IBM Informix Dynamic Server 11 New Features

2008年5月22日

Topics

- `BIGINT` and `BIGSERIAL` SQL data types
- Stored Procedure Language Enhancements
- XML Features
- Visual Explain Features
- Single Sign-On
- Secure Socket Layer
- OpenAdmin Tool for IDS

IBM Informix Dynamic Server 11.50
BIGINT and BIGSERIAL SQL data types

2008年5月22日

Overview

- New ANSI standard SQL data types **BIGINT** and **BIGSERIAL** were introduced in IDS.
- With **BIGINT** and **BIGSERIAL**, we provide a better-performance alternative to **INT8** and **SERIAL8** data types. However, IDS will continue its support of **INT8** and **SERIAL8** data types in existing customer applications.

Description

- **INT8 and SERIAL8 data types are internally implemented as a 10-byte structure, ifx_int8_t. INT8 and SERIAL8 take up to 10 bytes of storage in IDS.**
  - This feature will implement **BIGINT and BIGSERIAL** data types using native 8-byte integers. It will take 8 bytes to store these data types.
  - Like INT8, **BIGINT** will store numbers ranging from **–9,223,372,036,854,775,807** to **9,223,372,036,854,775,807** [or $-(2^{63}-1)$ to $2^{63}-1$]. The number **–9,223,372,036,854,775,808** [-2^{63}] is reserved for a NULL value and cannot be used.
  - Compared to INT8, **BIGINT** requires less storage space and, more important, should offer better performance in general because all arithmetic operations will be done using the native 8-byte integer instead of dealing with the 10-byte ifx_int8_t structure.
  - Like SERIAL8, **BIGSERIAL** will store numbers ranging from 1 to **9,223,372,036,854,775,807** [or 1 to $2^{63}-1$].

Description (cont'd)

- BIGINT **and** BIGSERIAL **will be reserved keywords.**
- BIGINT **and** BIGSERIAL **use the native 8-byte data types of the OS. On 32-bit platforms it is 'long long'. Under certain compiler options the 'long long' is not supported. During compilations of customer applications, if they detect an error they can remove the Compiler option that causes it to not define 'long long'. The alternative to this is to compile with –**DNOBIGINT**, by doing this customer may not be able to use this feature fully in their applications.**
- **A table can have only one SERIAL column, and either one** SERIAL8 **column or one** BIGSERIAL **column.**
- **The new SQL data types will be classified as *Exact Numeric Types* under Built-in Data Types and will be applicable to all DDL/DML statements wherever built-in data types are currently used.**
- **The** BIGINT **and** BIGSERIAL **SQL data types are supported in the following products:**
  - IDS and its utilities
  - ESQL/C
  - ODBC
  - JDBC
    - Common Clients and derivatives

Description

- **ALTER TABLE of** BIGINT/BIGSERIAL **columns uses the following mechanism:**

From	To BIGINT	Vice Versa
SMALLINT	**INPLACE**	**SLOW**
INTEGER	**INPLACE**	**SLOW**
INT8	**INPLACE**	**INPLACE**
CHAR(n)	**SLOW**	**C3**
DECIMAL(p1,s1)	**BIG1**	**D3**
DECIMAL(p1)	**SLOW**	**DF3**
SMALLFLOAT	**SLOW**	**FF**
FLOAT	**SLOW**	**INPLACE**
CHAR	**SLOW**	**SLOW**
BIGINT	**OMIT**	**OMIT**

  - OMIT: omitted (same data type, no conversion needed)
  - INPLACE: in place alter
  - SLOW: slow alter
  - C3, DF3, D3, IAPAFF, BIG1: use slow or in place alter depending on various factors.

Examples

```
CREATE TABLE T1 (C1 BIGSERIAL(12345), C2 BIGINT);
CREATE UNIQUE INDEX IX1 ON T1(C2);
INSERT INTO T1 VALUES (0, 1234567);

CREATE TABLE CT1 (A BIGINT , B BIGSERIAL)
      FRAGMENT BY EXPRESSION
      A <= 1000000000 IN DBSPACE2,
      A > 1000000000 AND A <= 29990000000 IN DBSPACE3,
      A > 29990000000 AND A <= 999999999999999 IN DBSPACE1;

CREATE TABLE T5 (col1 SERIAL, COL2 BIGSERIAL); -- OK
CREATE TABLE T6 (col1 SERIAL, COL2 SERIAL8);   -- OK
CREATE TABLE T7 (col1 SERIAL, COL2 SERIAL8,
                 col3 BIGSERIAL);
   -- Raises error because at most one serial8+ bigserial
   -- column can exist

ALTER TABLE T1 ADD C3 BIGINT DEFAULT 9223372036854775806;
CREATE DISTINCT TYPE BINT AS BIGINT;
```

Examples (cont'd)

```
int main( int argc, char **argv)
{
    EXEC SQL BEGIN DECLARE SECTION;
    bigint big_var = 124567;
    int    int_var = 10;
    EXEC SQL END DECLARE SECTION;

    EXEC SQL WHENEVER ERROR STOP;
    EXEC SQL database testdb;
    EXEC SQL create table big1
        (big_col BIGINT, int_col INT, bs BIGSERIAL(100));
    EXEC SQL insert into big1 values
                                (:big_var, :int_var, 0);
    EXEC SQL select big_col,  int_col
              into :big_var, :int_var from big1;
    return 0;
}
```

IBM Informix Dynamic Server 11.50
Stored Procedure Language Enhancements

2008年5月22日

Enhancements to SPL

- Overview
  - Until now only static SQL statements were possible.
  - Statements can now be dynamically constructed and executed.
  - New Dynamic SQL statement:
    - EXECUTE IMMEDIATE
- Syntax
  - `EXECUTE IMMEDIATE { SQL_quoted_string | Str_variable };`
- *`SQL_quoted_string`*: A string containing a single SQL statement
- *`Str_variable`*: A character variable containing the SQL statement

Example

```
CREATE PROCEDURE MYPROC()
       RETURNING INT;
       DEFINE A0 VARCHAR(30);
       DEFINE A1 VARCHAR(5);
       DEFINE A2 INT;
       DEFINE A3 VARCHAR(60);
       DEFINE A4 INT;
       LET A0 = "INSERT INTO DYN_TAB VALUES (";
       LET A1 = ")";
       FOR A2 = 1 TO 100
           LET A3 = A0 || A2 || A1;
           EXECUTE IMMEDIATE A3;
       END FOR;
       SELECT COUNT(DISTINCT C1) INTO A4 FROM
      DYN_TAB;
       RETURN A4;
END PROCEDURE;
-- should return 100 as 100 unique values got
-- inserted by the EXECUTE IMMEDIATE in loop
EXECUTE PROCEDURE MYPROC();
```

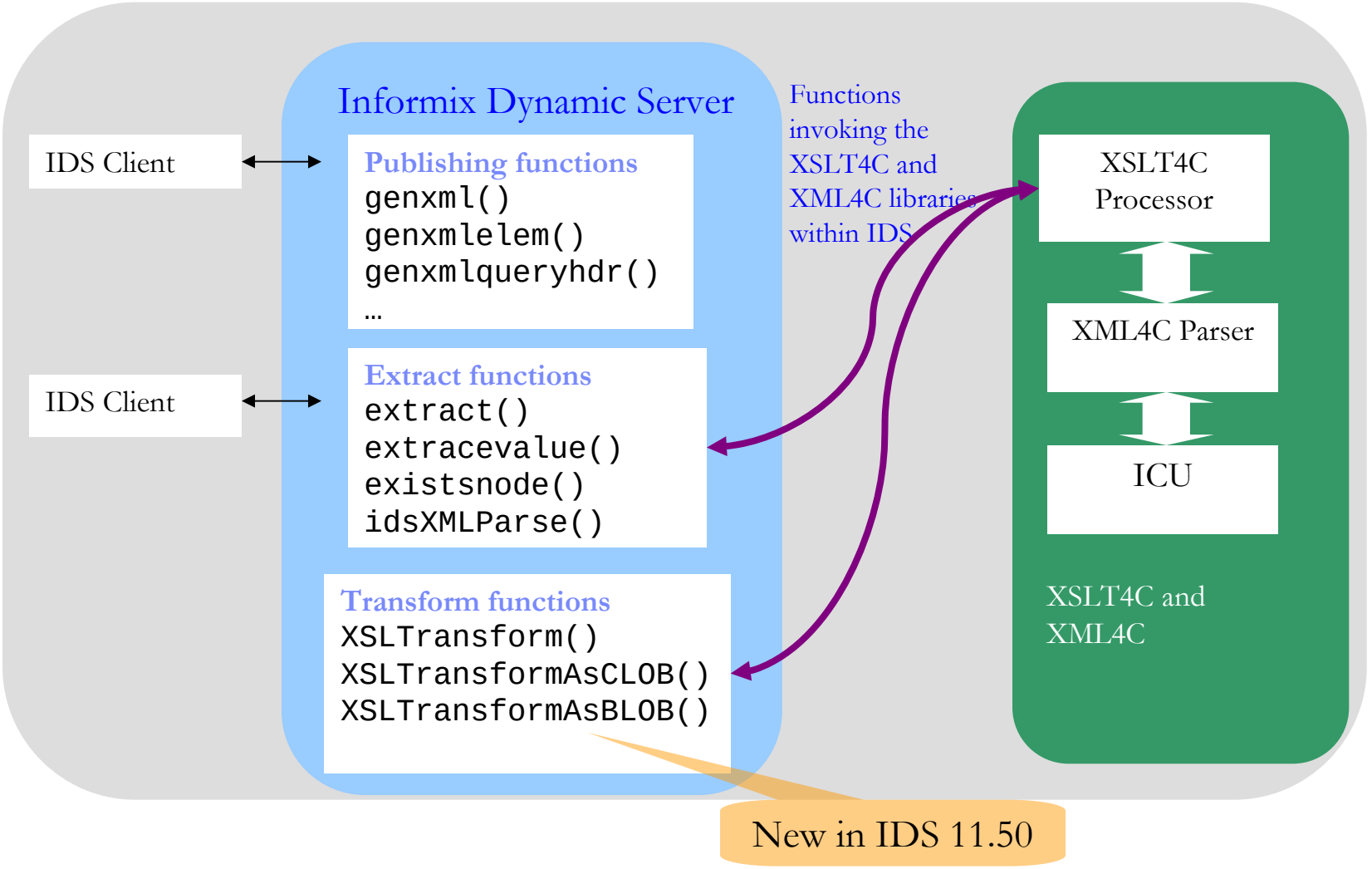
IBM Informix Dynamic Server 11.50 XML Features

2008年5月22日

Agenda

- XML in IDS 11.10
- XML in IDS 11.50
- Future Directions

What's in IDS 11?



What's in IDS 11.10?

- XML publishing functions:
  - Genxml() and GenxmlClob()
  - Genxmlqueryhdr() and GenxmlqueryhdrClob()
  - Genxmlelem() and GenxmlelemClob()

- XPath functions for pattern matching and extraction within XML document:
  - Extract() and ExtractClob()
  - ExtractValue()
  - ExistsNode()

Genxml() and GenxmlClob()

- Genxml(row_for_publishing, “doc name”) returns lvarchar(32739).
- Built-in function
- Aggregate function
- Output limited to the max size of lvarchar: 32739 bytes.
- Result set larger than that will raise an error.

```
SELECT genxml(ROW(c.cno, c.fname, c.lname, o.ordid), 'cust_order')
  FROM customer c, orders o
 WHERE c.cno = 58382 AND
       o.ord_date = '05/20/2006' AND
       c.cno = o.cno;
```

- GenxmlClob() -- Same except the return type is CLOB for larger documents.

Genxml() and GenxmlClob()

```
SELECT genxml("row", classes) FROM classes
 WHERE classid < 5;

SELECT genxmlclob("row", classes) FROM classes
 WHERE classid < 5;
```

The output for both queries (only the type is different):

```
(expression)
<row classid="1" class="125" subject="Chemistry"/>
<row classid="2" class="250" subject="Physics"/>
<row classid="3" class="375" subject="Mathematics"/>
<row classid="4" class="500" subject="Biology"/>
```

NOTE: actual output is not formatted for readability.

Genxmlelem() and GenxmlelemClob()

- Same as Genxml() except generates tags for each element separately:

```
SELECT genxmlelem(ROW(c.cno, c.fname, c.lname, o.ordid), 'cust_order')
  FROM customer c, orders o
 WHERE c.cno = 58382 AND
       o.ord_date = '05/20/2006' AND
       c.cno = o.cno;
```

Genxmlquery() and GenxmlqueryClob()

- Genxmlquery('docname', query) returns lvarchar(32760)
- GenxmlqueryClob('docname', query) returns CLOB
- Makes it easier to generate queries for static queries:
  - No need to write the ROW(…) in the projection list.
  - Only the XML document is returned.
  - Query is prepared and executed every time.
  - The element's names will be same as the column names in the projection list.

Genxmlqueryhdr() and GenxmlqueryhdrClob()

```
EXECUTE FUNCTION genxmlqueryhdr('manufact_set',
                                'SELECT * FROM manufact');

(expression)
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE manufact_set SYSTEM "../manufact_set.dtd">
<?xml-stylesheet type="text/xsl" href="../manufact_set.xsl" ?>
<manufact_set>
   <row>
      <manu_code>SMT</manu_code>
        ……   (removed for brevity)
      <manu_code>NKL</manu_code>
      <manu_name>Nikolus         </manu_name>
      <lead_time>   8</lead_time>
   </row>
   <row>
      <manu_code>PRC</manu_code>
      <manu_name>ProCycle        </manu_name>
      <lead_time>   9</lead_time>
   </row>
</manufact_set>

1 row(s) retrieved.
```

NOTE: actual output is not formatted for readability.

Extract() and ExtractValue()

```
SELECT extract(col2, '/personnel/person[3]/name/given')
  FROM tab;
SELECT extractvalue(col2,
                    '/personnel/person[3]/name/given')
  FROM tab;

EXECUTE FUNCTION extract("<person><name><fname>John
  </fname><lname>Kelly</lname></person>",
  "/person/name/lname");

SELECT col1 FROM tab
 WHERE existsnode(col2, '/personnel/person/*/email')
       = 1;
```

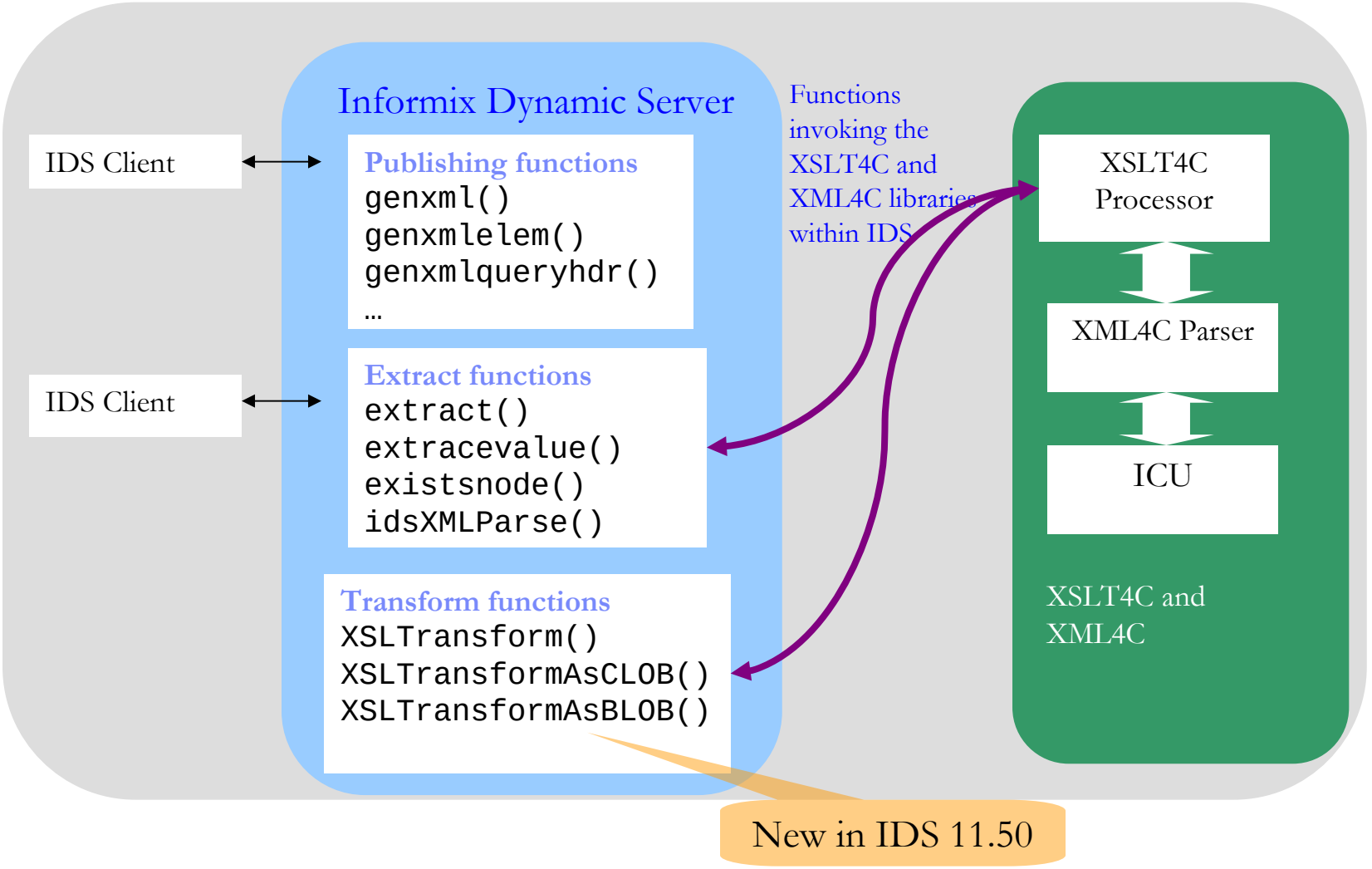
ExtractClob() and ExtractValueClob()

```
SELECT extractclob(col2,
       '/personnel/person[3]/name/given')
  FROM tab_clob_neg;

SELECT extractvalueclob(col2,
       '/personnel/person[3]/name/given')
  FROM tab_clob_neg;
```

ExistsNode()

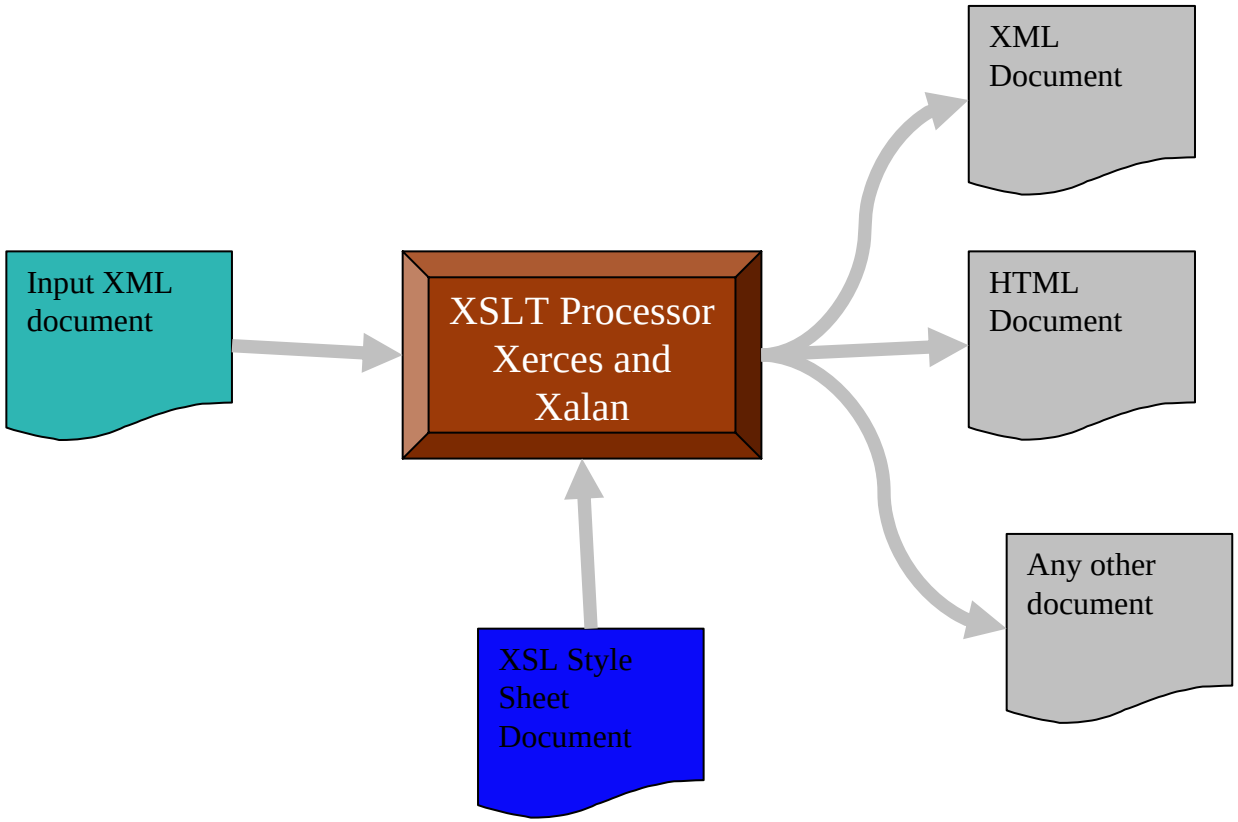
```
SELECT customer,
       extract(purchaseOrder,
                '/purchaseOrder/@orderdate')
                              AS purchase_orders
  FROM orders
 WHERE existsnode(purchaseOrder,
       '/purchaseOrder[list/item/desc/text
                           ="Shoes"]') = 1;
```

What's new in IDS 11.50?



XSLT: eXtensible Stylesheet Language Transformations

- How is it done?

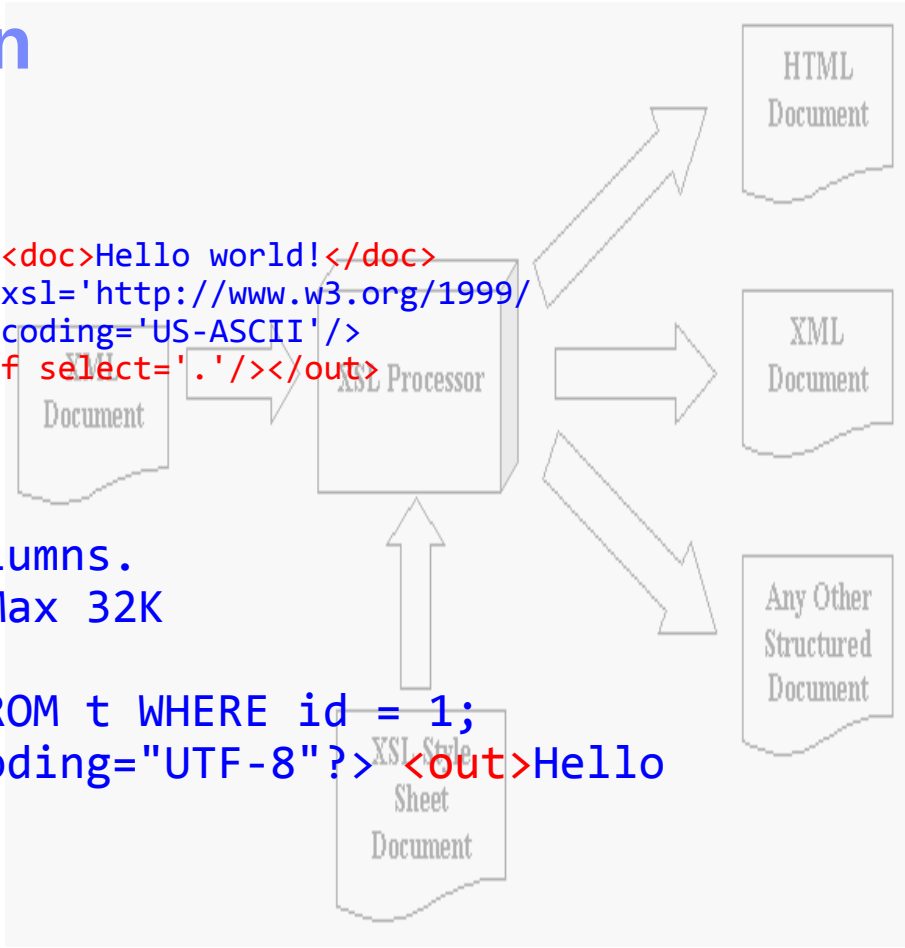


Transform functions based on XSLT: New in 11.50

- XSL and XSLT are W3C standards.
- XML is a device-independent representation of data.
- XSLT is used to transform XML documents:
  - XML to XML (transform to conform to a different schema/standard).
  - XML to HTML
  - XML to PDF
- Use XSLT to customize the display.
- XSLT enables loose integration through transformation.
- XSLT makes same data publishable to multiple targets with unique requirements.

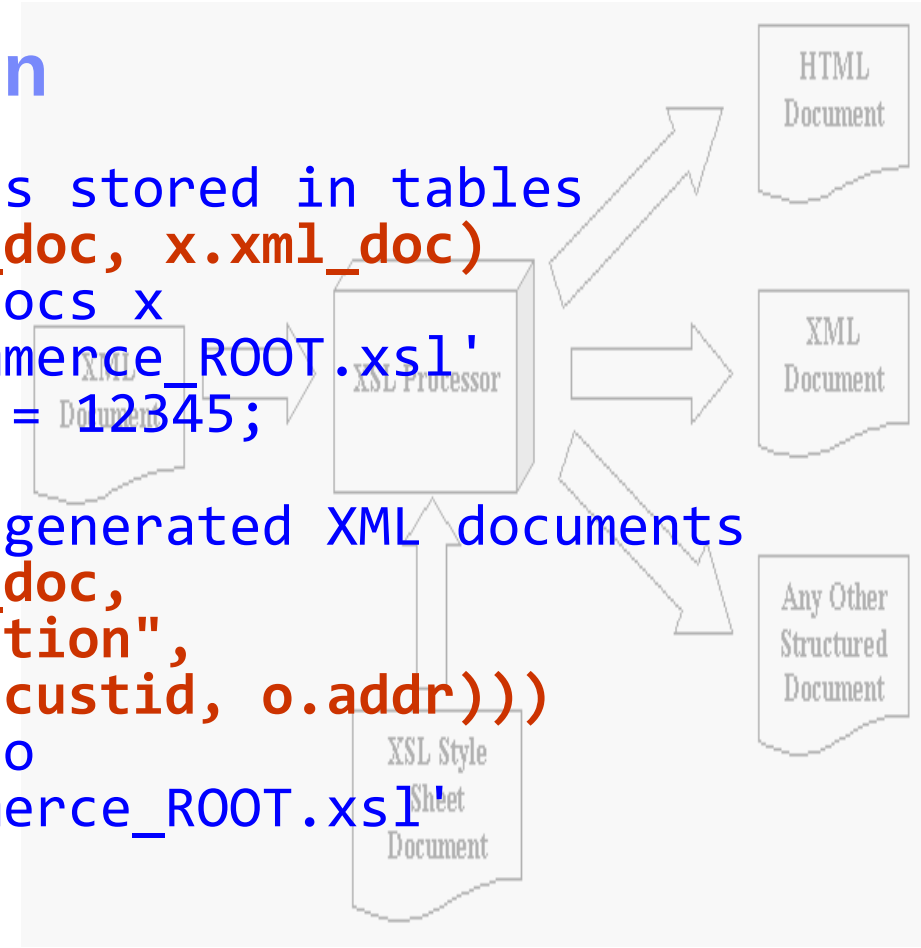
Using XSLT Transformation

```
> SELECT * FROM t WHERE id = 1;
id      1
info    <?xml version='1.0' encoding='ISO-8859-1' ?><doc>Hello world!</doc>
style   <?xml version='1.0'?> <xsl:stylesheet xmlns:xsl='http://www.w3.org/1999/
        XSL/Transform' version='1.0'> <xsl:output encoding='US-ASCII'/>
        <xsl:template match='doc'> <out><xsl:value-of select='.'/></out>
        </xsl:template> </xsl:stylesheet>
1 row(s) retrieved.

-- info and style simply lvarchar columns.
-- xsltransform() returns lvarchar. Max 32K

> SELECT xsltransform(info, style) FROM t WHERE id = 1;
(expression) <?xml version="1.0" encoding="UTF-8"?> <out>Hello
   world!</out>
1 row(s) retrieved.
```



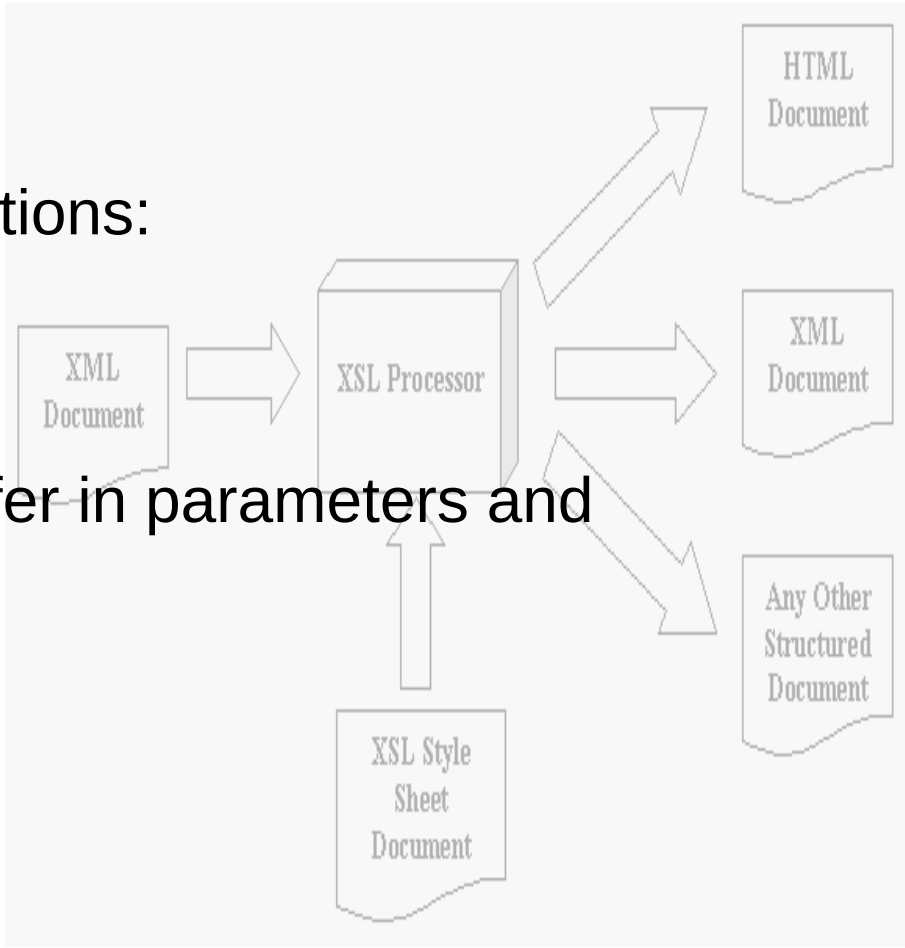
Using XSLT Transformation

```
-- Transforming XML documents stored in tables
SELECT XSLTransform(s.style_doc, x.xml_doc)
  FROM style_sheets s, xml_docs x
 WHERE s.style_title = 'ecommerce_ROOT.xsl'
   AND x.xml_transaction_id = 12345;

-- Transforming dynamically generated XML documents
SELECT XSLTransform(s.style_doc,
                    genxml("transaction",
                    row(o.trnid, o.custid, o.addr)))
FROM style_sheets s, orders o
WHERE s.style_title = 'ecommerce_ROOT.xsl'
  AND o.trnid = 12345;
```



XSLT Transform Functions

- Three overloaded XSLT functions:
  - XSLTransform()
  - XSLTransformAsClob()
  - XSLTransformAsBlob()
- The overloaded functions differ in parameters and return types.

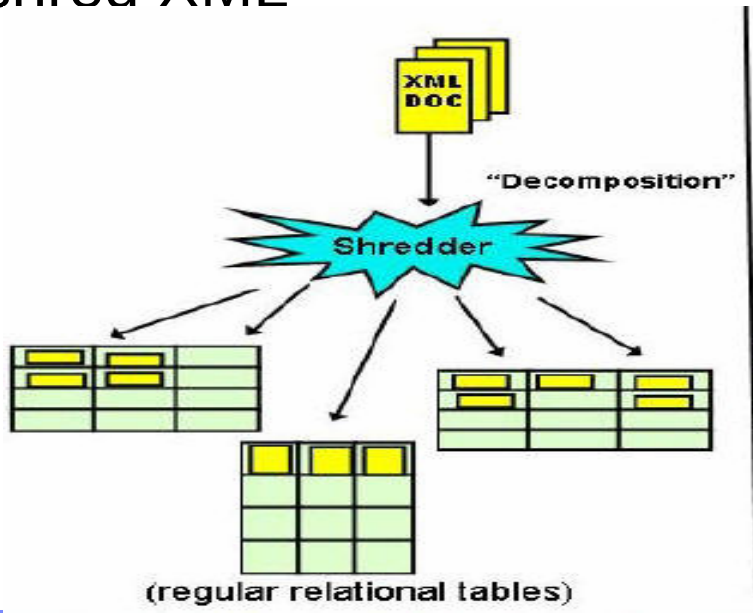


IDS – XML – Roadmap

XML Shredding – Next Release

- Shredding is the mechanism to map an XML schema into a relational schema and shred the XML document into relational tables.
- The schema has annotations indicating how XML data is entered into the database:
  - Based on actual values and serialization of input.
- Applications register a schema with IDS. Then, simply use stored procedures to shred XML documents and insert into tables.

```
xdbDecompXML
(
    rschema,
    xmlschemaname,
    xmldoc,
    documentid,
    validation_flag,
     …
)
```



Full-text index for XML

- Use when you store XML documents into columns.
- Helpful when looking for an XPath expression in the doc.
- Efficient because the XPath expression is evaluated by the index.

```
SELECT customer, address AS purchase_orders
  FROM orders
 WHERE
XMLExists(purchaseOrder,
  '/purchaseOrder[list/item/desc/text="Shoes"]');
```

XML Document

XMLExists evaluates the document for XPath expression and XML-aware index should handle it

XPath Expression

IBM Informix Dynamic Server 11.5
Visual Explain Features

2008年5月22日

Visual Explain

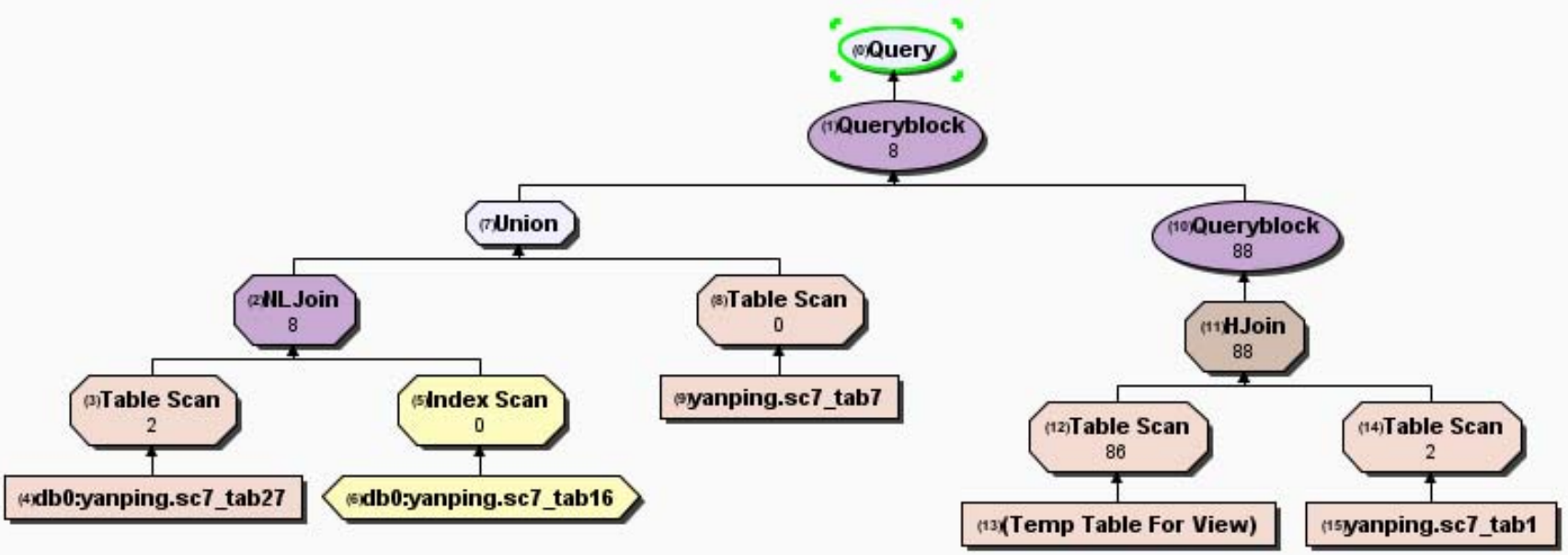
- A new function, EXPLAIN_SQL, was implemented in IDS 11.50.
- Use EXPLAIN_SQL to obtain a query explain output in XML format.
- IBM data studio can interpret the XML-formatted explain file and show the query plan graphically to the user.
- The EXPLAIN_SQL function is implemented mainly to allow common tooling such as the new IBM Data Studio to be able to get XML explain output through the function and show users the graphical query plans.
- If a user wants to obtain the XML explain output and use their own graphic tool to see the query plan, the function should be run by a JDBC or JCC program.

Sample XML Output

```
<?xml version="1.0" encoding="UTF-8"?>
<explain dbplatform="IDS" dbversion="11.11" timestamp="11-14-2007 11:50:29">
      <plans>
            <source>
                  <query>select *  from chartab where c1 = 2 ;</query>
            </source>
            <dataview id="v0" type="label">
                  <dataseq dataid="0" />
            </dataview>
            ......
            <diagram id="g0" name="Query" structure="tree">
                  <node id="n0" type="060f002b" labelviewid="l0">
                        <descriptorlink descriptorid="d0" />
                              ….
                                    <node id="n3" type="0212003a" labelviewid="l3">
                                          <descriptorlink descriptorid="d3" />
                                    </node>
                  </node>
            </diagram>
            <descriptor id="d0" name="Query" type="ids.query">
                  <datatitle nametitle="NAME">VALUE</datatitle>
            </descriptor> ……
      </plans>
</explain>
```

Data Studio Visual Explain

- Sample visual explain in Data Studio:



IBM Informix Dynamic Server 11.50
Single Sign-On

2008年5月22日

Traditional Authentication

- The user logs into the system using a username and password.
- The user has to provide username and password to authenticate to other applications.
- Complex password management and administration can result across multiple systems.

What is Single Sign-On (SSO) ?

- SSO is an authentication mechanism which allows users to enter a password once to gain access (authentication) to other resources (computers & software systems).

Single Sign-On (SSO) in IDS

- User enters the password during login.
- Users need not enter password again to authenticate to IDS.
- Authentication is invisible to the user.
- IDS uses Kerberos for SSO support.

So what does it mean?

- Easier administration:
  - The user does not have to reenter a password.
  - Password management becomes centralized and easier.

- SSO is becoming ubiquitous, in particular with more web apps. With IDS SSO support, IDS easily integrates with existing single sign-on's within the computing infrastructure.

IBM Informix Dynamic Server 11.50 Secure Socket Layer

2008年5月22日

Secure Socket Layer (SSL)

- A communications protocol that provides privacy and integrity for network data communications.
- Uses encryption to provide end-to-end secure connections.
- Presently, IDS only supports encrypted communication with SQLI clients (using Encryption Communication Support Module).
- With SSL, encrypted communication will be possible with DRDA clients too.

Digital Certificates, Certificate Authority (CA) and Keystores

- Digital Certificates are electronic ID cards issued by trusted parties known as the Certificate Authority (e.g. VeriSign).
- The SSL feature in IDS uses digital certificates to exchange keys for encryption and server authentication.
- Digital certificates are stored in a key database (also known as a keystore).
- IBM's Global Security Kit, bundled with IDS server and client, provides an iKeyman utility that can be used to create keystores and manage digital certificates.

Digital Certificates cont…

- Both the client and the server must have a keystore for housing digital certificates.
- A server-side keystore will store a digital certificate issued (or signed) by the Certificate Authority (such as VeriSign).
- A client-side keystore will store the digital certificate of the Certificate Authority (also called the root certificate or CA) that issued the server digital certificate.

Setting up SSL - SQLHOSTS

- New communication protocol:
  - drsocssl protocol for supporting SSL communication with DRDA clients.
  - onsocssl/olsocssl protocol for supporting SSL communication with SQLI clients. SSL will also be supported with server-to-server communication (distributed queries, HDR, ER, SDS/RSS).
- Example:

lenexa_on	onsoctcp	pinchy	lenexa_serv
menlo_on	**onsocssl**	pinchy	menlo_serv
portland_on	**drsocssl**	pinchy	portland_serv

Setting up SSL - ONCONFIG

- New parameters:
  - SSL_KEYSTORE_LABEL – Specifies the label of the server digital certificate in the keystore. If not configured, the server will use the default label in the keystore for SSL communications.

    Example: SSL_KEYSTORE_LABEL ids_label

- Changes to existing parameters:
  - NETTYPE – Describes the connection parameters such as the number of poll threads, max connections and class of virtual process for poll threads for connection protocols.

    NETTYPE protocol, poll threads, connections, VP class

    Specify the *protocol* as iiippp
    where: iii=[ipc|ipc|soc|tli]
    ppp=[shm|str|tcp|spx|imc|**ssl**]

  - Example: NETTYPE socssl, 3, 50, NET

Setting up SSL - ONCONFIG

- All SSL encryption/decryption operations are performed on the encrypt VP. Encrypt VPs can be configured via VPCLASS parameter:

  Example: VPCLASS encrypt, num=5

- SSL and non-SSL connection protocols can be configured for a single instance using server aliases:

  Example: DBSERVERNAME menlo_on
  DBSERVERALIASES lenexa_on, portland_on

  where menlo_on is onsocssl, lenexa_on is onsoctcp and portland_on is drsocssl connection protocol.

Setting up SSL – Keystores & Digital Certificates

- IBM's Global Security Kit (GSKit) will be installed as part of IDS and CSDK installations.
- GSKit contains iKeyman utility that can be used to create keystores and manage the digital certificates needed for SSL communication.

Setting up SSL – Keystores and Digital Certificates

- The keystore for the server is password-protected. The password is stored encrypted in the stash file (also created by iKeyman utility).
- There is one keystore per server instance. It stores the server's digital certificate and root CA certificates of the other servers it is connecting to (as in distributed queries, HDR, ER, SDS/RSS).
- The location and name of the server keystore and its password stash file is predefined:
  - $INFORMIXDIR/ssl/<servername>.kdb
  - $INFORMIXDIR/ssl/<servername>.sth

  The ownership/permissions of the above files must be informix:informix/600.
  - <servername> is the value of DBSERVERNAME onconfig parameter.

Setting up SSL – Keystores and Digital Certificates

- The password is optional for the client keystore.
- The client keystore stores root CA certificates of all servers the client is connecting to. SQLI and DRDA clients can share same keystore.
- The location and name of the client keystore and its password stash file can be configured via a new configuration file:
  - $INFORMIXDIR/etc/conssl.cfg

  New client configuration parameters:
  - SSL_KEYSTORE_FILE – Specifies the fully qualified filename of the client keystore.
  - SSL_KEYSTORE_STH – Specifies the fully qualified filename of the client stash file.
- If the file conssl.cfg does not exist or if any of above parameters are not configured, the keystore and stash file will default to:
  - $INFORMIXDIR/etc/client.kdb and $INFORMIXDIR/etc/client.sth

Recap of the SSL Setup

- Sqlhosts for client and server:
  - onsocssl/drsocssl
- Onconfig for server:
  - SSL_KEYSTORE_LABEL
  - NETTYPE for socssl
  - VPCLASS for encrypt VP
- Conssl.cfg for client:
  - SSL_KEYSTORE_FILE
  - SSL_KEYSTORE_STH
- Keystores and digital certificates for client and server.
- Initialize the server and all communications between the client and server or between servers on the onsocssl/drsocssl port will be encrypted using SSL protocol.

IBM Informix Dynamic Server 11.50
OpenAdmin Tool for IDS

2008年5月22日

OpenAdmin Tool Installation

- OAT prerequisites:
  - A web server
  - PHP 5.2.4
    - Complete with PDO, PDO_SQLITE, GD and SOAP-enabled
    - PDO_INFORMIX module
  - CSDK/I-Connect
- OAT is platform-independent:
  - OAT will run on any platform upon which the prerequisites are installed.
- However, installing and configuring the prerequisites is not easy, except on Windows and Linux.

OAT Installation Instructions

- OAT Readme:
  - Provides generic installation instructions.
  - NOT tied to any specific platform or web server.
  - Assumes the web server and PHP have already been installed.
  - Instructs on how to then configure PHP to run with OAT.
  - Links to more detailed instructions for Windows and Linux ONLY.
- Windows and Linux:
  - Users without web server or PHP knowledge:
    - install OAT on Windows or Linux because there is an open source package available that makes installation easy.
  - Download XAMPP:
    - Open source package that combines Apache + PHP.
  - Very easy to install!
    - Detailed, step-by-step articles for each platform takes you through the install steps.
  - XAMPP eases install problems, but is only available with the correct PHP version for Windows and Linux platforms.

IDSD: New Connection Daemon

- Used by OAT for MACH cluster administration.
- Allows users to start or stop an IDS server from OAT and to create new SDS secondaries:
  - Runs on the same host as the IDS server.
  - Started by inetd or xinetd.
  - Has a simple configuration.
  - Included with the IDS distribution.
- IDSD setup instructions in the OAT Readme:
  - Add services entry in the xinetd configuration file `/etc/xinetd.conf`.
  - Add a service named “idsd” in `/etc/services`.
  - Restart the xinetd service.

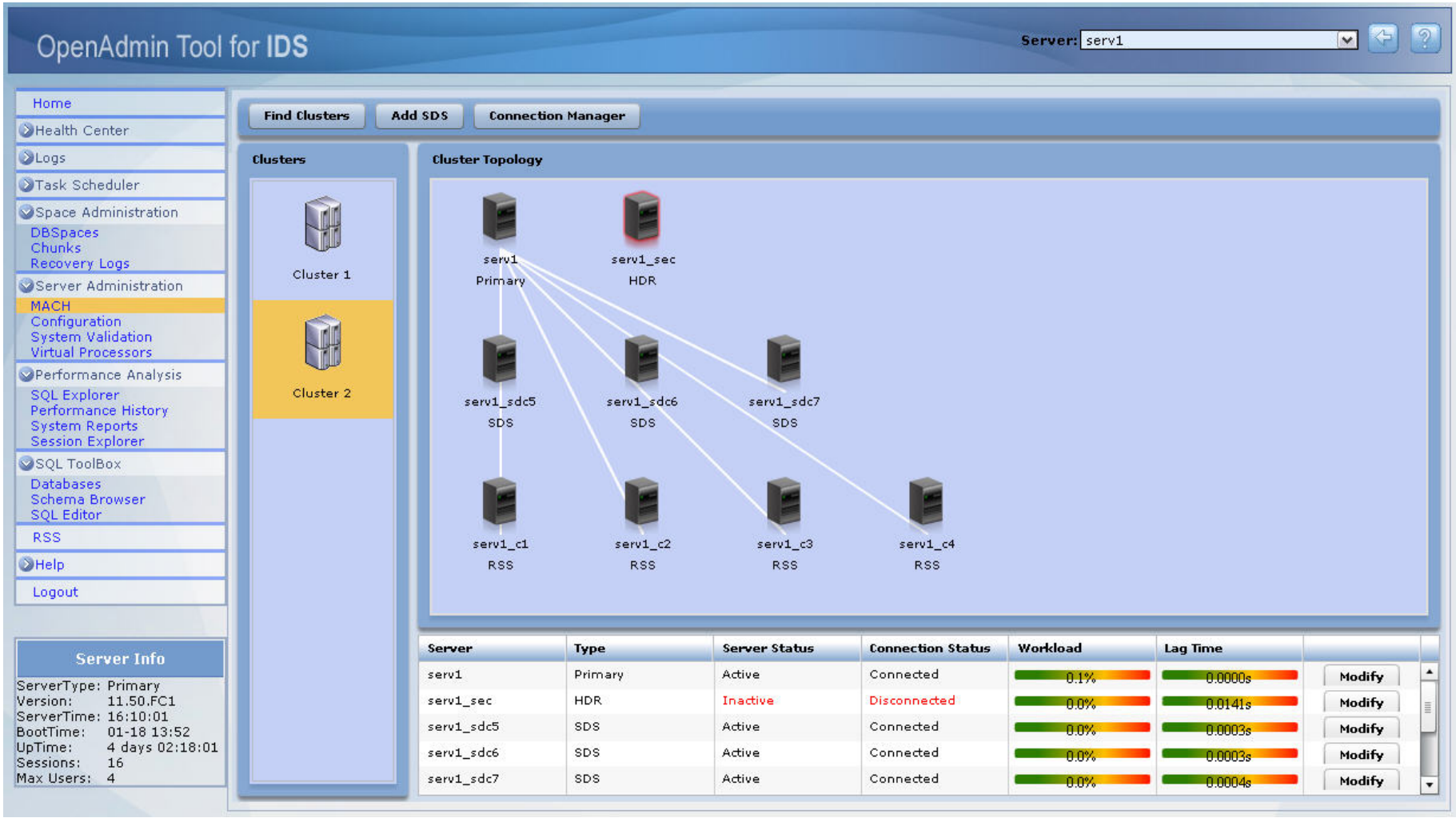
New OAT Features

- Managing high-availability clusters (MACH):
  - Configuring the IDSD daemon.
  - Starting and stopping servers in the cluster.
  - Managing service level agreements and failover configurations with the Connection Manager wizard.
- Server administration features:
  - Server configuration.
  - Historical data graphs.
  - Read-only access.
  - Integrity checks.
  - Server switch.
  - VP administration.

New OAT Features (cont'd)

- New User Interface.
- Managing High-availability clusters (MACH):
  - Creating an SDS server.
- Server administration features:
  - Automated UPDATE STATISTICS.
  - Task scheduler wizard.
  - Privileges manager.
  - Connection environment variable support.

MACH Administration



Support for OAT

- OAT is an open-source product that is NOT supported via normal tech support channels.
- OAT forum on IIUG website (oat@iiug.org) is the support mechanism.

- To Download:
  https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?lang=en_US&source=swg-informixfpd .

- Website:
  http://www.openadmintool.org

IDS 11 - Open Admin Tool Improvements – Auto Update Statistics Wizard

OAT – Update Statistics Wizard – Agenda

- Components
- Purpose
- Policies, Practices & Guidelines

What is Auto Update Statistics (AUS)?

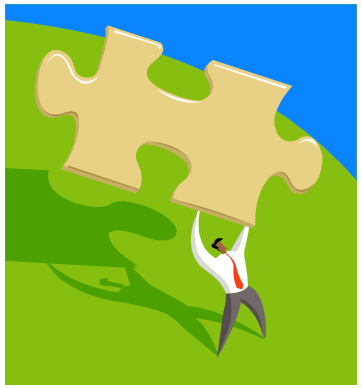
- The ability to automate the maintenance of optimizer statistics.
- It accomplishes this by:
  - Identifying tables based on a defined set of policies which require new or updated optimizer statistics.
  - Runs update statistics on the required tables in a priority order within a defined window of time.

Why Auto-Update Statistics (AUS)?

- To simplify the repetitive work DBA's are required to undertake to ensure optimal performance.
- Update statistics provides the information the query optimizer and database engine need to run queries quickly and efficiently:
  - Many customers new to Informix do not realize they must run update statistics and see poor performance due to the lack of optimizer statistics.
- The effort it takes a DBA to develop an optimal plan to run update statistics is complex and time consuming:
  - DBA's settle for a non-optimal plan or lack the understanding to develop an optimal plan.
  - Each customer must re-invent their own plan and procedures.

OAT – Auto Update Statistics (AUS)

- Two pieces comprise the AUS solutions:
  - A set of procedures which will be invoked by the database scheduler to automate update statistics
    - Installed as part of the server.
    - Users can write SQL statements to manipulate the AUS policies.
    - Allows for partners to embed programmatic solutions.
  - A OAT graphical interface allowing easier control of the different AUS policies
    - Open source download available from iiug.org and IBM websites.
    - Display AUS information in easy to read format.
    - Simple point and click interface.



AUS Database Tasks

- AUS has two database tasks in the database scheduler:
  - Auto Update Statistics Evaluation
    - Analyzes all the tables in all logged databases.
    - Locates tables which exceed the predefined polices.
    - Builds the update statistics commands.
    - Inserts the commands into the table aus_cmd_list.
  - Auto Update Statistics Refresh
    - Execute the update statistics commands in the aus_cmd_list table.
    - After the command completes, it is moved to the aus_cmd_comp table.

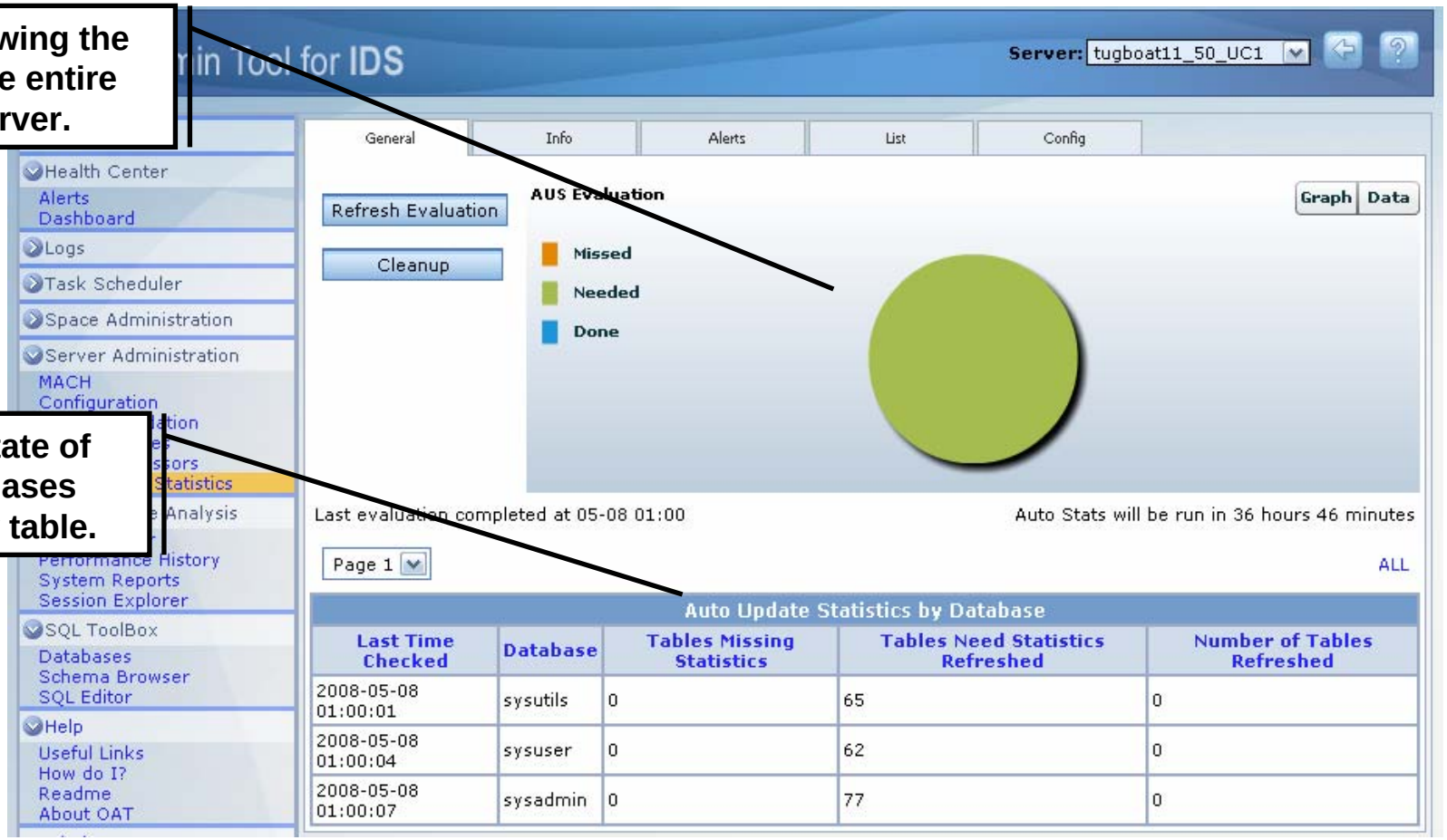
AUS also utilizes the information produced by the mon_table_profile task in the database scheduler to determine how much the table has changed. Do not disable this task!!

OAT's Graphical Interface for AUS

- General overview of the state of Statistics.

Graph showing the state of the entire data server.

Show the state of each databases statistics by table.

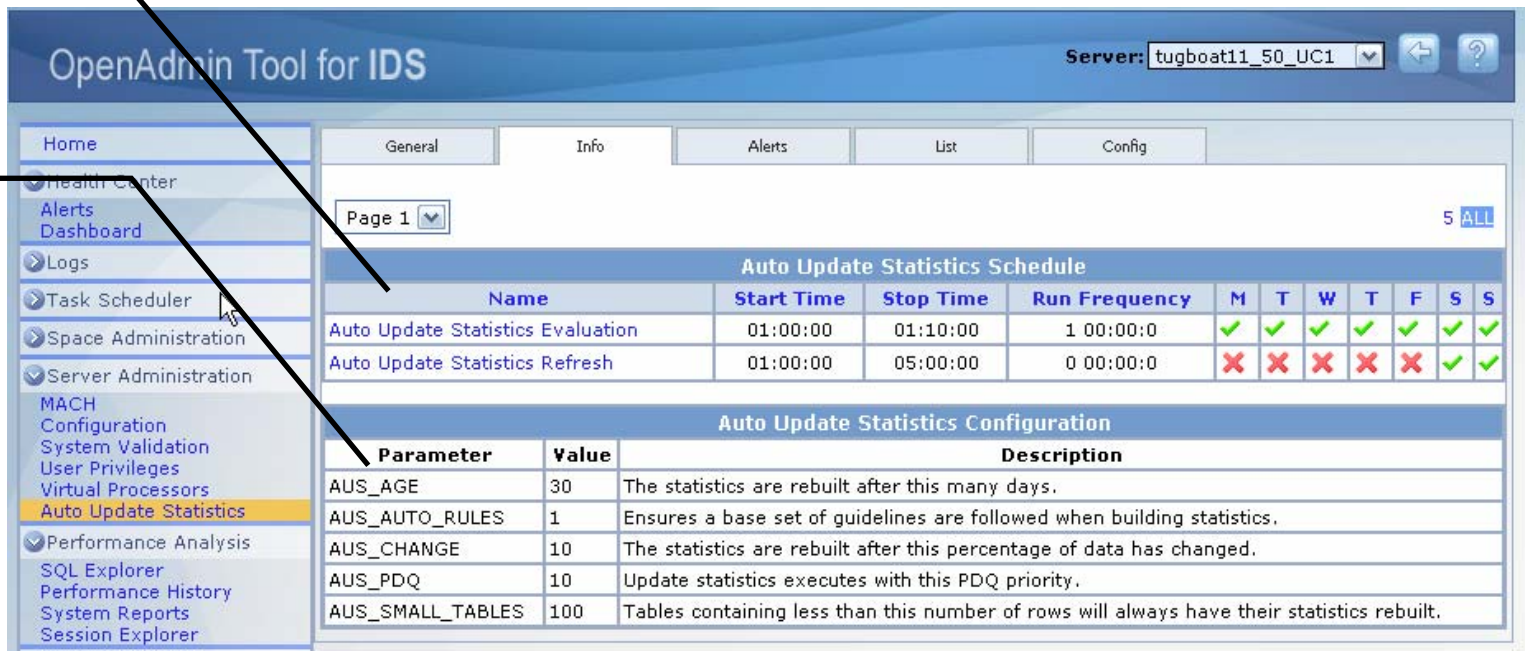


OAT's Graphical Interface for AUS

- Overview of AUS's schedule and policy settings.

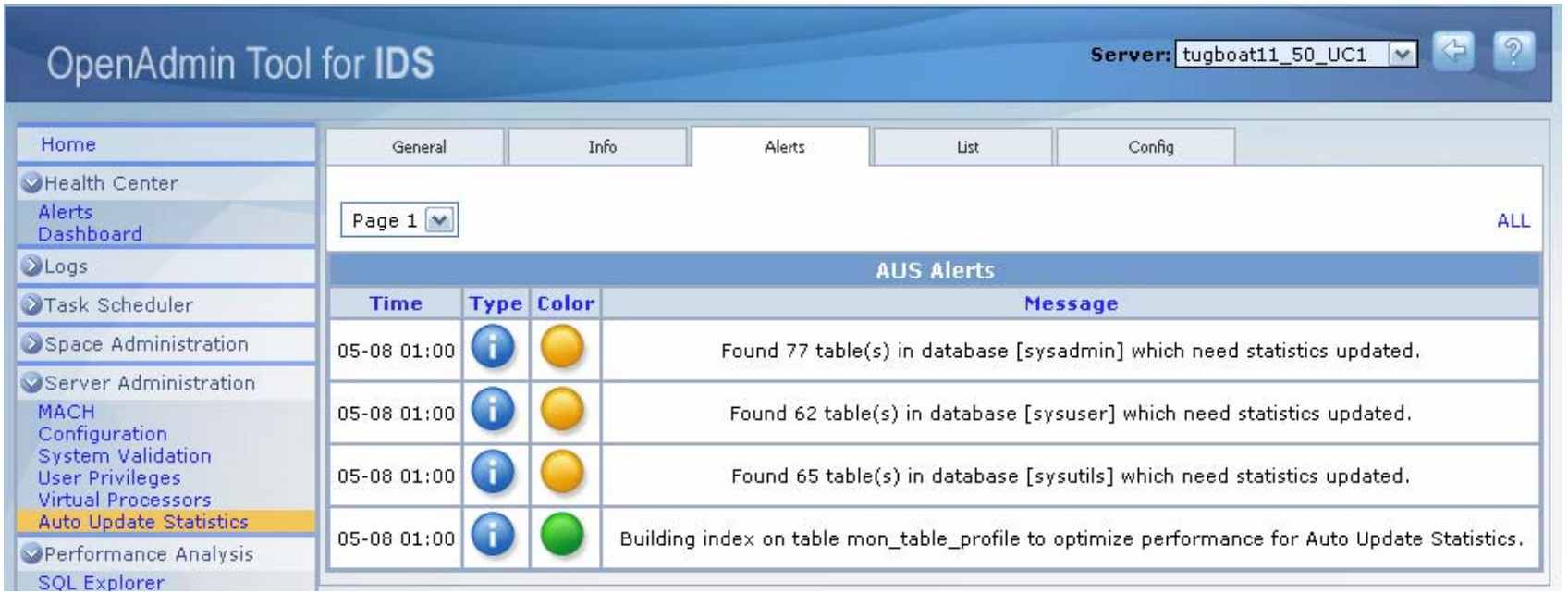
AUS Schedule

AUS Policies



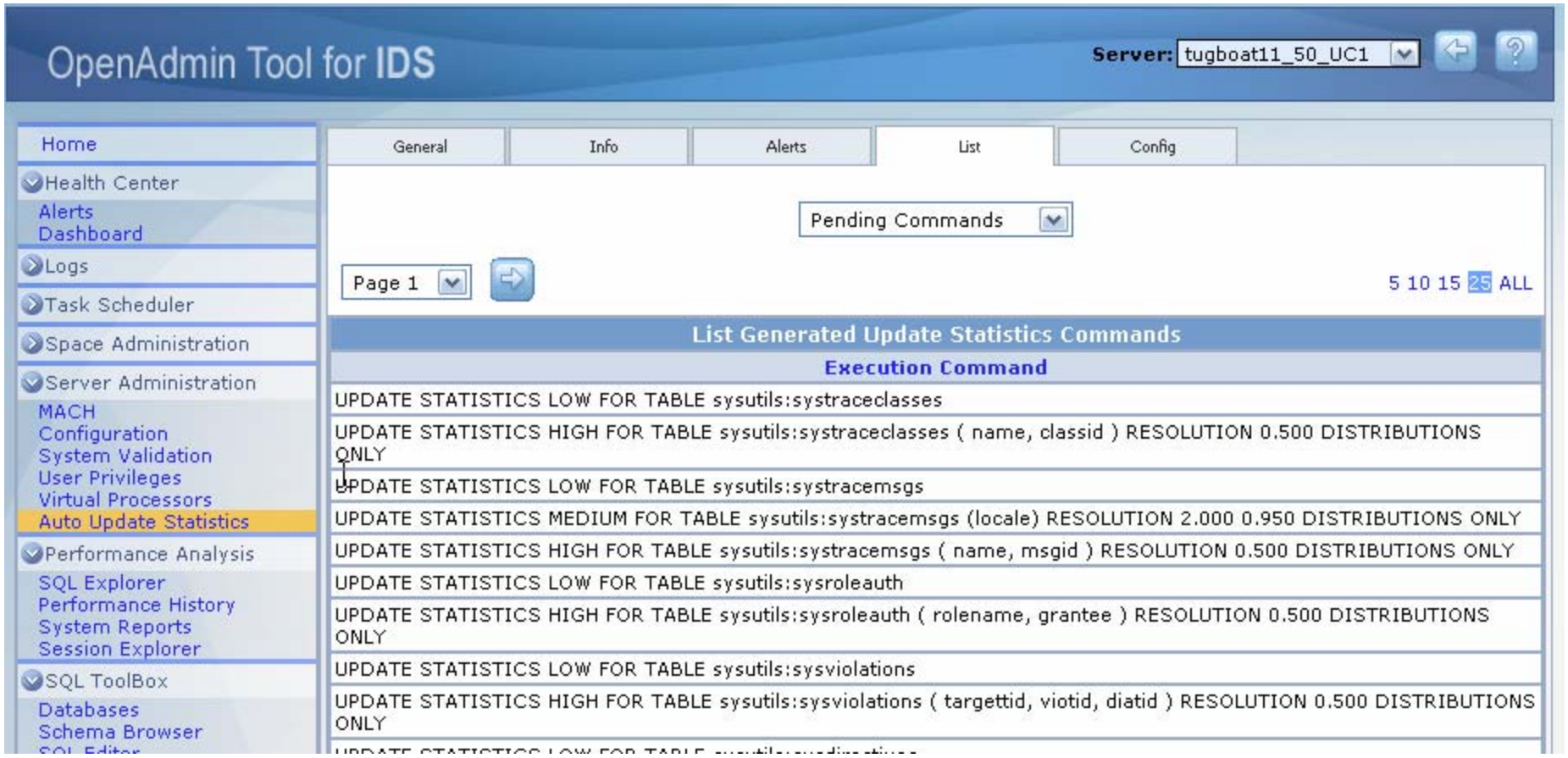
OAT's Graphical Interface for AUS

- The AUS evaluator will post alerts about databases which require new or refreshed statistics.



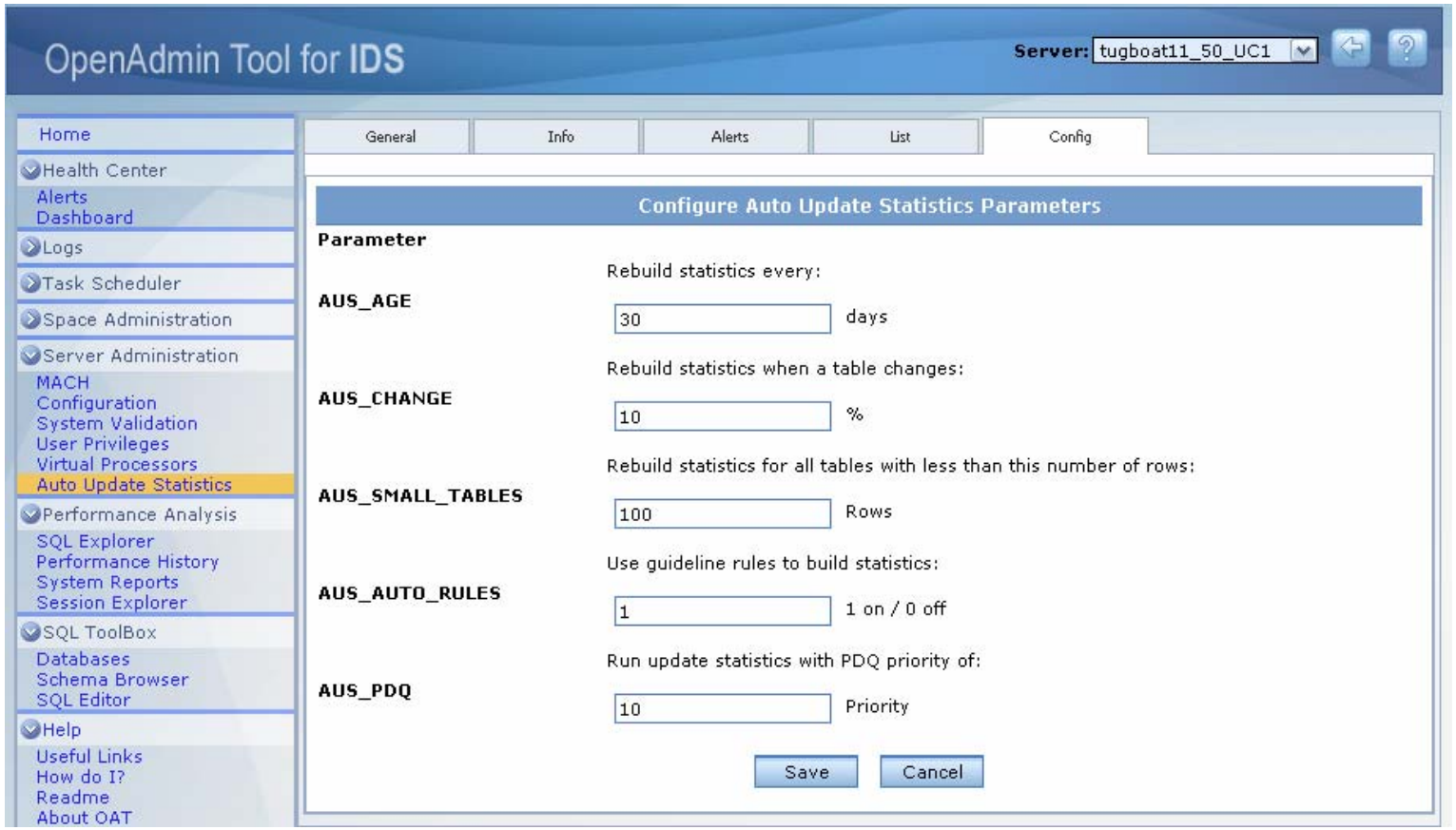
OAT's Graphical Interface for AUS

- The list of the Update Statistics commands created by the AUS Evaluator Task.



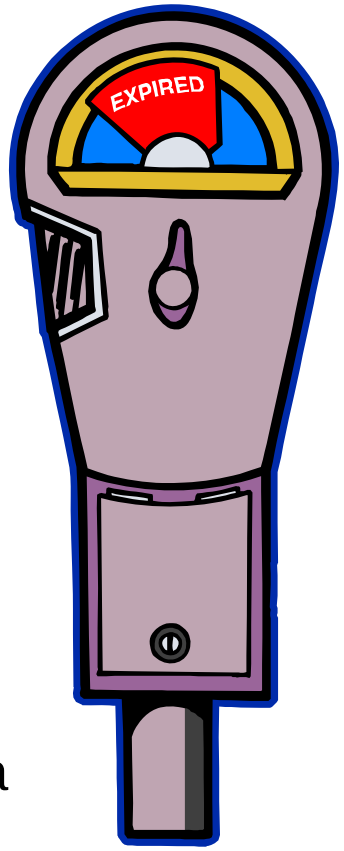
OAT's Graphical Interface for AUS

- Setting the configuration parameter of AUS.



Auto Update Statistics Policies

- Expiration policies:
  - A set of thresholds and rules which apply to different attributes of the optimizer statistics.
  - These policies are aimed at determining which tables have expired statistics or distributions.



- Two kinds of Expiration Policies:
  - Time Based
    - When the table statistics or distributions exceed a specified age, they are considered expired.
  - Modification Based
    - When the number of inserts, updates or deletes to a table exceeds a specific percentage, the statistics and distributions will be considered expired.

Implementation in 11.5

- Automates the monitoring and refreshing of statistics and distributions.
- A configurable mode is provided to assure that a current minimum level of statistics is maintained on the system. If a minimally required set of statistics or distributions are missing, they will be added to the system.
- All work performed in the sysadmin database with ph_task and ph_threshold tables being the primary repositories.

Configuration Parameters

- Configuration Parameters can be updated two ways:
  - Modified in OAT's AUS configuration page.
  - Directly by updating the table sysadmin:ph_threshold.

Ph_Threshold Table Parameter	Default Value	Description
AUS_AGE	**30 days**	**How old the statistics or distributions can be before they will be updated; even if there are no changes.**
AUS_CHANGE	**10**	**If the table has changed by more than this percentage then the statistics an distributions will be updated.**
AUS_AUTO_RULES	**1**	**Turning this on will ensure statistics and distributions are updated to a Informix minimum suggested guidelines. If the current statistics or distributions exceed the minimum suggested guidelines then the current setting will be used.**
AUS_SMALL_TABLES	**100**	**Tables containing less than this number of rows will always have their statistics rebuilt.**
AUS_PDQ	**10**	**Auto Update statistics executes with this PDQ priority.**

Minimum Auto Update Statistics Guidelines

- Low on all indexes (or on the table if no indexes exist).
- High on all lead keys of indexes.
- Medium on all non-lead keys:
  - The minimum resolution used for Medium is 2.0
  - The minimum confidence used for Medium is 0.95
  - The minimum resolution used for High is 0.5

- When AUTO_RULES is set to 1, for each table, the suggested update statistics guidelines will be followed. In the case of refreshing a user created distribution, we will never downgrade its level; as an example, we suggest medium mode distributions on the secondary columns of an index. If a user had created the distribution with high mode, high will be used. This is true not only for the level of distributions, but also the resolution, confidence and sampling size.

- Statistics for tables with rows less than AUS_SMALL_TABLES parameter will be unconditionally refreshed.

Scheduling Information

- **Run time window**

- Since AUS is a resource intensive operation, a specific run window can be defined. The run window definition includes the start and end times, and the days of the week to run the job.
- Configured from the Open Admin Tool.
- It is implemented by setting the start and end times of the aus_refresh_stats task in the ph_task table located in the sysadmin database.
- Default runtime window is 1:00 AM – 5:00 AM daily.

IDS 11 - Open Admin Tool Improvements – Auto Update Statistics Wizard - Appendix

aus_cmd_list

- **This table provides a list of the update statistics command which need to be executed.**

aus_cmd_id	**serial**	**Primary key**
aus_cmd_type	**char(1)**	**L Low, M Medium, H High**
aus_cmd_priority	**integer**	**A priority to give to the table**
aus_cmd_exe	**lvarchar**	**The command to be executed**
aus_cmd_dbs_partnum	**integer**	**Database partnum in which the table resides**
aus_cmd_partnum	**integer**	**Partnum of the table on which statistics is being executed.**

aus_cmd_info

- **This table provides a summary of the evaluators finding by database.**

aus_ci_dbs_partnum	integer	Database partnum.
aus_ci_stime	datetime year to second	Start time for database refresh statistics.
aus_ci_etime	datetime year to second	End time for database refresh statistics.
aus_ci_database	varchar	Database name.
aus_ci_missed_tables	integer	Number of tables missing statistics (Used when AUS_AUTO_RULES is 0 or disabled).
aus_ci_need_tables	integer	Number of tables needing statistics refresh.
aus_ci_done_tables	integer	Number of tables statistics completed yet in this run.

aus_cmd_comp

- **This table provides a list of update statistics commands which have been completed.**

aus_cmd_id	integer	Foreign key from aus_cmd_list.
aus_cmd_type	char	L Low, M Medium, H High.
aus_cmd_priority	integer	A priority to give to the table.
aus_cmd_dbs_partnum	integer	Database partnum.
aus_cmd_partnum	integer	Partnum of the table on which statistics is being updated.
aus_cmd_exe	lvarchar	The command to be executed.
aus_cmd_tme	datetime year to second	Number of table statistics completed yet in this run.

aus_work_info

- A temporary work table for AUS.

aus_info_id	serial	Primary key
aus_info_db_partnum	integer	The database partition number as stored in sysmaster:sysdatabases (systables.partnum).
aus_info_tabname	varchar(128)	Table name.
aus_info_tabid	integer	Table id.
aus_info_partnum	integer	Table partition number, for fragmented tables this is the lock id.
aus_info_ustlowts	datetime year to second	Time the last update statistics low was run.
aus_info_npused	bigint	Number of pages used in the table (as stored in systables).
aus_info_nrows	bigint	Number of rows in the table (as stored in systables).
aus_info_nindexes	smallint	Number of indexes on the table.

aus_work_dist

- A temporary work table for AUS.

aus_dist_id	serial	Primary key
Aus_dist_tabid	integer	Table id
aus_dist_colno	integer	Column colno from syscolumns.
aus_dist_mode	char(1)	H mode on leading index columns M mode on secondary index columns
aus_dist_resolution	float	Resolution previous update stats was executed with.
aus_dist_condience	float	Confidence previous update stats was executed with.
aus_dist_smpsize	bigint	The medium sample size used.
aus_dist_rowssmplede	bigint	Number of rows sampled.
aus_dist_constr_time	datetime year to second	Time most update stats was captured
aus_dist_ustnrows	bigint	Number of rows when stats updated

aus_work_icols

- **A temporary work table for AUS.**

aus_icols_tabid	integer	The table ID
aus_icols_colno	integer	The column number from syscolumns
aus_icols_lkey	char(1)	Y is lead key of index N is not a lead key of an index
aus_icols_mode	char(1)	H mode on leading index columns M mode on secondary index columns
aus_icols_colname	varchar(128)	The column name.

IBM Informix Dynamic Server 11 New Features – Appendix

2008年5月22日

BIGINT/BIGSERIAL Examples

- **CREATE PROCEDURE my_spl1(op1 BIGINT, op2 BIGINT, OUT res BIGINT) RETURNING BIGINT;**

  LET res = op1 +op2
  RETURN res

  END PROCEDURE;

- **CREATE FUNCTION myudr2(inpar bigint, OUT outpar bigint) returning bigint with (handlesnulls) external name "$USERFUNCDIR/cudr.udr" language C;**

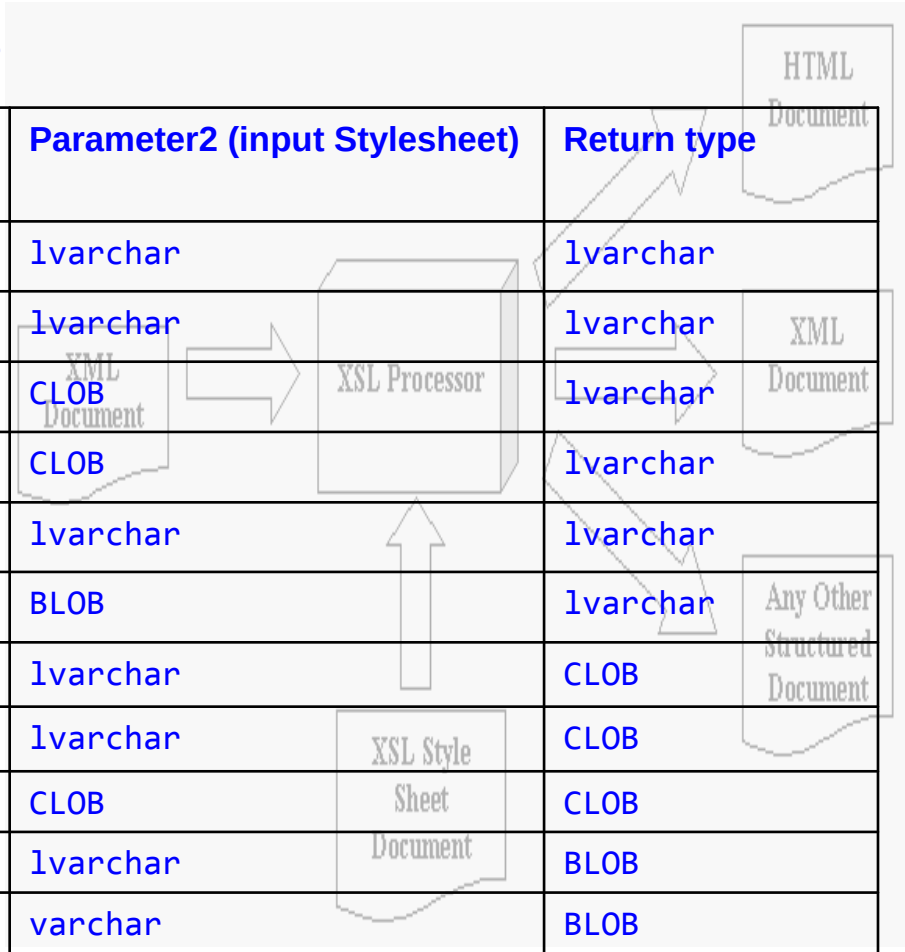
- **CREATE TABLE employee (empno bigserial, emp_name char(25), salary money(9,2));**
- **INSERT INTO employee values (1111111111111111,"Test Name 2",2000.00);**
- **select my_num, empno from employee where myudr1(111, my_num #bigint) = 1111111111111000;**

- **mi_bigint * myudr1(mi_bigint *inpar, mi_bigint *outpar, MI_FPARAM *fp)**

```
{  *outpar = 1111111111111111 - *inpar;
  mi_fp_setargisnull(fp, 1, 0);
  return(outpar);  }
```

XSLT Transform Functions

Function name	Parameter1 (input XML doc)	Parameter2 (input Stylesheet)	Return type
XSLTransform	lvarchar	lvarchar	lvarchar
XSLTransform	CLOB	lvarchar	lvarchar
XSLTransform	lvarchar	CLOB	lvarchar
XSLTransform	CLOB	CLOB	lvarchar
XSLTransform	BLOB	lvarchar	lvarchar
XSLTransform	lvarchar	BLOB	lvarchar
XSLTransformAsCLOB	lvarchar	lvarchar	CLOB
XSLTransformAsCLOB	CLOB	lvarchar	CLOB
XSLTransformAsCLOB	lvarchar	CLOB	CLOB
XSLTransformAsBLOB	lvarchar	lvarchar	BLOB
XSLTransformAsBLOB	BLOB	varchar	BLOB
XSLTransformAsBLOB	lvarchar	BLOB	BLOB



Visual Explain (con’td)

- EXPLAIN_SQL is defined in boot11.50.sql:

```
create function informix.explain_sql
    (  INOUT major_version int,
       INOUT minor_version int,
       requested_locale varchar(33),
       xml_input blob,
       xml_filter blob,
       OUT xml_output blob,
       OUT xml_message blob )
  returns blob as xml_plan
  with (HANDLESNULLS)
  external name '(explain_sql)'
  language C;

grant execute on function informix.explain_sql(int, int,
      varchar(33), blob, blob, blob, blob) to public as informix;
```

Visual Explain – XML Parameter Lists

- **What is a "plisty"?**
- **XML parameter lists.**
- All XML parameter documents adhere to a single, common parameter list Document Type Definition (DTD). This DTD is flexible enough to represent hierarchical structures and binary data.
- Property lists organize data into named values and lists of values using several core types: String, Number, Boolean, Date, Data, Array, and Dictionary.
- A dictionary (dict) contains associations of key-value pairs. A key-value pair within a dictionary is called an entry. Each entry consists of one object that represents the key and a second object that is that key's value. Within a dictionary, the keys are unique.
- An array is an ordered collection of values.

from "Common AD and admin stored procedure ..."

XML Parameter Lists (cont'd)

- **XML parameter lists DTD**

```
<!ENTITY % plistObject "(array | data | date | dict | real | integer | string | true |
   false )" >
<!ELEMENT plist %plistObject;>
<!ATTLIST plist version CDATA "1.0" >
<!-- Collections -->
<!ELEMENT array (%plistObject;)*>
<!ELEMENT dict (key, %plistObject;)*>
<!ELEMENT key (#PCDATA)>

<!--- Primitive types -->
<!ELEMENT string (#PCDATA)>
<!ELEMENT data (#PCDATA)> <!-- Contents interpreted as Base-64 encoded
   -->
<!ELEMENT date (#PCDATA)> <!-- Contents should conform to a subset of
   ISO 8601 (in particular, YYYY '-' MM '-' DD 'T' HH ':' MM ':' SS 'Z'. -->
```

XML Parameter Lists (cont'd)

- **XML parameter lists DTD (cont'd)**

```
<!-- Numerical primitives -->
<!ELEMENT true EMPTY>  <!-- Boolean constant true -->
<!ELEMENT false EMPTY> <!-- Boolean constant false -->
<!ELEMENT real (#PCDATA)> <!-- Contents should represent a floating
   point number matching ("+" | "-")? d+ ("."d*)? ("E" ("+" | "-") d+)? where d
   is a digit 0-9.  -->
<!ELEMENT integer (#PCDATA)> <!-- Contents should represent a
   (possibly signed) integer number in base 10. -->
```

XML Parameter Lists (cont'd)

- **Example file of XML parameter lists (plisty):**

```
<?xml version="1.0" encoding="UTF-8"?>
<plist version="1.0">
<dict>
<key>MAJOR_VERSION</key>
<integer>1</integer>
<key>MINOR_VERSION</key>
<integer>0</integer>
<key>REQUESTED_LOCALE</key>
<string>en_us.utf8</string>
<key>RETAIN</key>
<string>N</string>
<key>TRACE</key>
<string>Y</string>
<key>SQL_TEXT</key>
<string>select *  from tab where c1 = 2 and c2  &lt;= 100; </string>
</dict>
</plist>
```

Sample JDBC output

- Example of JDBC program running EXPLAIN_SQL

```
CallableStatement cstmt2 = conn.prepareCall("{call
informix.explain_sql(?, ?, ?, ?, ?, ?,  )}");

    /* set up the parameters */
    cstmt2.registerOutParameter( 1, Types.INTEGER );
    cstmt2.registerOutParameter( 2, Types.INTEGER );
    cstmt2.setString(3,null);
    cstmt2.setNull( 5, Types.BLOB );                    // Filter
    cstmt2.registerOutParameter( 6, Types.BLOB );       // XML_OUTPUT
    cstmt2.registerOutParameter( 7, Types.BLOB );       // XML_MESSAGE
    file = new File("./xmlins");
    fin = new FileInputStream(file);
    byte[] buffer = new byte[8000];

    IfxLobDescriptor loDesc = new IfxLobDescriptor(conn);
    IfxLocator loPtr = new IfxLocator();
    IfxSmartBlob smb = new IfxSmartBlob(conn);
    int loFd = smb.IfxLoCreate(loDesc, smb.LO_RDWR, loPtr);

    n = fin.read(buffer);
    if (n > 0) n = smb.IfxLoWrite(loFd, buffer);
    smb.IfxLoClose(loFd);
    Blob blb = new IfxBblob(loPtr);
    cstmt2.setBlob(4, blb); // set the blob column
```

Sample JDBC Output (cont'd)

- Example of JDBC program running EXPLAIN_SQL … continued

```
ResultSet rs = cstmt2.executeQuery();

int outmajver = cstmt2.getInt(1);
int outminver = cstmt2.getInt(2);

/* read the xml explain output if there is any */
while (rs.next())
{
   byte[] buf = new byte[80000];
   b = (IfxBblob) rs.getBlob(1);

   if (b != null
       {
         IfxLocator loptr = b.getLocator();
         IfxSmartBlob smbl = new IfxSmartBlob(conn);
         int lofd = smbl.IfxLoOpen(loptr, smbl.LO_RDONLY);
         outfile = new File("./out.xml");
         fout = new FileOutputStream(outfile);
         int size = smbl.IfxLoRead(lofd, fout, 80000);
         smbl.IfxLoClose(lofd);
         smbl.IfxLoRelease(loptr);
       }
}
```

Sample JDBC Output (cont'd)

- Example of JDBC program running EXPLAIN_SQL … cont'd

```
/* get blob out parameters */
       outmsg_b = (IfxBblob)cstmt2.getBlob(7);
       if (outmsg_b == null)
           System.out.println("outmsg_b is null");
      else
          {
           IfxLocator xml_msg_loptr = outmsg_b.getLocator();
           IfxSmartBlob xml_msg_smbl = new IfxSmartBlob(conn);
           int msg_out_lofd = xml_msg_smbl.IfxLoOpen(xml_msg_loptr,
                                                     xml_msg_smbl.LO_RDONLY);
           msg_out_outfile = new File("./xml_msg.xml");
           msg_out_fout = new FileOutputStream(msg_out_outfile);

           int xml_msg_size = xml_msg_smbl.IfxLoRead(msg_out_lofd,
                                                      msg_out_fout, 80000);
           xml_msg_smbl.IfxLoClose(msg_out_lofd);
           xml_msg_smbl.IfxLoRelease(xml_msg_loptr);
          }
```

Setting up SSL – Keystores and Digital Certificates

- Prerequisites for the iKeyman utility:
  - IBM JDK/JRE 1.3.1, 1.4.1 or higher with JCE PKS Security packages.

- Environment for the iKeyman utility:
  - export JAVA_HOME=<JDK/JRE installation>
  - export PATH=$JAVA_HOME/jre/bin:$PATH
  - export CLASSPATH=<GSKit installation>/classes/cfwk.zip:<GSKit installation>/classes/gsk7cls.jar:$JAVA_HOME/jre/lib/ext/ibmpkcs11.jar

Setting up SSL – Keystores and Digital Certificates

Sample commands for creating the keystore and the self-signed test certificates* using the iKeyman command line utility:

- Server Keystore:
  - gsk7cmd -keydb -create -db menlo_on.kdb -pw snoopy -type cms -stash
  - gsk7cmd -cert -create -db menlo_on.kdb -pw snoopy -label ids_label -dn "CN=menlo.ibm.com,O=ibm,C=US" -size 1024 -default_cert yes
  - gsk7cmd -cert -extract -db menlo_on.kdb -format ascii -label ids_label -pw snoopy -target ids_label.cert

    where DBSERVERNAME is menlo_on
    SSL_KEYSTORE_LABEL is ids_label

- Client Keystore:
  - gsk7cmd -keydb -create -db client.kdb -pw snoopy -type cms -stash
  - gsk7cmd -cert -add -db client.kdb -pw snoopy -label ids_label -file ids_label.cert -format ascii

* In production systems, digital certificates will be requested from the Certificate Authority. Refer to the iKeyman User Guide for more information on this.