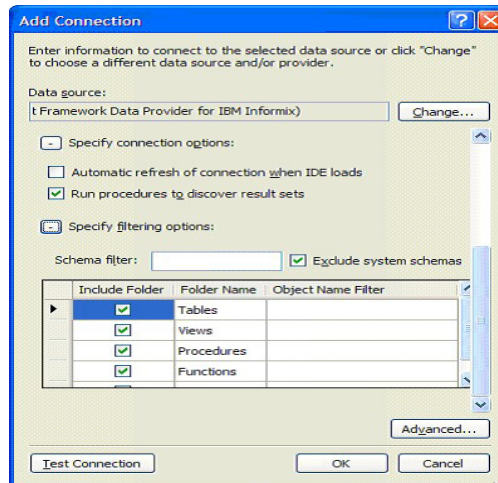
Visual Studio Add-Ins support - Installation



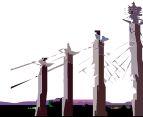
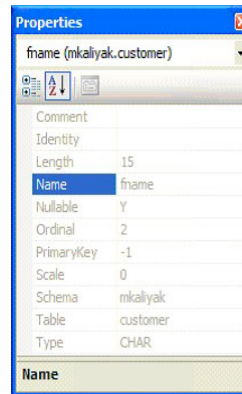
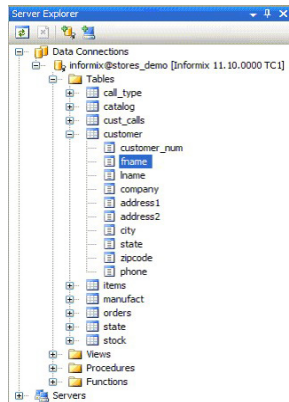
VSAI: Adding Connections



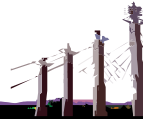
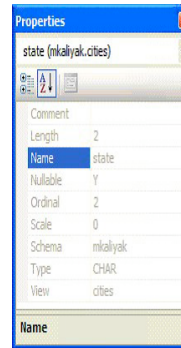
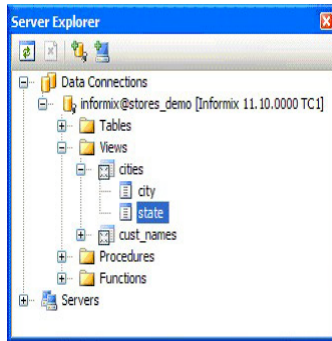
VSAI: Configurable Add Connection Objects



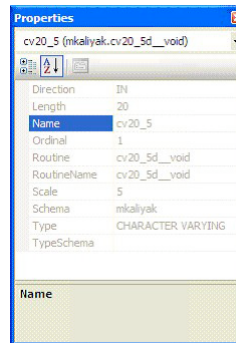
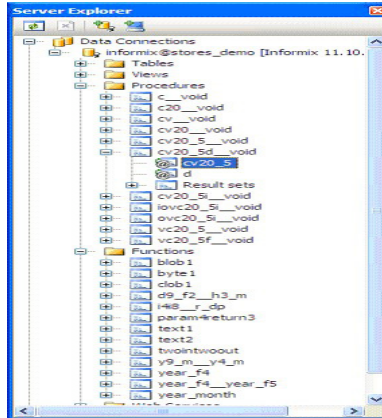
VSAI: Server Explorer Integration - Table Enumeration



VSAI: Server Explorer Integration – View Enumeration

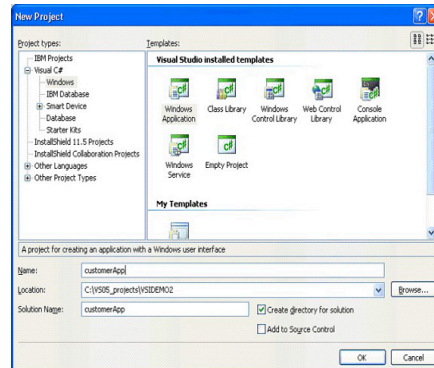


Server Explorer Integration – Procedure / Function Enumeration



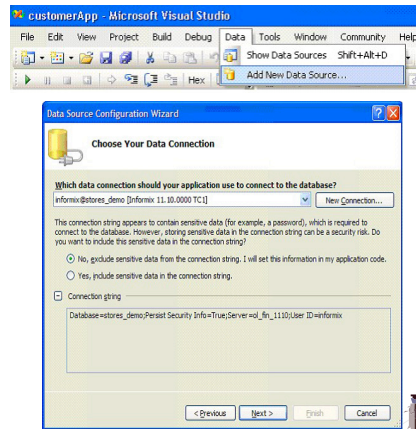
Developing C# Windows Application

1. Add a connection to an Informix database.
2. Create a new Windows application using the File > New > Project menu. Call it "customerApp".



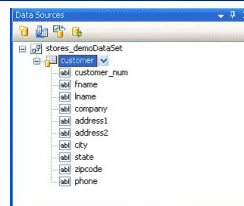
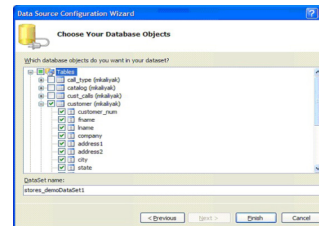
VSAI: Developing C# Windows Application

3. Using the top-level menu, choose **Data > Add New Data Source** to invoke the wizard.
4. In the wizard, keep the default selection for "Database". Click Next.
5. From the list of connections, select the connection to sample what was created earlier. Select the option to include sensitive data in the connection string. Click Next.



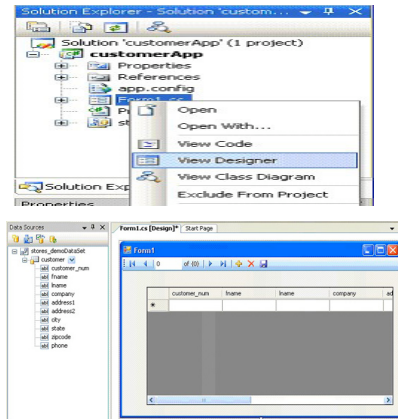
VSAI: Developing C# Windows Application

6. Select the **Customer** table from the list of tables, and click **Finish**.
7. A data source for the **Customer** table will be added in the Data Sources window. You can make the Data Sources window visible using the top-level menu, and selecting **Data** -> **Show Data Sources**.



VSAI: Developing C# Windows Application

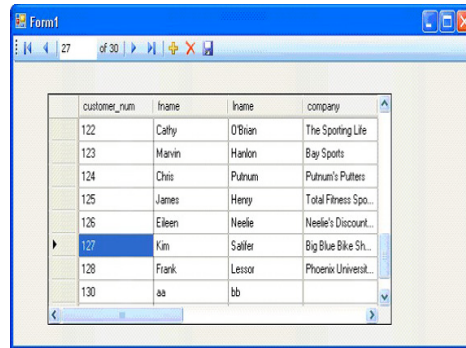
8. Ensure the Form1.cs is open in design mode.
9. Drag and drop the data source from the Data Sources window onto the form. Notice the default-generated user interface.



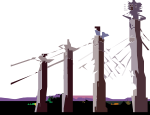
VSAI: Developing C# Windows Application

10. Run the application

You can use the pre-created user interface to navigate the rows. You can also add new rows, delete and update existing rows. Clicking the Save button will save the changes to the server.

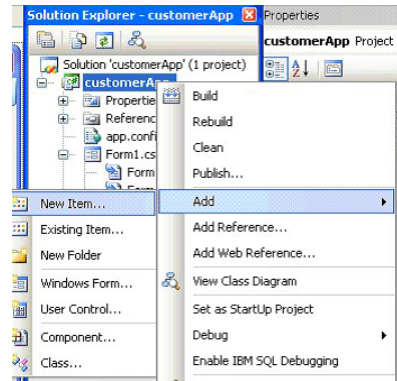


customer_num	lname	fname	company
122	Cathy	O'Brian	The Sporting Life
123	Marvin	Harkon	Bay Sports
124	Chris	Putnum	Putnum's Putters
125	James	Henry	Total Fitness Spo...
126	Eileen	Neele	Neele's Discount...
127	Kim	Salter	Big Blue Bike Sh...
128	Frank	Lessor	Phoenix Universit...
130	aa	bb	

www.iiug.org

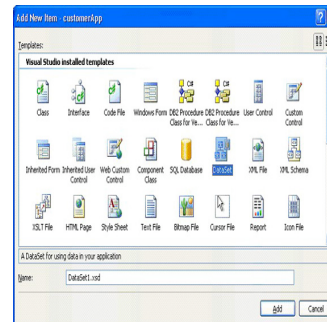
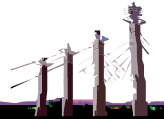
VSAI: Application Development using DataSet

1. Using the Microsoft Solution Explorer, right-click on the customerApp, and select Add > New Item.
2. Select DataSet. Click Add. The Microsoft DataSet designer will open.



VSAI: Application Development using DataSet

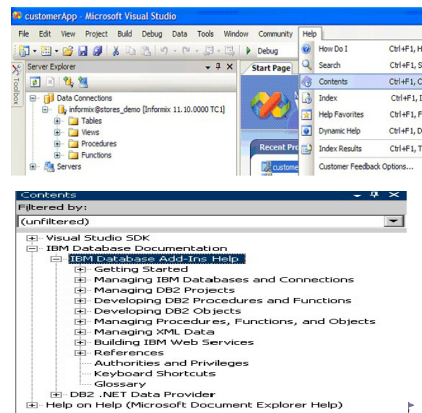
3. From the Microsoft Server Explorer, drag and drop another table, named "orders", onto the open designer. This creates the data source for "orders".
4. You can drag and drop the newly created data source onto a fresh form, as mentioned in Steps 4 and 5 of the Windows application development section, by running the form, insert/update/delete operations can be done in the table "orders".

www.iiug.org

VSAI: Help Documentation

The documentation help for IBM Database Add-Ins for Visual Studio 2005 is integrated into IDE. You can see this in the steps below:

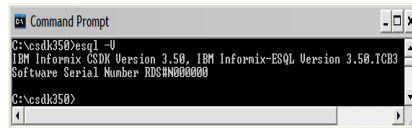
1. Click Help > Contents.
2. Click IBM Database Documentation > IBM Database Add-Ins Help. This documentation contains help for both DB2 and Informix, whereas DB2 specific help pages are marked with DB2



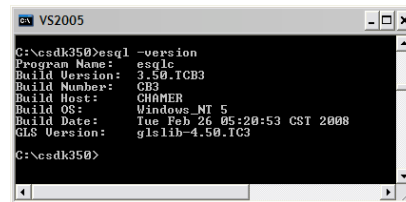
www.iiug.org

Cheetah: ESQL/C –version support

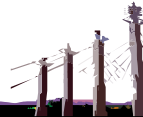
- Existing: esql -V
- Additional support for
“esql –version” similar to
“onstat –version”



```
Command Prompt
C:\ncadk350>esql -V
IBM Informix CSDK Version 3.50, IBM Informix-ESQL Version 3.50.TCB3
Software Serial Number RDSHN8000000
C:\ncadk350>
```

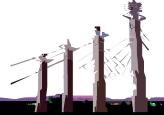
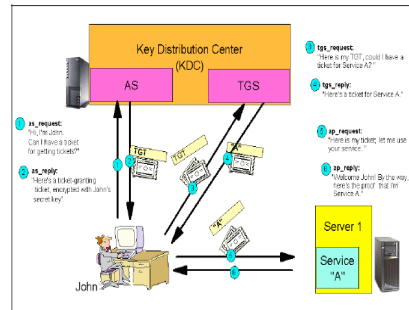


```
VS2005
C:\ncadk350>esql -version
Program Name:    esqlc
Build Version:   3.50.TCB3
Build Number:    CB3
Build Host:      CHARMER
Build OS:        Windows_NT 5
Build Date:      Tue Feb 26 05:20:53 CST 2008
GLS Version:     glslib-4.50.TC3
C:\ncadk350>
```



Cheetah: Single Sign-On (SSO) Client Configuration

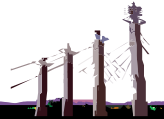
- This assumes that we have KDC (Key Distribution Center) configured, credentials / principal has been established and possibly any other setup is already in place.
- To ensure Kerberos setup is correct, you can simply test the "telnet <hostname>" without UID/PWD
- **Client side:**
 - Configure sqlhosts file. For GSSCSM use s=7 and csm=(anystring)
 - sso_11x onsoctcp lxvm-1141 9019 s=7,csm=(DCEGSS)
 - Configure concsm.cfg file in \$INFORMIXDIR/etc directory
 - DCEGSS("/work/manojm/sqldist/ssodist/lib/csm/igsss11a.so", "", "")



Cheetah: Secure Socket Layer (SSL) Support

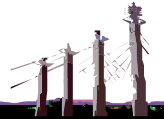
Secured Socket Layer (SSL) protocol is a communication protocol that provides privacy and integrity for data communication over the network. SSL uses encryption to provide reliable end-to-end secure and authenticated TCP/IP connection between two points over a network.

- On the client side two new protocols added
 - ONSOCSSL – For SQLI Clients
 - DRSOCSSL – For DRDA Clients (DB2)
- Example:
 - india_on **onsocssl** india-host1 india_serv
 - bangalore_on **drsocssl** india-host2 bangalore_serv



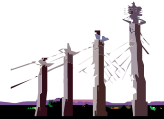
Cheetah: ESQL/C support for Static Lib on Windows

- ESQL/C product libraries are supported as both static and dynamic libraries. By default the script uses dynamic libraries. To use static libraries, the 'esql' script must be invoked with the -static option:
- **esql -static**
- Currently this -static option only works with the Microsoft Visual Studio Compiler
- The static library name is **isqlt09s.lib**. This library is installed in %INFORMIXDIR%/lib
- **Benefit**
 - No need to ship ESQL/C libs with your applications



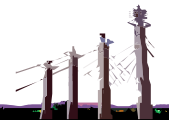
Cheetah: Trigger Enhancements

- Multiple INSERT, DELETE, SELECT and UPDATE triggers now allowed on the same table
- Multiple SELECT, UPDATE triggers now allowed on the same column
- It is now possible to access the relevant old and new values from procedures invoked from “FOR EACH ROW” trigger action statement
- **CSDK is enabled to support new SQL queries for multiple trigger**



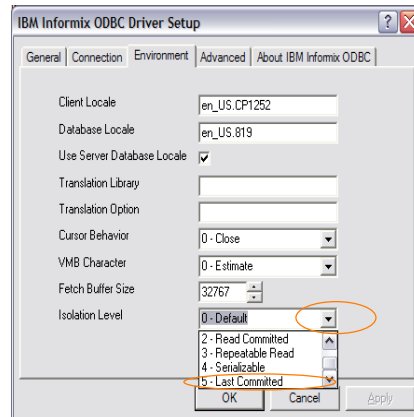
Cheetah : Sub-Query in the FROM Clause

- ANSI standard syntax now accepted for subqueries in FROM clause of SELECT statement.
 - Easier to generate dynamic queries without relying on temp tables, or views. ANSI compliant syntax makes development and porting easier.
- **CSDK is enabled to support new SQL queries for Sub-Query in the FROM Clause**



Cheetah: Last Committed (Optimistic Concurrency)

- Returns the most recently committed version of the rows, even if another concurrent session holds an exclusive lock.
- Isolation level designed to increase concurrency & throughput
- **How to enable?**
 - Execute the following SQL statement
 - **SET ISOLATION TO COMMITTED READ LAST COMMITTED**
- In ODBC
 - ODBC.INI or Connection String: ISOLATIONLEVEL=5
 - Windows: Environment Tab drop box



Last Committed Read Example - Transfer \$400 from Cust_ID
#1234 to Cust_ID #3456

Time	Transaction1	Transaction2 COMMITTED READ (default in logged database)	Transaction3 DIRTY READ	Transaction4 LAST COMMITTED (New in IDS 11.10)
1	-- Current balance of customer 1234 is 1250.00			
2	set isolation to read committed;			
3	begin work;			
4	update cust_tab set balance = balance - 400 where cust_id = 1234;	begin work;	begin work;	begin work;
5	-- balance of customer 1234 is 850.00			
6	update cust_tab set balance = balance + 400 where cust_id = 3456;	select balance from cust_tab where custid = 1234; -- wait for the lock on row for customer 1234	select balance from cust_tab where custid = 1234; -- No waiting. will return 850.00.	select balance from cust_tab where custid = 1234; -- No waiting -- will return 1250.00
7	insert into daily_tab("transfer", 1234, 3456, 400);	-- Status: lock wait	-- Continue processing	-- Continue processing
8	Commit work;	-- Status: lock wait	-- do more	-- do more
		--will return 850.00	-- do more	-- do more

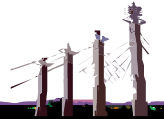


www.iiug.org



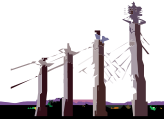
Cheetah: MACH11 Connection Manager

- Managing Connections for High Availability Clusters
- Connection Manager dynamically routes client application connection requests to the most appropriate server in a high-availability cluster.
- The Connection Manager utility is installed with CSDK.
- The Connection Manager is primarily documented in the Administration Guide and the Administrator's Reference.

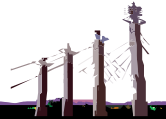


Cheetah: Visual Studio 2005 Compiler Support

- One more application development environment

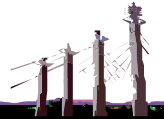


Recent Improvements



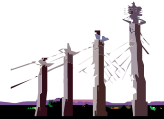
IDS .NET : Improved Connection Pooling

- “Min Pool Size” support
- Two new public properties to better monitor connection pool load
- GetIdleConnectionsCount
 - This will return the total number of connections currently opened but unused.
- GetActiveConnectionsCount
 - This will return the total number of connections currently opened and in use.



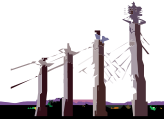
IDS .NET : Connection Pooling Example

- `IfxConnection conn=new IfxConnection(" ....; Min Pool Size=4;Max Pool Size=50;");`
- `conn.Open();`
- `idlecount = conn.GetIdleConnectionsCount;`
- `activecount = conn.GetActiveConnectionsCount;`
- `Console.WriteLine("Idle conns = " + idlecount + "and Active conns = " + activecount);`
- **// Ouput : Idle conns = 3 and Active conns = 1**
- `// Lets make another connection using the same connection string as above.`
- `IfxConnection conn1=new IfxConnection(" ....; Min Pool Size=4;Max Pool Size=50;");`
- `conn1.Open();`
- `idlecount = conn1.GetIdleConnectionsCount;`
- `activecount = conn1.GetActiveConnectionsCount;`
- `Console.WriteLine("Idle conns = " + idlecount + "and Active conns = " + activecount);`
- **// Output : Idle conns = 2 and Active conns = 2**



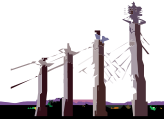
IDS .NET : Improved Result Set Fetch Performance

- Available with both .NET providers
- Win32 calls from managed code has been reduced significantly
- This is mainly in the data retrieval part
- You we will see significant performance enhancement



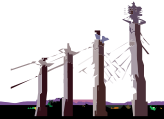
IDS .NET : COM+ Isolation setting

- Isolation level set for COM components now gets propagated to .NET Provider
- This is supported only in .NET 2.0 Framework:
System.Transaction.IsolationLevel
- Example:
 - ServiceConfig config = new ServiceConfig();
 - `config.IsolationLevel = TransactionIsolationLevel.ReadUncommitted;`



ODBC: Improved Insert Cursor Support

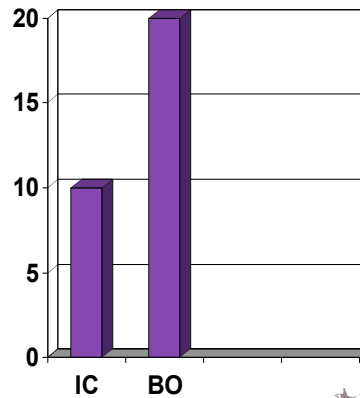
- This option reduces the number of network messages sent to and from the server by buffering the data at client layer
- VARCHAR and LVARCHAR data types support is added



ODBC: Insert Cursor Vs SQLBulkOperations

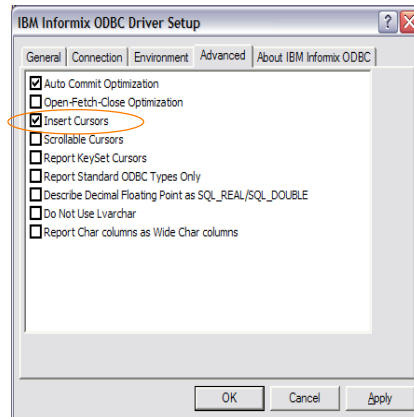
- **Sample Data:**

- 7 INT8, 3 SMALLINT, 3 INT, 10 VARCHAR(254), 2 LVARCHAR columns.
- 3000 rows
- **Insert Cursor Limitation** : TEXT and BYTE data types are not supported. If its enabled and these data types are there, it will operate in SQLBulkOperations mode. It will not report error.



ODBC: Insert Cursor Vs SQLBulkOperations

- **How to enable INSERT CURSOR?**
- **Through Application – 2 ways**
 - SQLSetStmtAttr
(SQL_ENABLE_INSERT_CURSOR)
 - SQLSetStmtAttr
(SQL_ATTR_PARAMSET_SIZE)
 - **OR**
 - connection attribute string:
"EnableInsertCursors=1"
- **Through ODBC.INI file**
 - UNIX: "EnableInsertCursors=1"
 - Windows: DSN->Advanced
- **Insert Cursor Limitation** : TEXT and BYTE data types are not supported. If its enabled and these data types are there, it will operate in SQLBulkOperations mode. It will not report error.



ODBC: True VARCHAR support

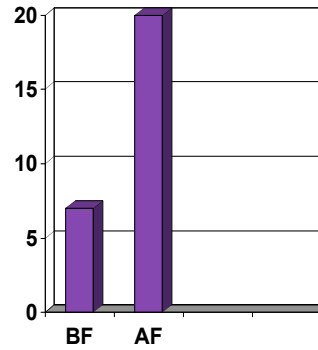
- **CSDK 2.90.TC6 onwards**
ODBC started supporting
True VARCHAR data type

- **Benefits**

- More amount of data in per round trip
- Huge performance improvement if "SELECT" query contains VARCHAR data type

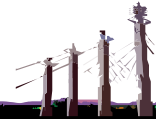
- **Sample Data:**

- 7 INT8, 3 SMALLINT, 3 INT, 10 VARCHAR(254), 2 LVARCHAR columns.
- Retrieve 3000 rows



TBD

- BF - Before
- AF - After



ODBC : “Skip Parser” support in

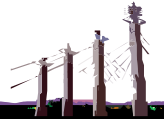
- This feature intend to enhance the performance of overall application by turning off the SQL query parsing at the ODBC end.
- There are few ways you can use this functionality
 1. In Connection String. For example:
connString=".....;**SKIPPARSING=1**;" Setting 1 will enable the “Skip Parsing”

2. In SQLSetConnectAttr. For example:
SQLSetConnectAttr(hdbc, SQL_INFX_ATTR_SKIP_PARSING,
(SQLPOINTER)SQL_TRUE, SQL_IS_USMALLINT);

When this is enabled at connection level all statement handles allocated using this connection will inherit this property.

3. In SQLSetStmtAttr. For Example:
SQLSetStmtAttr(hstmt, **SQL_INFX_ATTR_SKIP_PARSING**, (SQLPOINTER)SQL_TRUE,
SQL_IS_USMALLINT);
4. ODBC.INI (only Unix) by using "SKIPPARSING=1" value.

- Limitations:
 - Don't enable if you intend to use ODBC Escape sequences in your query.
 - Don't enable if you intend to call any catalog functions (i.e. SQLColumns, SQLProcedureColumns, SQLTables etc) after running your intended SQL query.



ODBC : Improved Unicode support

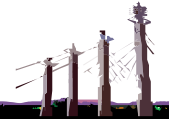
- UCS-2 support on native UCS-4 platforms
- ANSI application

Platform	Driver Manager	Default odbc.ini
UNIX	Data Direct/UnixODBC	UTF-8
AIX 32-bit	DMR or none	UCS-2
UNIX	DMR or none	UCS-4
Windows	Windows DM	N/A



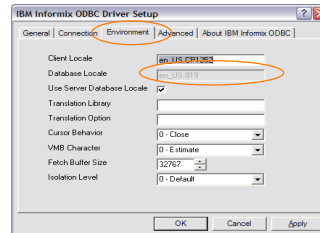
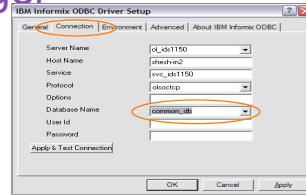
ODBC : Improved integration with MSDTC

- MSDTC support in ODBC has been improved to handle higher transactional loads efficiently
- Better enlistment strategy – with respect to connection pooling
- Many defect fixes to resolve session hang, crash etc



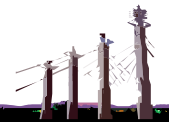
ODBC : Improved integration with unixODBC Driver Manager

- Improved integration with unixODBC 2.2.12 Driver Manager
- Improved Security – better password management
- Windows: Better DSN-less connection handling – No need of SQLHOST
- Windows: Auto DB_LOCALE setup in ODBC DSN



Improved integration with SQL Linked Server

- ODBC: Significant performance improvement while dealing with “view” using Enterprise Manager.
- OLEDB: Better support for TEXT, BYTE, FLOAT, DECIMAL, INTERVAL and MONEY data types.



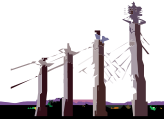
Pluggable Authentication Module (PAM) Support

- Currently supported in ESQL/C, ODBC and JDBC
- Issues around password authentication with PAM has been addressed
- DeveloperWork Articles
 - JDBC: <http://www.ibm.com/developerworks/db2/library/techarticle/dm-0704anbalagan/>
 - ESQL/C: <http://www.ibm.com/developerworks/db2/zones/informix/library/techarticle/0306mathur/0306mathur.html>

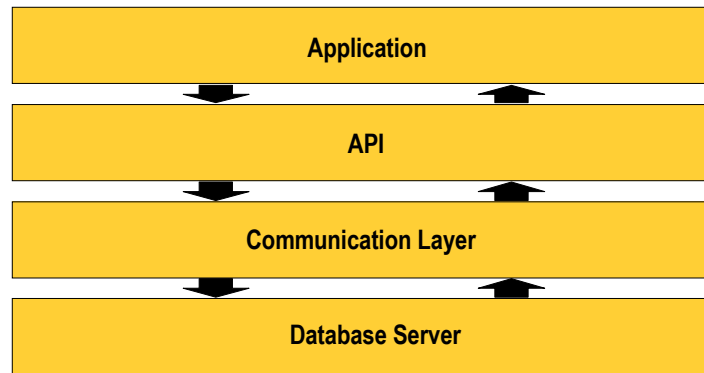


API / Application performance tips

- What is performance?
 - Speed of the Application
 - Speed of the Hardware
 - Speed of the Network
 - Type of Application
 - Human Perception
- **Performance is relative!**

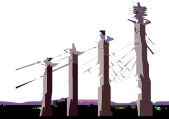


What is Performance?



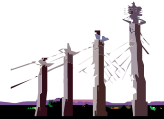
Optimization Targets

- Four main areas to focus optimization efforts
 1. Database server
 2. OS & hardware-level communications
 3. Application
 4. API



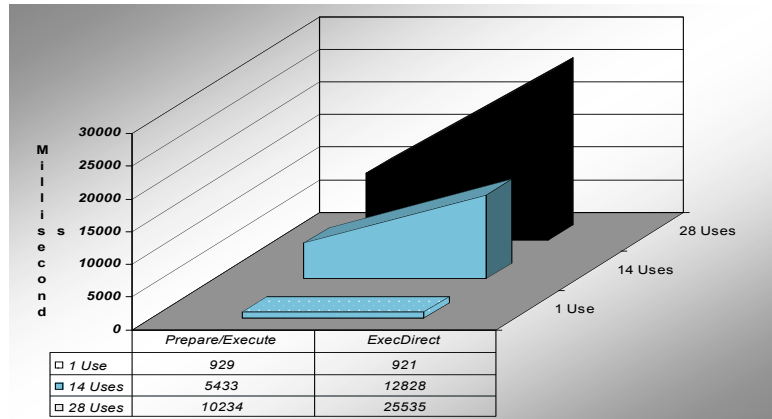
Application Optimizations

- Structure the code so that the amount of information transmitted between the client and the server is minimized. Some examples:
 1. Use joins instead of nested-loop queries
 2. Perform aggregate functions in the query
 3. Prepare and reuse queries
 4. Reuse API objects rather than allocating new ones: statements and especially connections
 5. Consider using stored procedures



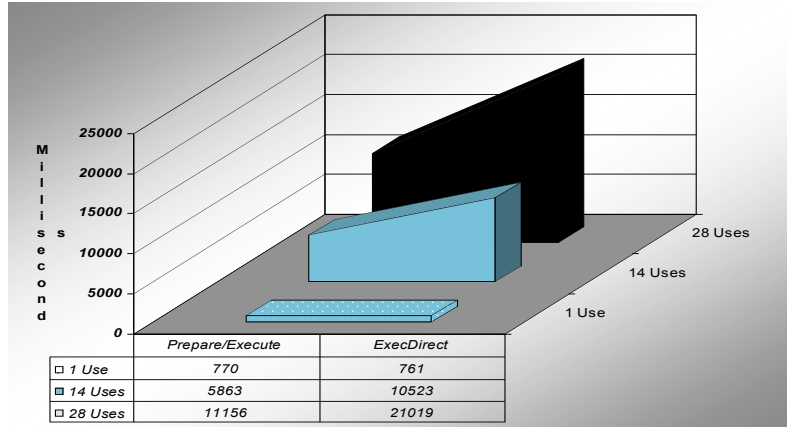
Reusing Prepared Queries Example

- select fname, lname from customer where customer_num = ?

www.iiug.org

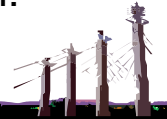
Stored Procedure Example

- select fname, lname from customer where customer_num = ?

www.iiug.org

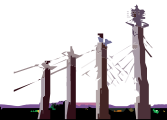
Principle API Functions

- Data type conversion
- Code set conversion
- Internal connection management
- Internal statement management
- Data buffer management
- Message handling
- All are candidates for optimization!



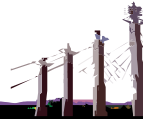
Data Type Conversion

- The most commonly used APIs allow you to select data of one type into a variable of another type - avoid overusing this ability
- Keep database and application types as similar as possible - refer to API datatype mapping tables



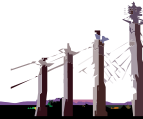
Data Conversion Example

- Required whenever datatypes differ between the application and the database
- Select customer_num from customer (100,000 rows)
- Variable binding - customer_num
 1. To character - 6500 msec
 2. To integer - 5640 msec
- Not something to worry about most of the time, but something to consider if developing a new application that processes a lot of data.



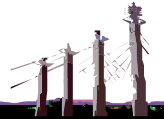
Code Set Conversion

- Required whenever CLIENT_LOCALE & DB_LOCALE are different
- Create the database using the same locale as the majority of the clients!



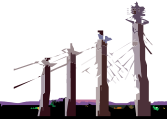
Connection Management

- Connections are resources; manage them like other resources
- Internal structures are allocated and freed
- Closing and reopening is less expensive than freeing and reinitializing
- Some APIs can optimize around the connect-disconnect cycle (connection pooling)



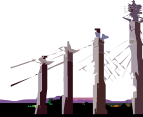
Statement Handling

- Statements are also resources; same rules apply
- Internal structures are allocated and freed
- Closing and reopening is less expensive than freeing and reinitializing



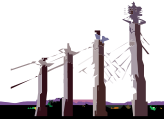
Message Handling

- SQLI messages carry information to and from the database server
- Informix products communicate using half duplex
 - Client sends message to server
 - Server processes message
 - Server sends response
 - Client sends next message
- Inherent delay in network and message processing adds to cost for each message
- Reduce the round trips by optimizing message transfers in your API



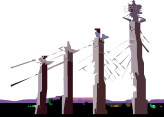
Message Optimizations

- Data Buffer Management
 - Fetch Array Size
 - Fetch Buffer Size
- Optimize Open-Fetch-Close (OPTOFC)
- Deferred Prepare
- **Percentage of effect goes up as network speed goes down**



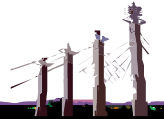
Data Buffer Management

- **Fetch Array Size**
 1. Is set programmatically
 2. Has query level scope
 3. Tells the API how many rows the application expects to receive on one fetch call and thereby guides it on how to set the fetch buffer size
- **Fetch Buffer Size**
 1. Can be set programmatically or in the environment
 2. Has connection level scope
 3. Tells the API how big to set the fetch buffer and this impacts how many rows are retrieved on a single internal fetch call



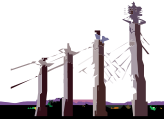
Fetch Array Details

- Effect goes up with setting value - has a ceiling at the number of descriptors that will fit in a fetch buffer
- Effect goes down as average blob size goes up
- Example:
 - Selecting 8103 text values of 255 characters each
 - One row at a time – 4914 msec
 - Fetch array = 585 (32K / 56 bytes) – 754 msec
- If there aren't simple large objects in the result set, the effect of setting fetch array to N is nearly identical to setting fetch buffer to $N * \text{tuplesize}$



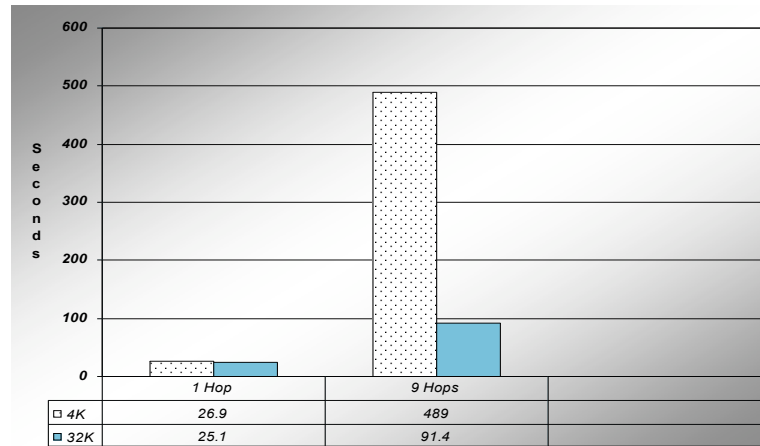
Fetch Buffer Details

- Benefits go up with setting value - has a ceiling of 32K
- Benefits diminish as the fetch buffer size exceeds the average result set size
- Effect is not large on fast networks



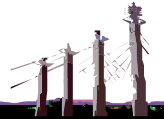
Fetch Buffer Example

- select * from customer (100,000 rows)



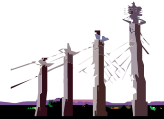
OPTOFC Details

- OPTOFC causes the open cursor message to be delayed until the first fetch call and causes the cursor to be closed automatically after the server returns the last row
- Most effective when there are a large number of queries with small result sets
- Example:
 - select fname, lname from customer where customer_num = ? (250 iterations)
 - OPTOFC = 0 - 1234 msec
 - OPTOFC = 1 - 958 msec



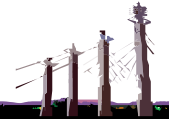
Deferred Prepare Details

- IFX_DEFERRED_PREPARE causes the prepare message to be delayed and sent with the declaration of a cursor
- Most effective when there are a large number of queries with small result sets
- Has a drawback in that if there is an error in the query, the error is not reported until sometime after the application has issued the prepare

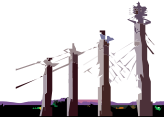
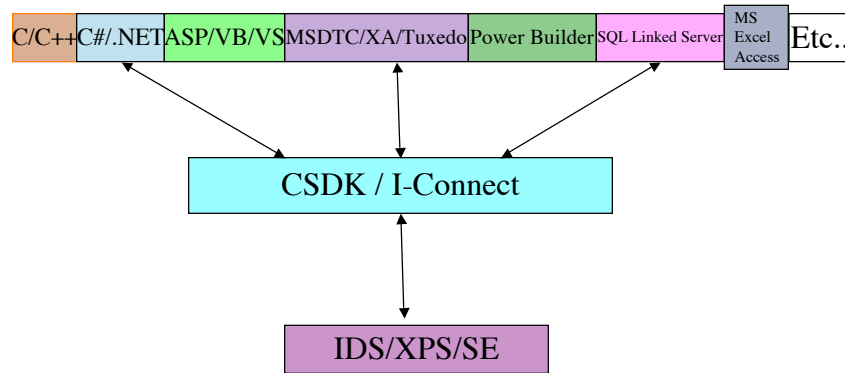


Performance Summary

- Attack the optimization issue in all four places
 - Application
 - API
 - Communication Layer
 - Database Server
- Reduce the number of client-server round trips
- Factors impacting performance are dynamic
- Performance should be periodically monitored !



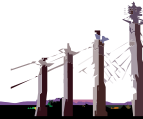
Applications Support



Other internal applications, tools, utilities -> Archecker, Blade Manager etc

Going Forward – In Future

- Support all future IDS features in CSDK
- Synchronization support in ADO.NET SQLI Provider
- We will continue to enhance our APIs for better functionalities and performance
- Optimization on the ODBC code conversion for ANSI / English data
- LOGIN TIMEOUT support in ODBC
- Reducing Stored Procedure execution round trip in ODBC/OLEDB/.NET
- Caching metadata information in ODBC
- ODBC DSN cleanup / deletion utility after uninstall
- Enhance Common ADO.NET Provider for IBM Data Servers - .NET Framework 3.5, PAM etc
- Custom install support on Unix platforms for CSDK / I-Connect
- Shipping DBACCESS with CSDK
- Integration of coledbp.sql / doledbp.sql on the server side
- Utility to tests various APIs for the verification of successful installation
- Extraction of files for various components of CSDK / I-Connect



Session #####

Session Title

Thank You

Sheshnarayan Agrawal

IBM, India

shagrawal@in.ibm.com

