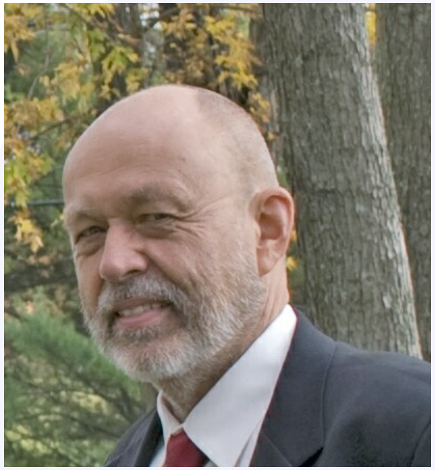
Advanced Schema Design Webcast
presented by:

Advanced DataTools

November 30, 2017

Lester Knutsen



Lester Knutsen is President of Advanced DataTools Corporation, and has been building large Data Warehouse and Business Systems using Informix Database software since 1983. Lester focuses on large database performance tuning, training and consulting. Lester is a member of the IBM Gold Consultant program and was presented with one of the Inaugural IBM Data Champion awards by IBM. Lester was one of the founders of the International Informix Users Group and the Washington Area Informix User Group.

lester@advancedatatools.com
www.advancedatatools.com
703-256-0267 x102

Tom Beebe

Tom is a Senior Database Consultant and has been with Advanced DataTools for over 14 years. He has been working with Informix since college with a long time fondness for open source languages. Tom is the lead consultant for Networking, Unix System Administration and Web Development needs. Currently, he is the Project Manager and lead developer on a variety of Web Development projects.

Contact Info:

tom@advancedatatools.com

www.advancedatatools.com

703-256-0267 x 106

Mike Walker

Mike Walker has been using Informix databases for 18 years, as a developer and as a database administrator.

Recently Mike has been developing and supporting large data warehouses for the Department of Agriculture.

Contact Info:
mike@advancedatatools.com
www.advancedatatools.com
Office: 303-838-0869
Cell: 303-909-4265

Art Kagel

Art Kagel is President and Principal Consultant at
ASK Database Management
partnering with Advanced DataTools.

Art is a member of the IIUG Board of Directors and a recipient of the IIUG Directors Award. Art is an eight time recipient of the IBM Champion Award for Information Management. Art has over thirty years of database experience.

He has served as Manager of Database Systems and Services for a major information systems and media corporation and is a leading consultant specializing in IBM and HCL Informix Servers.

Contact Info
art@askdbmgt.com
www.askdbmgt.com

Phone: 732-993-5367

Agenda

- Normalization in RDBMS
- Implementing many-to-many relationships
- Implementing relationship attributes
- Non-Normalized Data Types
  - – ROW Type
  - – Sets, Lists, Multisets
  - – BSON/JSON Types & Fields
    - Arrays
    - Nested Documents
- Non-SQL Databases

Normalization

The purpose of database schema Normalization is to improve the quality of data and eliminate pathological relationships within and between data elements that can make it difficult to satisfy reasonable queries.

Normalization

Normalization is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like Insertion, Update and Deletion anamolies. It is a multi-step process that puts data into tabular form by removing duplicated data from the relation tables.

Normalization

There are several levels of Normalization.

Some of these depend on or expand on the previous level. Others, concentrated on resolving a different problem, are independent of the lower levels.

However, it is very difficult to accomplish even these independent higher levels of normalization with data that does not already satisfy the lower level.

Normalization

Non-Normalized schema contain data duplication and other problems that make maintaining your data in a sane and consistent condition difficult and expensive and sometimes impossible to achieve!

Normalization

First Normal Form:

Each attribute is an atomic or simple value

Each attribute can have exactly one value (no arrays)

Every row of a table must have the same number of attributes.

Normalization

First Normal Form violation:

First Name	Last Name	Birth Date	Phone #	Email
Art	Kagel	1955-05-31	732-213-5367, 732-993-5367, 732-377-8739, 732-613-9511	art.kagel@gmail.com, art@advancedatatools.com, art@askdbmgt.com art@iiug.org

This table is not in 1NF because phone # and email are multi-valued fields.
Break out phone # and email into separate tables with the name tables primary key as foreign key.

Normalization

Second normal form:

The table adheres to First Normal Form and:

An attribute must be dependent on the table's whole key.

Normalization

Second normal form violation:

Warehouse#	Part #	Warehouse Addr.	Part Location
1	132453	23 Main St.	Row #7 Bin #6
2	345436	23 Smith Ave.	Row #2 Bin #23
2	132453	23 Smith Ave.	Row #4 Bin #6

The warehouse address is only dependent on the warehouse # while the natural key of this table is warehouse # and part #. The warehouse's address should be part of a separate warehouse table, so this table is not 2NF.

Normalization

Third normal form

Table adheres to Second Normal Form and:

Each attribute is dependent on the table's key, its whole key, and nothing but its key.

Normalization

Third normal form violation:

Employee Id	Department	Department Location
12345	Math	Bldg#7 Flr #2 Rm #4
23432	Languages	Bldg#5 Flr #1 Rm #4

Department location attribute is dependent on Department which is another attribute not a key, therefore this table is not in 3NF.
There should be a Department table with the department name and location and perhaps a surrogate key that can be used here as a foreign key here in the employee table.

Normalization

Boyce-Codd Normal Form (BCNF)

A table cannot have overlapping candidate keys.

Every determinant key must be a candidate key (ie must determine all attributes of the table)

Normalization

Boyce-Codd Normal Form (BCNF) violation:

Student	Major	Advisor
Kagel	Math	Rabinowitz
Lowell	Literature	Franks
Menendez	Math	Rabinowitz
Farber	History	Jones

If Lowell is the only Literature major and he leaves the school, we lose the knowledge/information that Dr. Franks is the Literature Advisor. This should be decomposed into two tables, Student-Major and Major-Advisor and so is not in BCNF.

Obviously if Advisors are personal to the student that's a different animal and may indicate the need for a many-to-many relationship between advisors and students and maybe even majors.

Normalization

Fourth Normal Form:

A table must not contain more than one independent attribute of the entity.

All attributes must be related to each other.

Normalization

Fourth Normal Form violation:

Employee	Skill	Language
Kagel	Modula2	French
Kagel	FORTRAN	German
Kagel	SQL	
Beebe	PHP	English

The attributes Skill and Language are both dependent only on Employee, so the table is 3NF and BCNF. However, there is no functional dependency between Skill and Language. Does Kagel only program Modula2 in French? FORTRAN in German? Nien! So, not 4NF.

Normalization

Fifth Normal Form:

Table cannot be decomposed into multiple tables with different keys that can be joined to produce the same information content

Normalization

Fifth Normal Form violation:

SalesPerson	Territory	Product
Smith	East End	First Aid
Smith	East End	Surgical Supplies
Smith	East Central	Foot Relief
Jones	West End	Surgical Supplies
Jones	Central	First Aid

Are salespersons restricted to selling specific products in specific territories? Or, rather, are they authorized to sell a set of products across a set of territories? If the latter, then this table is not 5NF and should be two tables.

Implementing many-to-many relationships

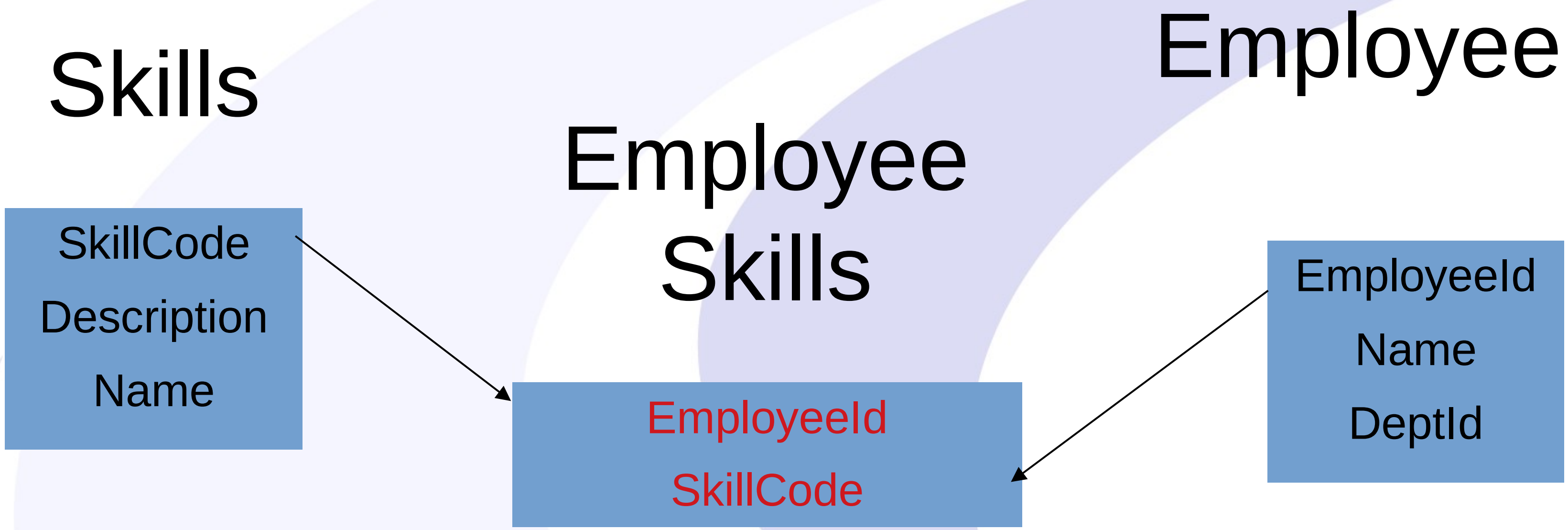
Implementing a many-to-many relationship between the elements of two tables is not as simple as adding a foreign key to one table or the other.

Implementing many-to-many relationships

Let's return to our 4NF example

- Employees can be accomplished in many skills
- A skill can be an accomplishment of many employees
- One could say that each employee is linked to many skills and each skill is linked to many employees. That link is the "key" to the relationship.

Implementing many-to-many relationships



Implementing many-to-many relationships

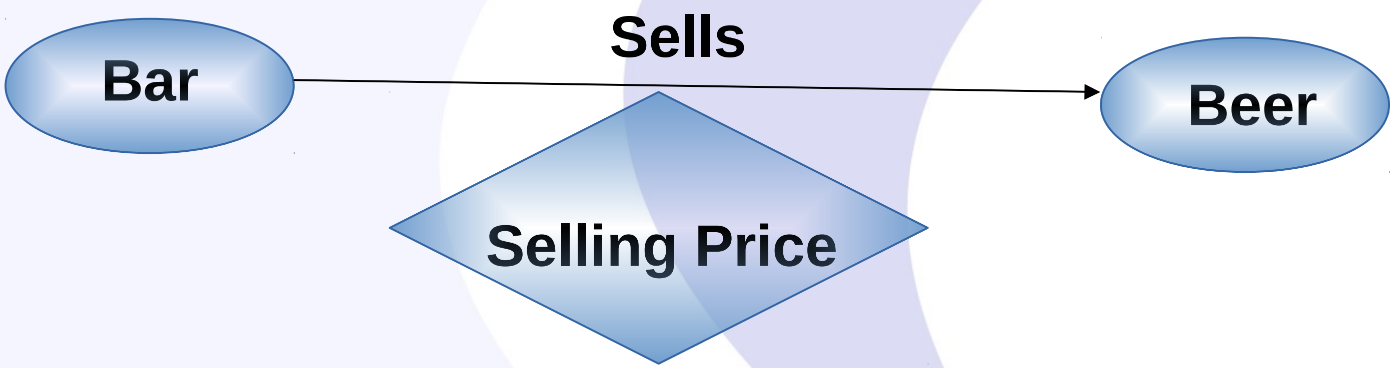
Let’s return to our 4NF example

You need to create a third table know variously as a link table or a gerund table that contains the foreign keys for the records of the two tables being related. There will be a record in the link table for every existing pairing of a record in one table to a record in the other table.

Implementing Relationship Attributes

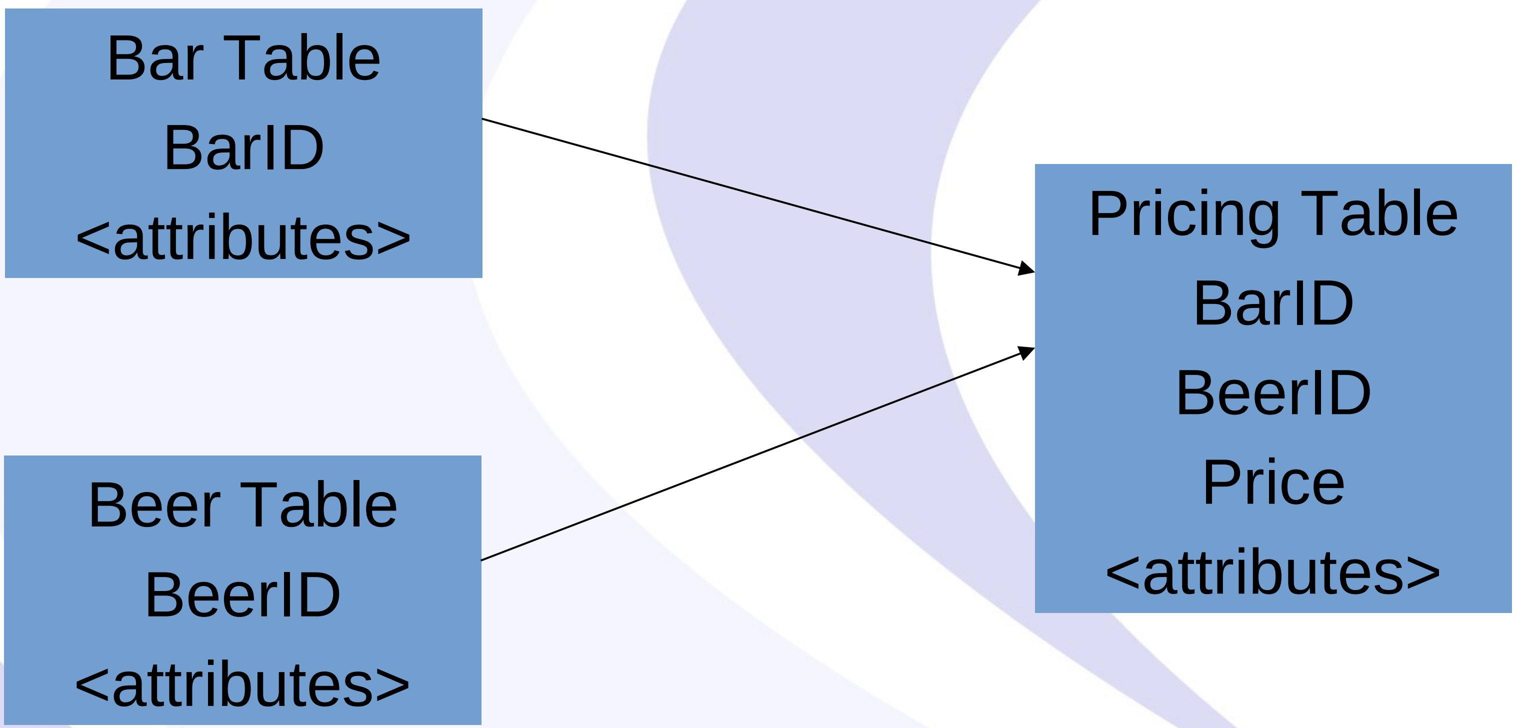
Sometimes the relationship between tables has attributes of its own that do not belong to any of the tables that the relationship describes!

Implementing Relationship Attributes

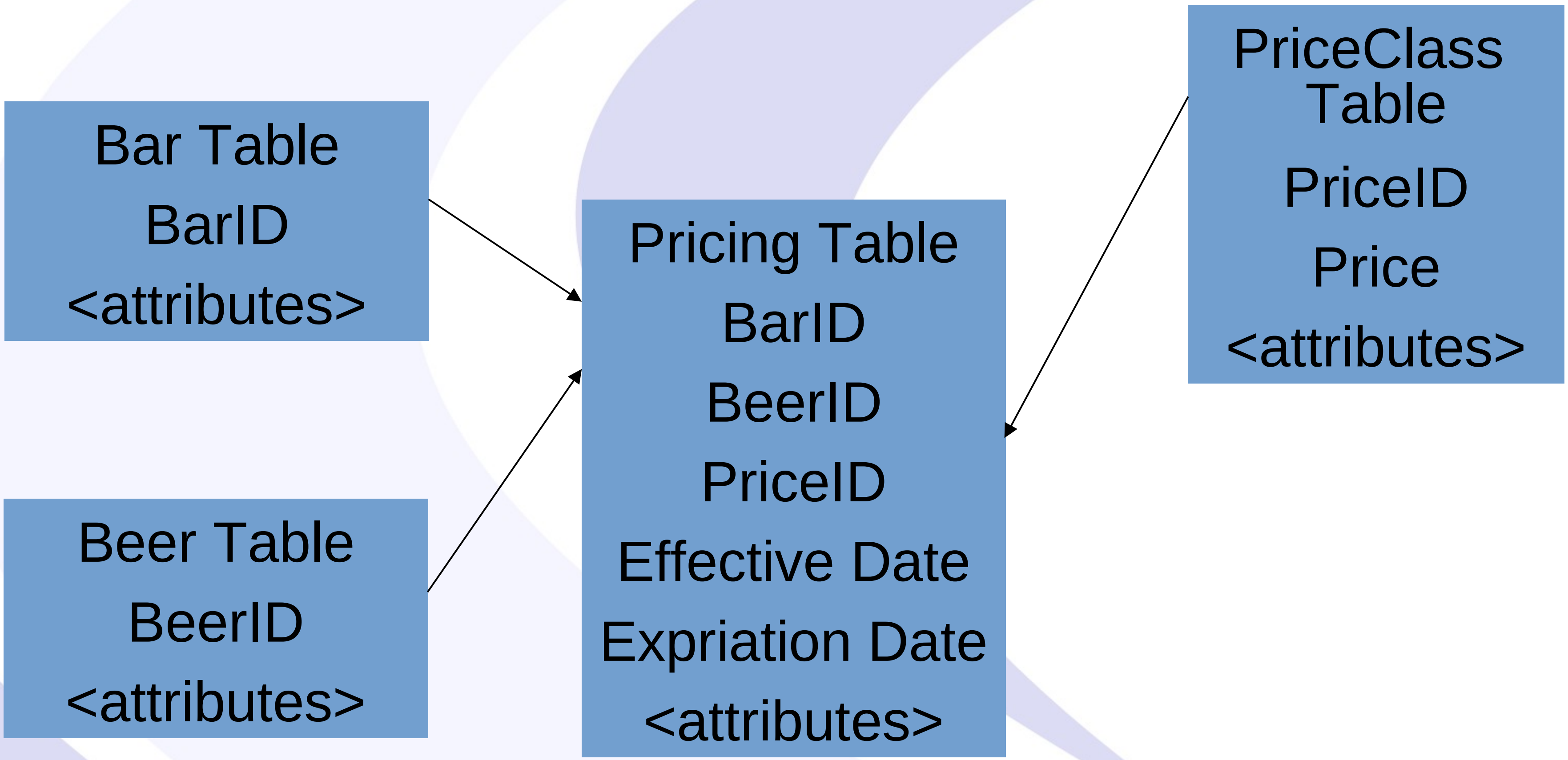


In a database for Internet users searching for a nearby bar that sells their favorite beers at the best prices, price cannot be an attribute of the beer alone as it could be for a database designed for a single bar!

Implementing Relationship Attributes



Implementing Relationship Attributes



Non-Normalized Data Types

The data types we will cover will cause your table structure to violate one or more of the rules of Normalization at one level or another.

Non-Normalized Data Types

ROW Types

Create a complex data type from other types:

```
CREATE ROW TYPE address_t (
    StreetAddress LVARCHAR(100),
    City LVARCHAR(100),
    StateCode CHAR(2),
    ZipCode CHAR(5),
    ZPlus4 CHAR(4)
);
```

Non-Normalized Data Types

SET

A SET is a collection of non-null unique items of any type presented in no particular order.

```
CREATE TABLE person (
    ...
    addresses SET( address_t NOT NULL ),
    ...
);
```

Non-Normalized Data Types

LIST

A LIST is an ordered collection of non-null items of any type. Items in the LIST are presented in the order in which they are added to the collection by default. Duplicate valued items are supported.

```
CREATE TABLE person (
    …
    addresses LIST( address_t NOT NULL ),
    …
);
```

Non-Normalized Data Types

MULTISET

A set is a collection of non-null items or any type presented in no particular order. Duplicate valued items are supported.

```
CREATE TABLE person (
    …
    addresses MULTISET( CHAR(10) NOT NULL ),
    …
);
```

Non-Normalized Data Types

BSON & JSON

JSON – Java Script Object Notation

A text based document interchange format that uses tag/value elements to represent fields. Fields are implicitly typed and the internal format of collected documents may differ from one document to another.

BSON – a binary format for storing JSON documents that enhances the document with formal field type and length data needed to efficiently parse the document's fields.

Non-Normalized Data Types

JSON

Informix supports JSON type columns, however, the only valid conversions from JSON are to BSON and character types. You cannot access individual fields within a JSON type column or host variable.

Non-Normalized Data Types

BSON

Informix supports BSON type columns. The only valid conversion or cast from BSON is to JSON and only JSON can be cast to BSON.

You can reference fields within a BSON type column using "dot" notation: document.fieldname. If a particular fieldname is not present in the column's value for a row Informix returns a NULL.

You can index BSON fields and use them as foreign key constraints.

There are functions for extracting BSON fields in addition to the "dot" notation.

Non-Normalized Data Types

BSON & JSON

Field types include:

- Text strings
- Integer
- Floating point numbers
- Embedded JSON/BSON documents
- Arrays of supported types (including embedded JSON/BSON)
- Links to other documents (actually primary keys)
- Boolean
- BSON supports other types that do not correspond to valid SQL types and are represented in text when cast to JSON.

Non-SQL Databases

Not all data is best represented by a relational model database like Informix. While ideal for OLTP and for many well structured data warehouse applications, some data sets lend themselves to other representations that are not handled well by RDBMS and other databases that use query languages like SQL.

Non-SQL Databases

Tag/Value Databases

These are fairly simple databases that store a single element or attribute that describes a tag. The elements are very similar to the fields in a JSON document. Ex:

Name: Fred

Examples include:

- BerkleyDB
- GNU DBM
- Riak
- Oracle NoSQL

Non-SQL Databases

Hierarchical

Parent records contain links to child records. Often parents and children are stored contiguously on disk. Very fast to navigate down the hierarchy, ie from parent to child. Cannot navigate up the hierarchy. Cannot search for or navigate directly to a child.

Expensive to restructure data or modify the hierarchy.

Examples:

- IMS
- MariaDB

Non-SQL Databases

Network aka CODASYL

Similar to Hierarchical but references can be more complex. Links between siblings and from child to parent are common.

Similar problems as hierarchical to reorganize relationships.

Examples:

- IDMS
- RDM Server

Non-SQL Databases

Object Database

Store complex objects, sometimes including methods. Support inheritance and other OO features.

Many ODs cannot update existing objects and must delete and re-insert modified data making updates expensive. Expensive to modify the structure of an object set.

Examples:

- Encore/DB
- Objectivity/DB
- Versant

Non-SQL Databases

Object Relational

These are databases that are primarily Relational but also support defining complex data types, inheritance, and other OO features.

Examples:

- Informix Dynamic Server
- PostgreSQL
- Cache

Non-SQL Databases

Graph or Navigational

These support complex relationships and are a modern take on Network databases. Simple graph structures also emulate Hierarchical databases.

Examples:

- Dgraph
- Datastax
- Marklogic

Non-SQL Databases

Unstructured Data Stores

These are formalized stores for unstructured, unfiltered, data in whatever format the data is naturally found. They are very efficient at importing vast quantities of data as little massaging of the incoming data is performed. Searches use massively parallel operations like MapReduce that are independent of the data store itself to produce results in reasonable time. Unstructured data stores provide few if any query functionality.

Example:

- Hadoop

Questions?

Send follow-up questions to lester@advancedatatools.com or art@askdbmgt.com

Next Webcast

Date: **Thursday, December 14, 2017**

Time: **2:00pm EST**

Informix Best Practices

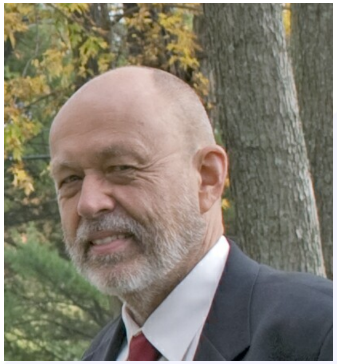
Getting Started with Enterprise Replication

by Thomas Beebe

Informix Training in 2018

- **February 5-8, 2018 - Advanced Informix Performance Tuning.**
- **April 23-26, 2018 - Informix for Database Administrators.**
- **September 17-20, 2018 - Informix for Database Administrators.**

- All courses can be taken online on the web from your desk or at our training center in Virginia.
- We guarantee to *NEVER* cancel a course and will teach a course as long as one student is registered!



Thank You

Lester Knutsen

Advanced DataTools Corporation

lester@advancedatatools.com

For more information:

http://www.advancedatatools.com