

# 2012 IIUG Informix Conference

The largest Informix gathering in the world

## Informix SQL Using New SQL Features

Jeff Filippi

Integrated Data Consulting, LLC

Session B01 Monday April 23 – 9:30am





# Introduction

- 21 years of working with Informix products
- 17 years as an Informix DBA
- Worked for Informix for 5 years 1996 – 2001
- Certified Informix DBA
- Started my own company in 2001 specializing in Informix Database Administration consulting services
- IBM Business Partner / Reseller of Informix
- OLTP and Data warehouse systems
- Informix 4.x, 5.x., 7.x, 9.x, 10.x, 11.10, 11.50, 11.70



# Overview

- Take a look at new SQL features that have been introduced since Informix 11.10
- Show how they can be used in applications

# SQL Features – 11.10


- **Informix 11.10 New Features**

- **11.10xC1**

- Informix now supports the following new built-in SQL functions to perform common mathematical, casting, and bitmap operations, and for manipulating character string, date, and datetime values:

- |                     |                |
|---------------------|----------------|
| • ADD_MONTHS()      | ASCII()        |
| • BITAND()          | BITANDNOT()    |
| • BITNOT()          | BITOR()        |
| • BITXOR()          | CEIL()         |
| • FLOOR()           | FORMAT_UNITS() |
| • LAST_DAY()        | LTRIM()        |
| • MONTHS_BETWEEN () | NEXT_DAY ()    |
| • NULLIF()          | POWER()        |
| • ROUND()           | RTRIM()        |
| • SYSDATE()         | TO_CHAR()      |
| • TO_NUMBER()       | TRUNC()        |

These built-in SQL functions can simplify the migration to IDS of applications that have been developed for other database servers.



# SQL Features – 11.10

- **Informix 11.10 New Features**

- **11.10xC1 - ADD\_MONTHS()**

- The **ADD\_MONTHS** function takes a DATETIME or DATE expression as its first argument, and requires a second integer argument, specifying the number of months to add to the first argument value. The second argument can be positive or negative.
- The value returned is the sum of the DATE or DATETIME value of the first argument and an INTERVAL UNITS MONTH value that is based on the number of months that the second argument specifies.
- The returned data type depends on the data type of the first argument:
- If the first argument evaluates to a DATE value, **ADD\_MONTHS** returns a DATE value.
- If the first argument evaluates to a DATETIME value, **ADD\_MONTHS** returns a DATETIME YEAR TO FRACTION(5) value.
- If the *day* and *month* time units in the first argument specify the last day of the month, or if the resulting month has fewer days than the *day* in the first argument, then the returned value is the last day of the resulting month. Otherwise, the returned value has the same day of the month as the first argument.
- The returned value can be in a different year, if the resulting month is later than December (or for negative second arguments, earlier than January) of the year in the first argument.

# SQL Features – 11.10


- **Informix 11.10 New Features**

- **11.10xC1 – ADD\_MONTHS()**

- The following query calls the **ADD\_MONTHS** function twice in the Projection clause, using column expressions as arguments. Here the column names indicate the column data types, and the **DBDATE** setting is MDY4/:
- `SELECT order_delay, order_date, ADD_MONTHS(order_date, order_delay), delivery_datetime, ADD_MONTHS(delivery_datetime, order_delay)`  
`FROM orders`  
`WHERE order_num = 100;`

In this example **ADD\_MONTHS** returns DATE and DATETIME values:

- |                     |                           |
|---------------------|---------------------------|
| – order_delay       | 8                         |
| – order_date        | 08/08/2011                |
| – (expression)      | 04/08/2012                |
| – delivery_datetime | 2011-09-09 12:00:00.00000 |
| – (expression)      | 2012-05-09 12:00:00.00000 |




# SQL Features – 11.10

- **Informix 11.10 New Features**

- **11.10xC1 – ASCII**

- **ASCII** function takes a single argument of any character data type. It returns an integer value, based on the first character of the argument, corresponding to the decimal representation of the codepoint of that character within the ASCII character set.
    - If the argument is NULL, or if the argument is an empty string, the **ASCII** function returns a NULL value.





# SQL Features – 11.10


- **Informix 11.10 New Features**

- **11.10xC1 – ASCII**

- The following query returns the ASCII value of uppercase W:
      - `select ASCII(first_name) from employee  
where first_name = 'William'`
    - The following shows the output of this SELECT statement.

87






# SQL Features – 11.10

- **Informix 11.10 New Features**

- **11.10xC1 – BITWISE LOGIC FUNCTIONS**

- **BITAND(), BITANDNOT(), BITNOT(), BITOR(), BITXOR()**
  - The arguments to these functions can be any numeric data type that can be converted to the INT8 data type.
  - Except for **BITNOT**, which takes a single argument, these bitwise logical functions take two arguments that can be converted to an INT8 value.
  - If both arguments have the same integer types:
    - » The data type of the returned value is the same type as the arguments.
    - » If they are different integer types the returned value is the one with the greater precision.
  - If the arguments are any other numeric type, such as DECIMAL, SMALLFLOAT, FLOAT, or MONEY, or some combination of those types, the returned data type is DECIMAL(32).




# SQL Features – 11.10

- **Informix 11.10 New Features**

- **11.10xC1 – CEIL()**

- The **CEIL** function takes as its argument a numeric expression, or a string that can be converted to a DECIMAL data type, and returns the DECIMAL(32) representation of the smallest integer that is greater than or equal to its single argument.
  - Value of “avg\_score” (20.2) - Returns 21
    - » `SELECT CEIL(avg_score) FROM team_stats WHERE team_id = 2;`
  - Value of “avg\_score” (-20.2) - Returns -20
    - » `SELECT CEIL(avg_score) FROM team_stats WHERE team_id = 3;`



# SQL Features – 11.10

- **Informix 11.10 New Features**

- **11.10xC1 – FLOOR()**

- The FLOOR() function is like the CEIL() function, but it returns the larger integer that is smaller than or equal to the FLOOR argument:
      - Value of “avg\_score” (20.2) - Returns 20
        - » SELECT CEIL(avg\_score) FROM team\_stats  
WHERE team\_id = 2;
      - Value of “avg\_score” (-20.2) - Returns -21
        - » SELECT CEIL(avg\_score) FROM team\_stats  
WHERE team\_id = 3;




# SQL Features – 11.10

- **Informix 11.10 New Features**

- **11.10xC1 – FORMAT\_UNITS()**

- The **FORMAT\_UNITS** function can interpret strings that specify a number and the abbreviated names of units of memory or of mass storage. It can accept one, two, or three quoted string arguments.
    - The values used in the function are:
      - Bytes - 'B' or 'b'
      - Kilobytes - 'K' or 'k'
      - Megabytes - 'M' or 'm'
      - Gigabytes - 'G' or 'g'
      - Terabytes - 'T' or 't'
      - Petabytes - 'PB'
      - Pages - 'P'



# SQL Features – 11.10

- **Informix 11.10 New Features**

- **11.10xC1 – FORMAT\_UNITS()**

- **Examples**

- EXECUTE FUNCTION FORMAT\_UNITS('1024 M')

- » Returns "1.00 GB"

- SELECT FORMAT\_UNITS('1024 k') FROM systables  
WHERE tabid = 1

- » Returns "1.00 MB"

# SQL Features – 11.10

- **Informix 11.10 New Features**

- **11.10xC1 – LAST\_DAY()**

- It returns the date of the last day of the month that its argument specifies. The difference between the returned value and the argument is the number of days remaining in that month.
- The following query returns the DATE representation of the current date, the date of the last day in the current month, and the integer number of days (calculated by subtracting the first DATE value from second) before the last day in the current month:

```
– SELECT TODAY AS today, LAST_DAY(TODAY) AS last,  
        LAST_DAY(TODAY) - TODAY AS days_left  
FROM game_days  
WHERE team_id = 4;
```

- If the query were issued on 20 April 2012, with MDY4/ as the **DBDATE** setting for the default locale, it would return the following information:

today	last	days_left
04/20/2012	04/30/2012	10



# SQL Features – 11.10

- **Informix 11.10 New Features**

- **11.10xC1 – LTRIM()**

- The **LTRIM** function scans the *source\_string* from the left, deleting any leading characters that appear in the *pad\_string*. If no *pad\_string* argument is specified, only leading blanks are deleted from the returned value.
    - In the following example, the *pad\_string* is 'Blackhawks':
      - SELECT LTRIM('BlackhawksRedWings WIN!', 'Blackhawks')  
FROM teams;
        - » Returns: 'RedWings WIN!'





# SQL Features – 11.10

- **Informix 11.10 New Features**

- **11.10xC1 – MONTHS\_BETWEEN()**

- The value returned is a DECIMAL data type, representing the difference between the two arguments, expressed as a DECIMAL value in units based on 31-day months. If the first argument is a point in time later than the second argument, the sign of the returned value is positive. If the first argument is earlier than the second argument, the sign of the returned value is negative.
    - If the dates of the arguments are both the same days of a month or are both the last days of a months, the result is a whole number.
      - ```
SELECT MONTHS_BETWEEN(TO_DATE('2-2-2012', '%m-%d-%Y'),  
                      TO_DATE('9-10-2011', '%m-%d-%Y')) AS months  
FROM team_seasons  
WHERE sport = 'Football'
```
      - The value returned by the query expresses the 32-day difference between the two DATE arguments as a positive number of 31-day months:
        - » months 4.74193548387097

# SQL Features – 11.10

- **Informix 11.10 New Features**

- **11.10xC1 – NEXT\_DAY()**


- The **NEXT\_DAY** function requires a DATE or DATETIME expression as its first argument, and requires a second *weekday* argument that is a quoted string representing the abbreviation of the English name for a day of the week.

- The valid values are ('SUN', 'MON', 'TUE', 'WED', 'THU', 'FRI', 'SAT').

- SELECT game\_date, NEXT\_DAY(game\_date, 'SAT') AS next\_saturday,  
NEXT\_DAY(game\_date, 'SAT') - game\_date AS num\_days  
FROM game\_schedule;

- The result set of this query might include the following data from the **game\_schedule** table:

| game_date  | next_saturday | num_days |
|------------|---------------|----------|
| 02/22/2012 | 02/25/2012    | 3        |



# SQL Features – 11.10

- **Informix 11.10 New Features**

- **11.10xC1 – NULLIF**

- **NULLIF** evaluates its two arguments, *expr1* and *expr2*. If their values are equal. then **NULLIF** returns NULL.
    - If their values are not equal. then **NULLIF** returns *expr1*.
      - SELECT team\_name, win, NULLIF(win, 'n')  
FROM teams;
      - Values Returned
        - » ANSWER is 'n' then value returned is NULL
        - » ANSWER is 'y' then value returned is 'y'




# SQL Features – 11.10

- **Informix 11.10 New Features**

- **11.10xC1 – POWER (POW)**

- The **POW** function raises its first numeric argument, the *base*, to the power of its second numeric argument, the *exponent*. The returned value is a FLOAT data type.
    - The following example returns all rows from the **circles** table in which the **radius** column value implies an area less than 1,000 square units, using an approximation to pi with a scale of 4:
      - SELECT \* FROM circles  
WHERE (3.1416 \* POW(radius,2)) < 1000;




# SQL Features – 11.10

- **Informix 11.10 New Features**

- **11.10xC1 – ROUND()**

- The **ROUND** function can reduce the precision of its first numeric, DATE, or DATETIME argument, and returns the rounded value. If the first argument is neither a number nor a point in time, it must be cast to a numeric, DATE, or DATETIME data type.
- Positive-digit values specify rounding to the right of the decimal point; negative-digit values specify rounding to the left of the decimal point.
  - Examples of negative, zero, and positive rounding factors:
    - » `ROUND(12,420.7846,-2) = 12,400.00`
    - » `ROUND(12,420.7846,0) = 12,421.00`
    - » `ROUND(12,420.7846,2) = 12,420.78`
  - `SELECT ROUND(125.46,0), ROUND(total_price) FROM items;`
  - `SELECT ROUND(order_dt, 'YEAR') FROM orders;`




# SQL Features – 11.10

- **Informix 11.10 New Features**

- **11.10xC1 – RTRIM()**

- The **RTRIM** function removes specified trailing pad characters from a string.
- The first argument to the **RTRIM** function must be a character expression from which to delete trailing pad characters. The optional second argument is a character expression that evaluates to a string of pad characters. If no second argument is provided, only blank characters are regarded as pad characters.
  - `SELECT RTRIM('Super Bowl Victory... WON','WON') FROM football`
  - Returns “Super Bowl Victory...”



# SQL Features – 11.10

- **Informix 11.10 New Features**

- **11.10xC1 – SYSDATE()**

- The **SYSDATE** operator returns the current DATETIME value from the system clock.
    - **SYSDATE** is identical to the **CURRENT** operator, except that the default precision of **SYSDATE** is YEAR TO FRACTION(5), while the default precision of **CURRENT** is YEAR TO FRACTION(3).
    - You can use **SYSDATE** in any context where the **CURRENT** operator is valid.
    - You can use **SYSDATE** in CREATE TABLE statement and in SQL statements:
      - CREATE TABLE orders ( order\_id SERIAL, order\_name CHAR(30), order\_time DATETIME YEAR TO FRACTION(5) DEFAULT SYSDATE, );
      - INSERT INTO tab1 VALUES (0, 'Dryer', SYSDATE);
      - SELECT SYSDATE AS sysdate, order\_id FROM orders;




# SQL Features – 11.10

- **Informix 11.10 New Features**

- **11.10xC1 – TO\_CHAR()**

- The **TO\_CHAR** function converts DATETIME or DATE values to character string values.
- The **TO\_CHAR** function evaluates a DATETIME value according to the date-formatting directive that you specify and returns an NVARCHAR value.
- You can also use the TO\_CHAR function to convert a DATETIME or DATE value to an LVARCHAR value.
- The following query uses the TO\_CHAR function to convert a DATETIME value to a more readable character string.
- The symbols mean the following:
  - %A Full weekday name, as defined in the locale
  - %B Full month name, as defined in the locale
  - %d Day of the month as an integer (01 through 31). A single-digit value is preceded by a zero (0).
  - %Y Year as a 4-digit decimal number
  - %R Time in 24-hour notation (equivalent to %H:%M format, as defined below).
- ```
SELECT order_num, TO_CHAR(order_date, "%A %B %d %Y") order_date
FROM orders WHERE order_num = 130405;
```
- Results: order\_num 130405 order\_date Monday March 05 2012



# SQL Features – 11.10

- **Informix 11.10 New Features**

- **11.10xC1 – TO\_NUMBER()**


- The **TO\_NUMBER** function converts its argument to a DECIMAL data type. The argument can be the character string representation of a number or a numeric expression.
- This function can be useful, when you are migrating SQL applications that were originally written for other database servers, if the application makes calls to a function of this name that returns a DECIMAL value.
- The following example retrieves a DECIMAL value that the **TO\_NUMBER** function returns from the literal representation of a MONEY value:
  - SELECT TO\_NUMBER('\$2500.00') from winnings;
  - Results: 2500.000000000000

# SQL Features – 11.10

- **Informix 11.10 New Features**

- **11.10xC1 – TRUNC()**

- The **TRUNC** function can reduce the precision of its first numeric, DATE, or DATETIME argument by returning the truncated value. If the first argument is neither a number nor a point in time, it must be cast to a numeric, DATE, or DATETIME data type
- The **TRUNC** function resembles the **ROUND** function, but truncates (rather than rounds to the nearest whole number) any portion of its first argument that is smaller than the least significant digit or time unit within the precision that its second argument specifies. For numeric expressions, **TRUNC** replaces with zero any digits less than the specified precision.
- For DATE or DATETIME expressions, **TRUNC** replaces any time units smaller than the format specification with 1 for *month* or *day* time units, or with 0 for time units smaller than *day*.
  - » SELECT TRUNC(invoice\_date, 'YEAR') FROM invoices;
  - » Returns: 2012-01-01 00:00.



# SQL Features – 11.50

- **Informix 11.50 New Features**

- **11.50xC1 – Dynamic SQL in SPL Routines**

- You can now use the following dynamic SQL statements in SPL routines:
      - EXECUTE IMMEDIATE
      - PREPARE
      - DECLARE
      - OPEN
      - FETCH
      - CLOSE
      - FREE



# SQL Features – 11.50

- **Informix 11.50 New Features**

- **11.50xC1 – EXECUTE IMMEDIATE**

- Use the EXECUTE IMMEDIATE statement to perform tasks equivalent to what the PREPARE, EXECUTE, and FREE statements accomplish, but as a single operation.

- ESQL


- » `sprintf(cdb_text1, "create database %s", usr_db_id);`

- » `EXEC SQL execute immediate :cdb_text1;`

- Stored Procedure

- `LET query_string='select * from test where num = 1'`

- `EXECUTE IMMEDIATE query_string`




# SQL Features – 11.50

- **Informix 11.50 New Features**

- **11.50xC1 – SQLCODE**

- Built-in function for Stored Procedures
    - Returns the value of **sqlca.sqlcode** for the most recently executed SQL statement. This function expression, which can be invoked only from SPL routines, is useful in error handling and in program logic to exit from a loop after the last row of the active set of a cursor has been processed.




# SQL Features – 11.50

- **Informix 11.50 New Features**

- **11.50xC2**

- Subquery Support in UPDATE and DELETE Statements
      - The FROM clause of a subquery in the WHERE clause of the DELETE or UPDATE statement can specify as a data source the same table or view that the FROM clause specifies.
    - SQL Expressions with the IS [NOT] NULL Predicate
      - Now you can use SQL expressions as operands in Boolean conditions that use the IS NULL or IS NOT NULL predicate.






# SQL Features – 11.50

- **Informix 11.50 New Features**

- **11.50xC2 – SUBQUERY SUPPORT (UPDATE/DELETE)**

- The FROM clause of a subquery in the WHERE clause of the UPDATE statement can specify as a data source the same table or view that the Table Options clause of the UPDATE statement specifies.
- UPDATE operations with subqueries that reference the same table object are supported only if all of the following conditions are true:
  - The subquery either returns a single row,
  - Has no correlated column references.
- No SPL routine in the subquery can reference the same table that UPDATE is modifying.
- Example:
  - UPDATE stock SET unit\_price = unit\_price \* 0.95
  - WHERE unit\_price IN (SELECT unit\_price FROM stock WHERE unit\_price > 50);




# SQL Features – 11.50

- **Informix 11.50 New Features**

- **11.50xC3 – SAVEPOINT**

- Ability to perform a partial rollback within a transaction.
    - In this example, it would rollback the insert with the value of 'TEST3':
      - BEGIN WORK;
      - INSERT INTO tab1 VALUES ('TEST1');
      - SAVEPOINT ins\_01;
      - INSERT INTO tab1 VALUES ('TEST2');
      - SAVEPOINT ins\_02;
      - INSERT INTO tab1 VALUES ('TEST3');
      - SAVEPOINT ins\_03;
      - ROLLBACK TO SAVEPOINT ins\_02;



# SQL Features – 11.50

- **Informix 11.50 New Features**

- **11.50xC6**


- **Load and Unload data with External Tables**

- You can read and write data from a source that is external to the database server. External tables provide an SQL interface to data in text files managed by the OS.

- **Light Scans on Tables**

- You can now enable Informix to perform light scans on compressed tables, tables with rows larger than a page, tables with VARCHAR, LVARCHAR, and NVARCHAR data. To enable light scans enable as follows

- » `BATCHEDREAD_TABLE 1` (ONCONFIG)
- » `IFX_BATCHEDREAD_TABLE=1` (Environment Variable)



# SQL Features – 11.50

- **Informix 11.50 New Features**

- **11.50xC6 – External Tables**

- **Ways to use External Tables**

- **Creating external tables**

- » Create table ext\_xyz (col1 integer, col2 char(20))  
USING (DATAFILES (“DISK:/data/xyz.unl”),  
FORMAT “DELIMITED”)

- » Create table ext\_xyz SAMEAS xyz  
USING (DATAFILES (“DISK:/data/xyz.unl”),  
FORMAT “DELIMITED”)



# SQL Features – 11.50

- **Informix 11.50 New Features**

- **11.50xC6 – External Tables**

- **Ways to use External Tables**

- Load data from external table into database table

- » `Insert into xyz select * from ext_xyz;`

- Unload data to external table

- » `Insert into ext_xyz select * from xyz;`



# SQL Features – 11.50

- **Informix 11.50 New Features**

- **11.50xC6 – External Tables**

- Using PIPE feature

- When you do not want to unload to a flat file, but pipe the data to another server/table you can use the PIPE feature.
- Here is what the create table statement looks like:
  - » Create external table ext\_xyz SAMEAS xyz  
USING ( DATAFILES("PIPE:/data/ext\_xyz1"))
- To improve performance using PIPES make sure that there are enough FIFO VP's defined. The database server uses one FIFO VP for each named pipe that specify in the DATAFILES clause.
  - » To add a FIFO VP – onmode -p + {# to add} FIFO




# SQL Features – 11.50

- **Informix 11.50 New Features**

- **11.50xC8**

- Lock tables from Updatable Secondary Server
      - You can set exclusive locks and shared locks from updatable secondary servers in a cluster.





# SQL Features – 11.70

- **Informix 11.70 New Features**

- **11.70xC1 – NEW OPTIMIZER DIRECTIVES**

- Query optimizer support for star-schema and snowflake-schema queries. A primary key column in each dimension table must correspond to a foreign key in the fact table.
      - New optimizer directives have been added:
        - » STAR\_JOIN, FACT, AVOID\_STAR\_JOIN & AVOID\_FACT
        - » Can also enable with SET OPTIMIZATION



# SQL Features – 11.70

- **Informix 11.70 New Features**

- **11.70xC1**

- Session-Level control of how much memory can be allocated to a query:
      - The SET ENVIRONMENT supports new BOUND\_IMPL\_PDQ and IMPLICIT\_PDQ.
      - BOUND\_IMPL\_PDQ – database server uses the explicit PDQPRIORITY as the upper bound for memory.
      - IMPLICIT\_PDQ – unless BOUND\_IMPL\_PDQ is also set, the database server ignores the current explicit setting of PDQPRIORITY and automatically determines an appropriate value.




# SQL Features – 11.70

- **Informix 11.70 New Features**

- **11.70xC1**

- Syntax support for DDL statement with IF [NOT] EXISTS:
      - Can include “IF [NOT] EXISTS” condition to SQL statements that create a database object or a database.




# SQL Features – 11.70

- **Informix 11.70 New Features**

- **11.70xC2**

- Table and column Aliases in DML statements.
  - SELECT – statements and subqueries can declare an alias in the projection clause for columns in the select list.
  - DELETE / UPDATE – can declare an alias for a local or remote target table.




# SQL Features – 11.70

- **Informix 11.70 New Features**

- **11.70xC3**

- Built-in SQL compatibility functions for string manipulation and trigonometric support.
      - These functions return either a character string or an integer that describes a string argument.
        - » CHARINDEX
        - » INSTR
        - » LEFT
        - » LEN
        - » REVERSE
        - » RIGHT
        - » SPACE
        - » SUBSTRING\_INDEX



# SQL Features – 11.70

- **Informix 11.70 New Features**

- **11.70xC3**

- There are two built –n trigonometric support functions. They convert the unit of angular measurement of a numeric expression argument from radians into degrees or from degrees into radians:
      - DEGREES
      - RADIANS



# SQL Features – 11.70

- **Informix 11.70 New Features**

- **11.70xC3 – CHARINDEX()**

- Searches a character string for the first occurrence of a target substring, where the search begins at a specified or default character position within the source string.
    - Example:
      - CHARINDEX('www.iiug.org','org')
      - Returns: 10






# SQL Features – 11.70

- **Informix 11.70 New Features**

- **11.70xC3 – INSTR()**

- Searches a character string for a specified substring, and returns the character position in that string where an occurrence of that a substring ends, based on a count of substring occurrences.
    - Example:
      - This specifies a count of 2, starting the search in the first character of the *source\_string*:
        - » `INSTR("www.espn.todayespn.com", "es", 1, 2)`
        - » The expression above returns 15, the character position where the second 'es' begins.



# SQL Features – 11.70

- **Informix 11.70 New Features**

- **11.70xC3 – LEFT()**

- Returns a substring consisting of the leftmost *N* characters from a string.
- Example:
  - LEFT('www.espn.com',8)
    - » Returns: “www.espn”




# SQL Features – 11.70

- **Informix 11.70 New Features**

- **11.70xC3 – LEN()**

- Just like LENGTH function
    - Returns the number of bytes in a character column, not including any trailing blank spaces. For BYTE or TEXT columns, **LENGTH** returns the full number of bytes, including any trailing blank.
    - Example:
      - SELECT order\_num, LENGTH(order\_desc) + LENGTH(order\_type)  
FROM orders WHERE LENGTH(order\_desc) > 10;
      - EXECUTE FUNCTION LEN("www.espn.com");



# SQL Features – 11.70

- **Informix 11.70 New Features**

- **11.70xC3 – REVERSE()**

- Accepts a character expression as its argument, and returns a string of the same length, but with the ordinal positions of every logical character reversed.
    - Example:
      - SELECT REVERSE('Baseball has 162 games') FROM sports\_schedule
      - Returns “semag 261 sah llabesaB”




# SQL Features – 11.70

- **Informix 11.70 New Features**

- **11.70xC3 – RIGHT()**

- Returns a substring consisting of the rightmost *N* characters from a string.
    - Example:
      - SELECT RIGHT('Cubs Win World Series',13) FROM baseball
      - Returns "World Series"



# SQL Features – 11.70

- **Informix 11.70 New Features**

- **11.70xC3 – SPACE()**

- Creates a character string of a specified number of blank spaces. The maximum length of the returned string value can be 32,739 blank.
    - The argument to the **SPACE** function must be of a built-in data type.
    - The **SPACE** function returns an LVARCHAR string of the specified number of blank (ASCII 32 ) characters.
    - Example: `select SPACE(123) from employee`
      - Returns a single blank character



# SQL Features – 11.70

- **Informix 11.70 New Features**

- **11.70xC3 – SUBSTRING\_INDEX()**

- Searches a character string for a specified delimiter character, and returns a substring of the leading or trailing characters, based on a count of a delimiter that you specify as an argument to the function.
- Example:
  - SUBSTRING\_INDEX("www.iiug.org", ".", 2)
  - Returns: the leading characters www.iiug because *count* > 0.





# SQL Features – 11.70

- **Informix 11.70 New Features**

- **11.70xC3**

- Case-insensitive queries on NCHAR and NVARCHAR text strings. You can use the NLSCASE INSENSITIVE option with the CREATE DATABASE statement.
      - Querying “McDonalds” returns:
        - » “McDonalds”
        - » ”mcdonalds”
        - » “MCDONALDS”



# SQL Features – 11.70

- **References**

- IBM Informix Information Center – 11.70

- <http://publib.boulder.ibm.com/infocenter/idshelp/v117/index.jsp>

- IBM Informix Information Center – 11.50

- <http://publib.boulder.ibm.com/infocenter/idshelp/v115/index.jsp>

- IBM Informix Information Center – 11.10

- <http://publib.boulder.ibm.com/infocenter/idshelp/v111/index.jsp>

# Questions?!?



# 2012 IIUG Informix Conference

The largest Informix gathering in the world

## Informix SQL Using New SQL Features

Jeff Filippi



Integrated Data Consulting, LLC

[jeff.filippi@itdataconsulting.com](mailto:jeff.filippi@itdataconsulting.com)

[www.itdataconsulting.com](http://www.itdataconsulting.com)

630-842-3608