NoSQL & Informix

The Power of Hybrid – Informix and JSON

Andreas Weininger
Andreas.Weininger@de.ibm.com

Why MongoDB

- **Rapid Application Prototyping**
  - Schema rides as objects inside the JSON Script, not as a separate item. Allows for constant unit test and retest as logic expands
    - Schemas change quite quickly in modern apps as business needs change.
    - Apps have shorter production lives and even shorter life-cycles as a result
    - Knows and understands JSON natively.
    - No need to normalize per se.
    - 1 to 1 and many to 1 RDBMS relationships do not apply here.
  - Less bureaucratic
    - Stripped down IS organizations, lean and mean
    - Programmer can be the DBA, system administrator as well.

- **Scalability**
  - Scalability is not by database tuning per se and adding resources, but rather by sharding (fragmenting the data by a key) the data when necessary across cheap, inexpensive commodity network hardware
    - Automatic data rebalancing across shards by key occurs during the shard operation.

- **Costs of initial “up and running” are less**
- **Quicker project completions times**

Why MongoDB - Business

- **600+ Customers worldwide**
- **5,000,000+ Free Software Downloads**
- **Enterprise Big Data customers**
- **Training is Free**
- **Charges for Services as an add on, complimentary business**
  - Backup/Restores
  - Consulting

Partnership with IBM and MongoDB

- **MongoDB and IBM announced a partnership in June 2013**

- **There are many common use cases of interest addressed by the partnership**
  - Accessing JSON Data in DB2, Informix MongoDB using JSON query
  - Schema-less JSON Data for variety of applications
  - Making JSON data available for variety of applications
  - Securing JSON Data
- **IBM and MongoDB are collaborating in 3 areas:**
  - Open Governance: Standards and Open Source
  - Technology areas of mutual interest
  - Products and solutions

NoSQL Market Data

RedMonk

"MongoDB is the new MySQL."

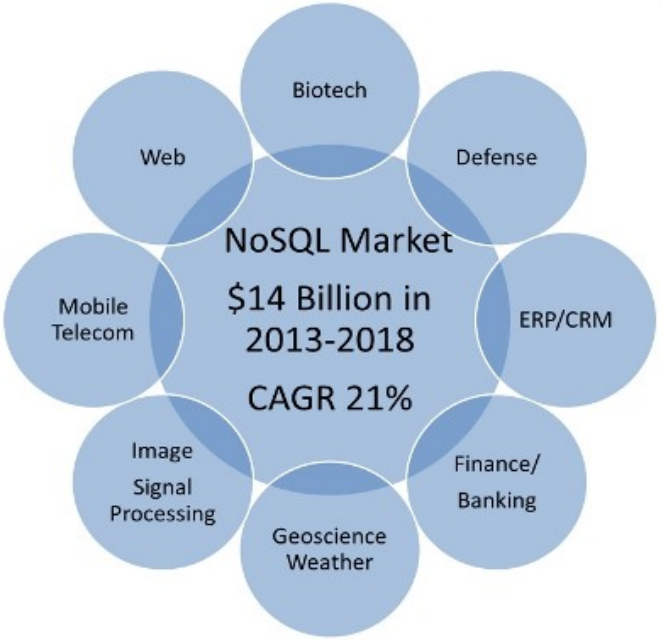
FORRESTER®

"Current adoption of NoSQL in enterprises is 4% in 2010. expected to double by 2014, and grow to 20% by 2015. 10% of existing apps could move to NoSQL. [Sept 2011]

Gartner

"NoSQL is in its infancy, many immature with only community support and limited tool capability; however, expect to see a few used more widely during the next 5 years."

- **2011: Market Research Media noted worldwide NoSQL market was expected to reach ~$3.4B by 2018, generating $14B from 2013-2018 with a CAGR of 21%**
- **Comparison: data implies NoSQL market ~$895M**
  - MySQL in 2011 ~$100M



Informix *and* MongoDB Have Free Editions

Editions	Informix	MongoDB
Free	Developer Innovator-C	Basic
For Purchase	Express, Workgroup, Advanced Workgroup, Enterprise, Advanced Enterprise	Standard, Enterprise

MongoDB Subscriptions

	Basic	Standard	Enterprise
Edition	MongoDB	MongoDB	MongoDB Enterprise
Support	9am-9pm local, M-F	24x7x365	24x7x365
License	AGPL	Commercial	Commercial
Emergency Patches	Not Included	Included	Included
Price	$2,500 / Server / Year	$5,000 / Server / Year	$7,500 / Server / Year

Additional monthly charges for backup services.

Subscription information obtained from 10Gen site, June 26, 2013.

Price Point Comparison Estimate, 3-year cost

Dual Core Intel Nehalem	Innovator-C	Express (4 core, 8 GB, 2 nodes)	Workgroup (16 core, 16 GB, unlimited nodes)
Product Cost	$0	$8,540	$19,740
Support Subscription Year 1 24 x 7 x 365 Production System Down Development Call Emergency Patches Free Upgrades	$1,680	Included	Included
Support Renewal Year 2	$1,680	$1,708	$3,948
Support Renewal Year 3	$1,680	$1,708	$3,948
Total	$5,040	$11,956	$27,636

MongoDB Enterprise, 3-year cost*: $22,500*

Subscription information obtained from 10Gen site, June 26, 2013.
Retail U.S. prices subject to change, valid as of June 26, 2013.

Informix as a Hybrid Data Storage Database

- **Many of MongoDB's 600+ customers are enterprise that have both relational and non-relational data sitting in the same facility and need to make sense of the data they have for business purposes.**

- **MongoDB has no transaction capabilities in the SQL sense**
  - Informix will enforce transactions on all application statements, with consistency assured at the end of the transaction, MongoDB does not assure here and cannot
    - It is possible that in MongoDB that 4 out of 7 deletes will delete and at the end the other 3 remain, to be dealt with manually …… someday.

- **Joining noSQL and SQL based queries in the same statement is not possible within MongoDB**
  - Many customers using MongoDB in production circumstances with large document store databases need to join this data to their **existing** SQL based database repositories for business purposes, often stored, housed and queried separately:
    - Informix can join the two kinds of data
      - Document store portion stored natively in a BSON as a single row data type within Informix, accessed as JSON, with a set full set of functions to access the document store data.

NoSQL – Feature Checklist

JSON Features in Informix

Flexible Schema	▪ Use BSON and JSON data type. ▪ Complete row is stored in a single column ▪ BSON, JSON are multi-rep types and can store up to 2GB.
Access Key Value Pairs within JSON	▪ Translate the Mongo queries into SQL expressions to access key-value pairs. ▪ Informix has added expressions to extract specific KV pairs. ▪ Select bson_new(data, “{id:1, name:1, addr:1}”) from tab;
Indexing	▪ Support standard B-tree index via functional indexing. ▪ Type Less Index → Create index ix_1 on t(bson_extract(data, “id”); ▪ Typed Index → Create index ix_2 on t(bson_value_lvarchar(data, “name”), bson_value_date(data, “expire”); ▪ Informix also supports indexing bson keys with different data types.
Sharding	▪ Supports range & hash based sharding. ▪ Informix has built-in technology for replication. ▪ Create identical tables in multiple nodes. ▪ Add meta data for partitioning the data across based on range or hash expressions.

JSON Features in Informix

Select	▪ Limited support required for sharding, Mongo API disallowing joins. ▪ The query on a single sharded table is transformed into a federated UNION ALL query; includes shard elimination processing.
Updates (single node)	▪ INSERT: Simple direct insert. ▪ DELETE: DELETE statement with WHERE bson_extract() > ?; or bson_value..() > ? ▪ UPDATE: ▪ bson_update(bson, “update expr”) will return a new bson after applying the bson expression. Simple updates to non_sharded tables will be direct UPDATE statement in a single transaction. ▪ UPDATE tab bsoncol = bson_update(bsoncol, “expr”) where …
Updates (sharded env)	▪ **INSERT** – All rows are inserted to LOCAL shard, replication threads will read logs and replicate to the right shard and delete from local node (log only inserts underway). ▪ **DELETE** – Do the local delete and replicate the “delete statements” to target node in the background ▪ **UPDATE** – Slow update via select-delete-insert.
Transaction	▪ Each statement is a distinct transaction in a single node environment. ▪ The data and the operation is replicated via enterprise replication.

JSON Features in Informix

Isolation levels	▪ **NoSQL session can use any of Informix isolation levels.**
Locking	▪ **Application controls only on the node they're connected to.** ▪ **Standard Informix locking semantics will apply when data is replicated and applied on the target shard.**
Hybrid Access **(From MongoAPI to relational tables)**	▪ **INSERT** – All rows are inserted to LOCAL shard, replication threads will read logs and replicate to the right shard and delete from local node (log only inserts underway). ▪ **DELETE** – Do the local delete and replicate the "delete statements" to target node in the background ▪ **UPDATE** – Slow update via select-delete-insert.
Hybrid Access **(From SQL to JSON data)**	▪ **Directly get binary BSON or cast to JSON to get in textual form.** ▪ **Use expressions to extract to extract specific key-value pairs.** NoSQL collections will only have one BSON object in the table. We can "imply" the expressions when the SQL refers to a column. **SELECT t.c1, t.c2 from t; ➔ SELECT bson_extract(t.data, "{c1:1}"), bson_extract(t.data, "{c2:1}") from t;**

Flexible Schema

- **Clients exchange BSON document with the server both for queries and data.**
- **Thus, BSON becomes a fundamental data type.**
- **The explicit key-value(KV) pairs within the JSON/BSON document will be roughly equivalent to columns in relational tables.**
- **However, there are differences!**
  - The type of the KV data encoded within BSON is determined by the **client**
  - Server is unaware of data type of each KV pair at table definition time.
  - No guarantees that data type for each key will remain consistent in the collection.
  - The keys in the BSON document can be arbitrary;
  - While customers exploit flexible schema, they're unlikely to create a single collection and dump ***everything under the sun*** into that collection.
  - Due to the limitations of Mongo/NoSQL API, customers typically de-normalize the tables
    - (customer will have customer+customer addr + customer demographics/etc) to avoid joins.

Flexible Schema – Informix Implementation

- **Informix has a new data type, BSON, to store the data.**
- **Informix also has a JSON data type to convert between binary and text form.**
- **BSON and JSON are abstract data types (like spatial, etc).**
- **BSON and JSON multi-representational types.**
  - Objects up to 4K is stored in data pages.
  - Larger objects (up to 2GB) are stored out of row, in BLOBs.
    - MongoDB limits objects to 16MB

  - **This is all seamless and transparent to applications.**

Flexible Schema – Informix Implementation

- **CREATE TABLE customer (data BSON)**

- **BSON is the binary representation of JSON.**
  - It has length and types of the key-value pairs in JSON.

- **MongoDB drivers send and receive in BSON form.**

```
{"hello":          →  "\x16\x00\x00\x00\x02hello\x00
"world"}                \x06\x00\x00\x00world\x00\x00"

                     "\x31\x00\x00\x00\x04BSON\x00\x26\x00
{"BSON":              \x00\x00\x020\x00\x08\x00\x00
["awesome",       →   \x00awesome\x00\x011\x00\x33\x33\x33\x33\x33\x33
5.05, 1986]}          \x14\x40\x102\x00\xc2\x07\x00\x00
                      \x00\x00"
```

Accessing KV pairs within JSON

- **Extract Expressions/functions returning base type**

  bson_value_bigint(BSON, “key”);
  bson_value_lvarchar(bsoncol, “key.key2”);
  bson_value_date(bsoncol, “key.key2.key3”);
  bson_value_timestamp(bsoncol, “key”)
  bson_value_double(bsoncol, “key”);
  bson_value_boolean(bsoncol, “key”);
  bson_value_array(bsoncol, “key”);
  bson_keys_exist(bsoncol, “key”);
  bson_value_document(bsoncol, “key”)
  bson_value_binary(bsoncol, “key”)
  bson_value_objectid(bsoncol, “key”)

- **Expression returning BSON subset. Used for bson indices.**

  bson_extract(bsoncol, “projection specification”)

- **Expressions to project out of SELECT statement.**

  bson_new(bsoncol, “{key1:1, key2:1, key3:1}”);
  bson_new(bsoncol, “{key5:0}”);

Accessing Key-Value (KV) pairs within JSON.

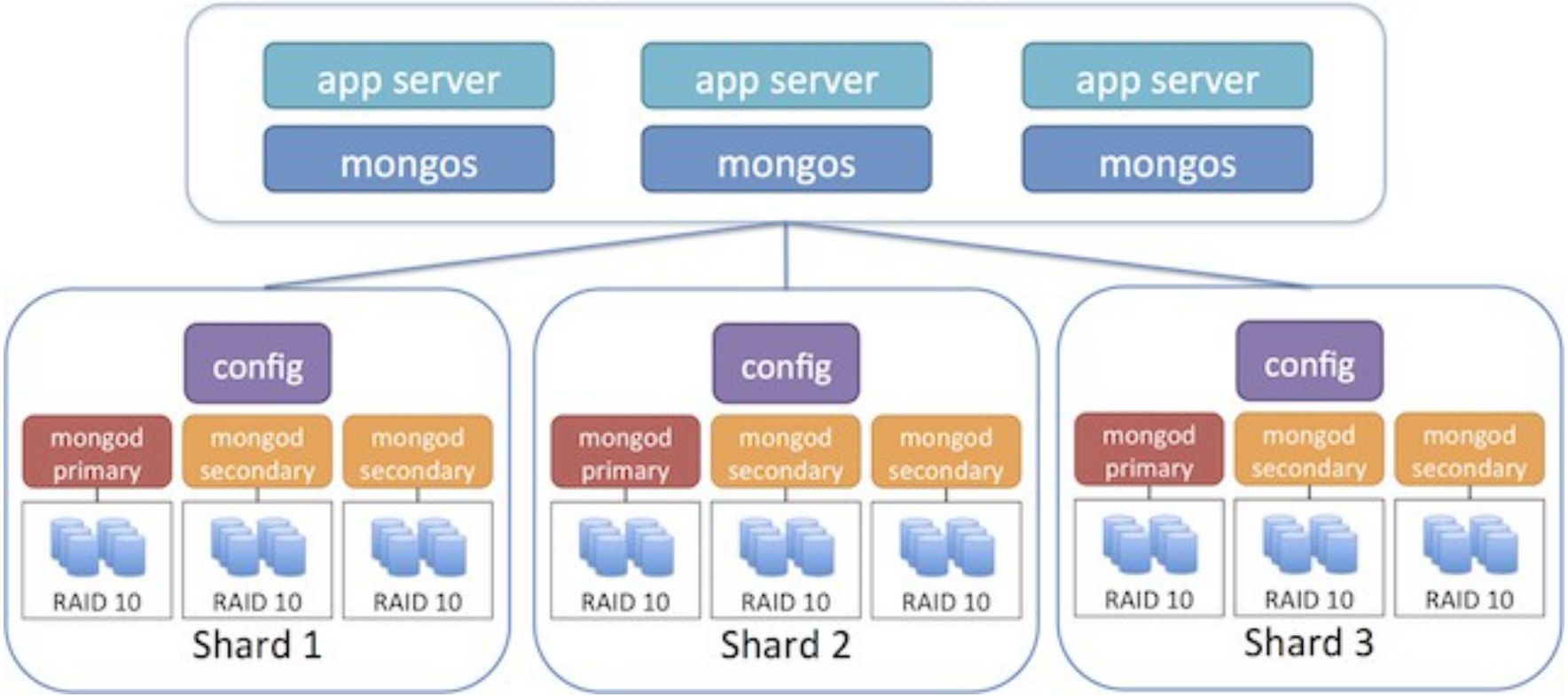
Mongo Query	SQL Query
db.customers.find();	SELECT SKIP ? data::bson FROM customers
db.customers.find({}, {num:1,name:1});	SELECT SKIP ? bson_new(data, '{ "num" : 1.0 , "name" : 1.0}')::bson FROM customers
db.customers.find({}, {_id:0,num:1,name:1});	SELECT SKIP ? bson_new(data, '{_id:0.0, "num" : 1.0 , "name" : 1.0}')::bson FROM customers
db.customers.find({status:"A"})	SELECT SKIP ? data::json FROM customers WHERE bson_value_lvarchar(data,"status")= "A"
db.customers.find({status:"A"}, {_id:0,num:1,name:1});	SELECT SKIP ? bson_new(data, '{ "_id" : 0.0 , "num" : 1.0 , "name" : 1.0}')::bson FROM customers WHERE bson_extract(data, 'name') = "A"

Indexing

- Supports B-Tree indexes on any key-value pairs.
- Indices could be on simple basic type (int, decimal) or BSON
- Indices could be created on BSON and use BSON type comparison
- Listener translates ensureIndex() to CREATE INDEX
- Listener translates dropIndex() to DROP INDEX

Sharding

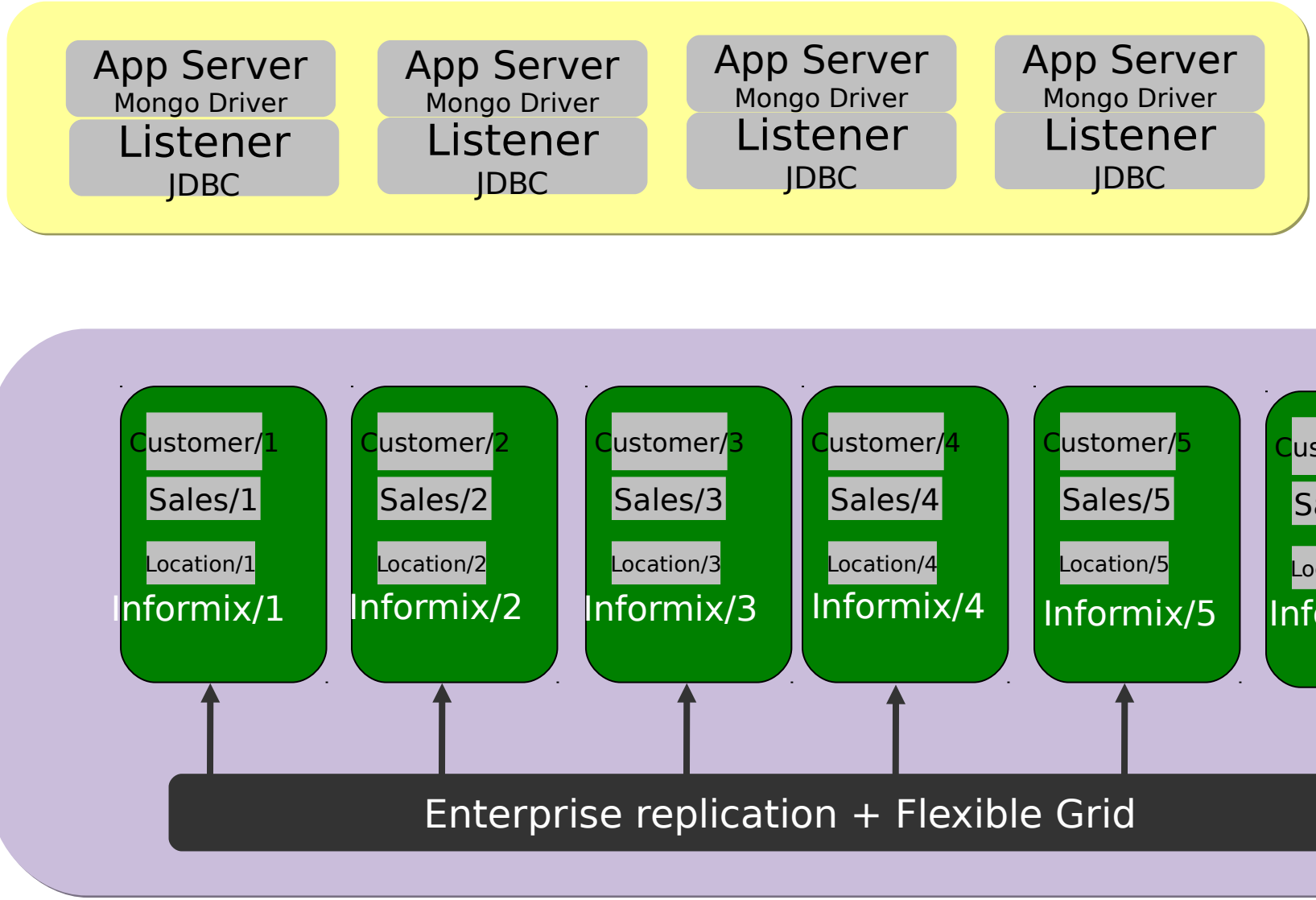
Mongo Sharding



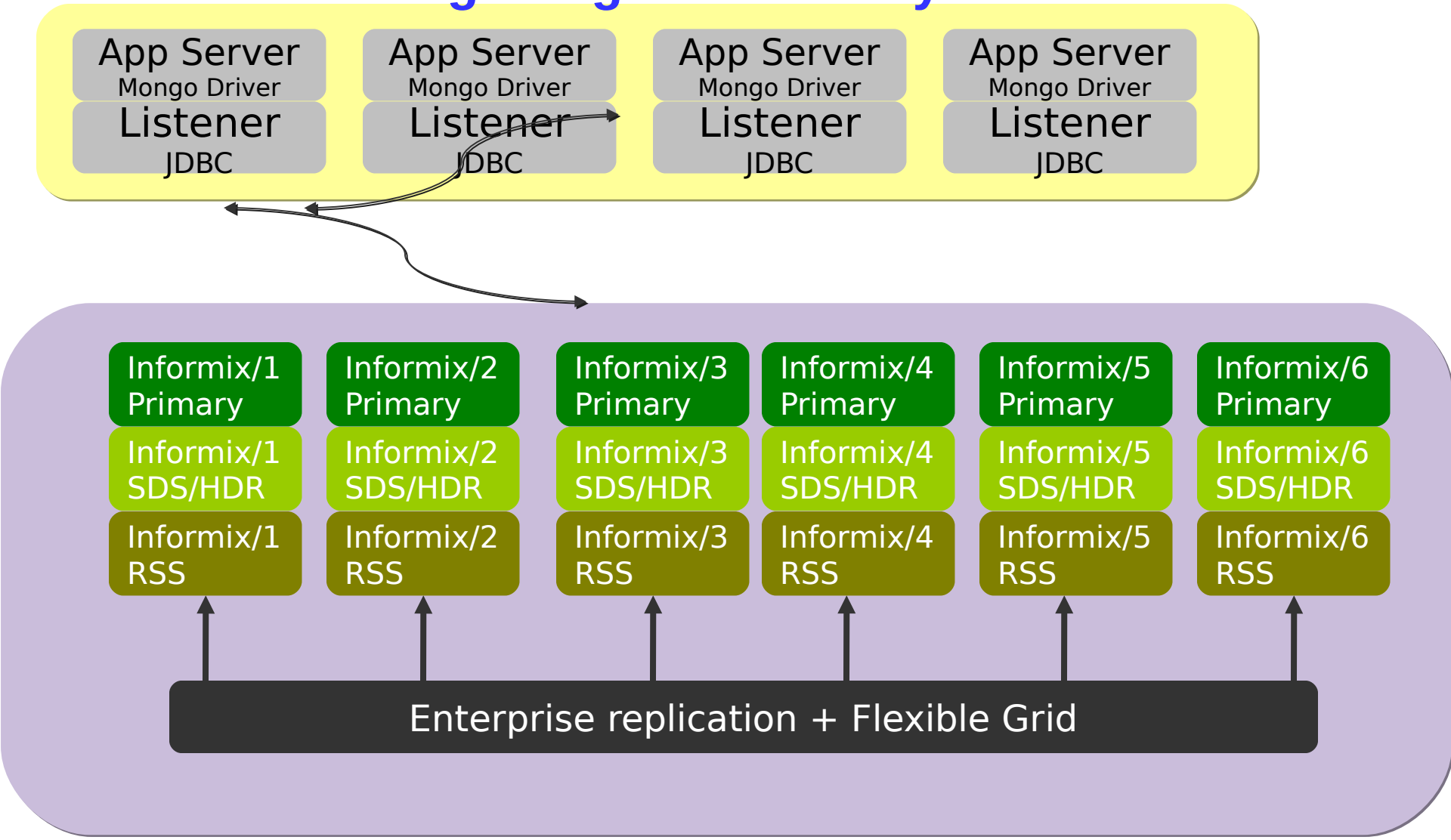
MongoDB SHARDING (roughly)

- **Shard a single table by range or hashing.**
- **Mongos will direct the INSERT to target shard.**
- **Mongos tries to eliminate shards for update, delete, selects as well.**
- **FIND (SELECT) can happen ONLY a SINGLE table.**
- **Mongos works as coordinator for multi-node ops.**
- **Once a row is inserted to a shard, it remains there despite any key update.**
- **No transactional support on multi-node updates.**
  - Each document update is unto its own.

Informix Sharding



Informix Sharding + High Availability



Transactions (single node)

- **Mongo does not have the notion of transactions.**
  - Each document update is atomic, but not the app statement

- **For the first release of Informix-NoSQL**
  - By default, **JDBC** listener simply uses **AUTO COMMIT** option
  - Each server operation **INSERT, UPDATE, DELETE, SELECT** will be automatically be committed after each operation.
    - No locks are held across multiple application operations.

Transactions (sharded environment)

- **In sharded environment, mongo runs databases via two different instances: mongos and mongod.**
  - **Mongos** simply redirects operations to the relevant **mongod.**
  - No statement level transactional support.

- **Informix**
  - Does not have the 2-layer architecture
  - The server the application connected to becomes the transaction coordinator
  - Does have the 2-phase commit transaction support
    - Unused for NoSQL presently.
  - **INSERT, UPDATE, DELETE** direct thru Enterprise Replication as always.

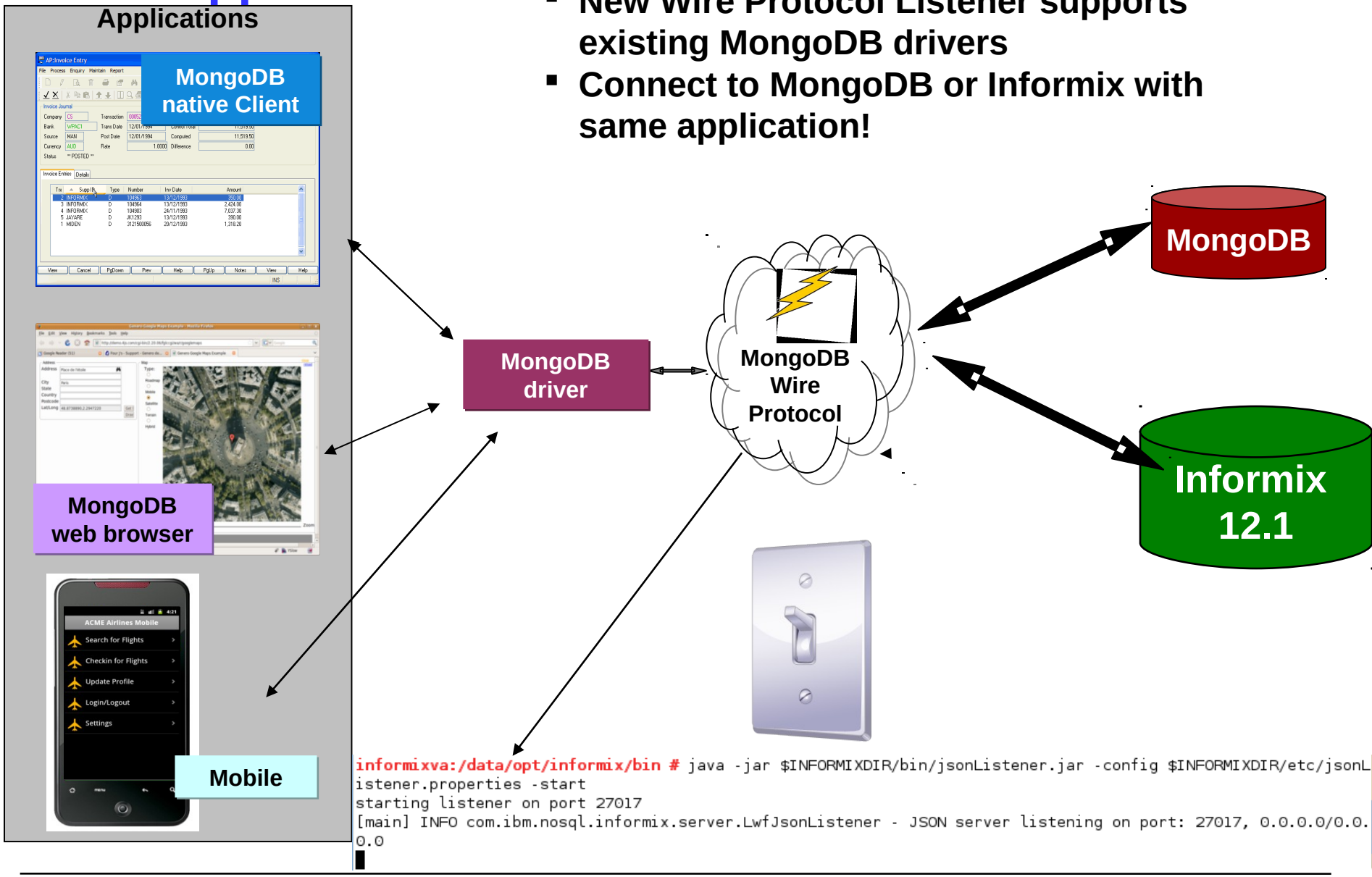
MongoDB Application Source Drivers & Informix
Where new business for IBM is likely from

MongoDB Drivers – The Power of Hybrid

- **One of the big secrets here is what the MongoDB company, formerly known as 10Gen, does not have to do.**
- **Lots of user community support for the product and its drivers.**
- **With the new MongoDB client interface to Informix as a hybrid data store, there are a series of MongoDB officially supported drivers, They are formally listed as the MongoDB "ecosystem" and linked to on the website for MongoDB.**
- **These have support from MongoDB directly as a corporation and also broad support from the MongoDB user community**
- **And now, those users can be part of the Informix "ecosystem" as well.**
- **A whole new universe:**
  - to go, where Informix has not gone before.

Client Applications

- New Wire Protocol Listener supports existing MongoDB drivers
- Connect to MongoDB or Informix with same application!



MongoDB Drivers – Officially Supported (1)

- **With the new MongoDB client interface, with Informix as a hybrid data store, there comes a whole new world of drivers necessary for programmers to access and interface Informix, thru mongo:**

- **C**
  - C Driver Library
- **C++**
  - C++ Driver Library
  - Download and Compile C++ Driver
  - Getting Started with the C++ Driver
  - SQL to mongo Shell to C++
  - C++ BSON Helper Functions
  - C++ BSON Array Examples
- **C#**
  - CSharp Language Center
  - CSharp Community Projects
  - Getting Started with the CSharp Driver
  - CSharp Driver LINQ Tutorial
  - Serialize Documents with the CSharp Driver
  - CSharp Driver Tutorial
- **Erlang**
  - Erlang Language Center

MongoDB Drivers – Officially Supported (2)

- **Java**
  - Java Language Center
  - Java Types
  - Java Driver Concurrency
  - Java Driver Replica Set Semantics
  - Getting Started with Java Driver
  - Java Driver and Aggregation Framework
  - Java DBObject to Perform Saves
- **JavaScript**
  - JavaScript Language Center
- **Node.js**
  - Node.js Language Center
- **Perl**
  - Perl Language Center
  - Contribute to the Perl Driver
- **PHP**
  - PHP Language Center
  - PHP Libraries, Frameworks and Tools
- **Python**
  - Python Language Center
  - Write a Tumblelog Application with Flask and MongoEngine
- **Ruby**
  - Ruby Language Center
  - Ruby External Resources
  - MongoDB Data Modeling and Rails
  - Getting Started with Rails
  - Getting Started with Rails 3
- **Scala**
  - Scala Language Center

MongoDB Drivers – Community Supported (1)

- **There is a large and vast MongoDB user community for the support and maintenance of various drivers used in interfacing MongoDB not officially supported by MongoDB.**

- **They are programmers or dba's controlling a database in many cases and are interested in getting solutions to complex programming and database technical issues.**

- **They experiment and try out new technologies, as such they represent a new frontier for Informix. They support the drivers they use thru the standard modern day Internet forums, chat rooms, code repositories, etc.**
  - These ecosystems outside of the MongoDB official environment are testament to the large user community following it has developed.
  - In part, its why they presently have 25% market share in the document store aspect of the database marketplace.
  - The drivers are numerous …… worth looking at briefly.
  - Perhaps one of your customers is using one of these …….

MongoDB Drivers – Community Supported (2)

- ActionScript3
  - http://www.mongoas3.com
- C
  - libmongo-client
- *C# and .NET* Clojure
  - See the MongoDB *Java Language Center*
- ColdFusion
  - Blog post: **Part 1 | Part 2 | Part 3**
  - http://github.com/virtix/cfmongodb/tree/0.9
  - http://mongocfc.riaforge.org/
- D
  - Port of the MongoDB C Driver for D
  - Dart
- Delphi
  - Full featured Delphi interface to MongoDB built on top of the *mongodb.org* supported *C driver*
  - Pebongo
    - Early stage Delphi driver for MongoDB
  - TMongoWire
    - Maps all the VarTypes of OleVariant to the BSON types, implements IPersistStream for (de)serialization, and uses TTcpClient for networking
- Entity
  - entity driver for mongodb on Google Code, included within the standard Entity Library

MongoDB Drivers – Community Supported (3)

- **Erlang**
  - emongo - An Erlang MongoDB driver that emphasizes speed and stability. “The most emo of drivers.”
  - Erlmongo - an almost complete MongoDB driver implementation in Erlang
- **Factor**
- **Fantom**
- **F#**
- **Go**
  - gomongo
  - go-mongo
  - Mgo
- **Groovy**
  - gmongo
    - Also see the *Java Language Center*
  - Blog Post:
    - Groovy on Grails in the land of MongoDB
  - Grails Bates:
    - Grails business audit trails plugin
- ***Javascript***
- **Lisp**
  - https://github.com/fons/cl-mongo
- **Lua**
  - LuaMongo on Google Code
  - LuaMongo fork on Github
- **MatLab**
  - mongo-matlab-driver
- ***node.js***

MongoDB Drivers – Community Supported (4)

- **Objective C**
  - NuMongoDB
  - ObjCMongoDB
- **OCaml**
  - Mongo.ml
- **PHP**
  - Asynchronous PHP driver using l ibevent|
- **PowerShell**
  - mosh Powershell provider for MongoDB
  - mdbc module cmdlets using the o fficially supported .NET driver
  - Doug Finke's blog post on using the original community C# driv er with PowerShell
- **Prolog**
- **Python**
  - MongoEngine
  - MongoKit
  - Django-nonrel
  - Django-mongodb
  - Django-mongonaut
- **R**
  - rmongodb
    - Full featured R interface to MongoDB built on top of the mongodb.org supported ***C driver***
  - **RMongo for R client**
    - R client to interface with MongoDB
- **Racket (PLT Scheme)**
  - Docs
- **REST**

MongoDB Drivers – Community Supported (5)

- **Ruby**
  - MongoMapper
  - rmongo Ruby driver
    - An event-machine-based Ruby driver for MongoDB
  - jmongo
    - A thin ruby wrapper around the mongo-java-driver for vastly better jruby performance.
  - em-mongo
    - EventMachine MongoDB Driver (based off of RMongo).
- **Scala**
  - *Java Language Center*
- **Smalltalk**
  - Squeaksource Mongotalk
  - Dolphin Smalltalk

Nothing Like the Present …….

- **In addition to connectivity thru the wire listener, we are offering meaningful enhancements to Informix for JSON capabilities:**
  - Native JSON and BSON data storage types
    - With functions necessary to manipulate the data as well, in and out.
  - Hybrid Applications
    - Execute SQL within JSON apps
    - Execute JSON within SQL based apps.
  - Hybrid database
    - Store Relational and non-relational data within the same database
    - Join both data organization types within SQL
  - Standard MongoDB application performance scaling techniques via sharding
    - We do it too, and have for years.
    - We call it Enterprise Replication, and now we store on a node (shard) by key value and load balance the storage across the shards as does MongoDB for scalability.
  - Use standard Informix utilities and techniques on both.
    - Backup/restore
    - Enterprise Replication
    - Compression
    - Time Series,
    - Etc.

Summary: What is JSON's Role in the Enterprise?

- **Flexible schema is agile, liberating for application developers**
- **But will we abandon years of expertise in data modeling/normalization theory?**
  - How to maintain control in an enterprise, mission critical DBMS?
- **Identification of appropriate applications is critical**
- **Application deployment procedures need to adapt**
  - New controls to prevent schema chaos
  - Application development groups need to implement controls
- **When combining with application that uses relational schema**
  - Identify portions that need to remain dynamic
  - Allocate/accommodate space for that as JSON
  - Future – combination of SQL and JSON will make this easier

What Data Store Format to Use?

- **Consider NoSQL JSON when**
  - Application and schema subject to frequent changes
  - Prototyping, early stages of application development
  - De-normalized data has advantages
    - Entity/document is in the form you want to save
      - Read efficiency – return in one fetch without sorting, grouping, or ORM mapping
  - "Systems of Engagement"
    - Less stringent "CAP" requirements in favor of speed
      - Eventual consistency is good enough
    - Social media

- **Relational still best suited when these are critical**
  - Data normalization to
    - Eliminate redundancy
    - Ensure master data consistency
  - Database enforced constraints
  - Database-server JOINs on secondary indexes

Data Normalization - Choose the Right Solution

```
create table department (
    dept char(3),
    deptname varchar(40),
    manager varchar(20),
    empno integer,
    projno integer);

create table employee(
    empno integer
    lastname varchar(20)
    edlevel smallint );

create table project(
    projno integer,
    projname varchar(20),
    respemp  integer);
```

Unique indexes on department.dept, employee.empno, project.projno
Composite index on department (dept, empno, projno)

Constraints (minimally) :

Between department.empno and employee.empno,
Between department .projno and project.projno

- **Possibility exists of mistyped data with the NoSQL schema.**
- **Application joins a must**

- **No mismatched data types on Informix.**
- **No Application Joins.**

NoSQL JSON - Two approaches

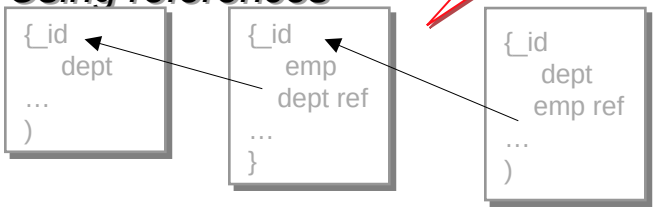
Embedded (de-normalized)

```
{dept: "A10",
 deptname:"Shipping",
 manager:"Samuel",
 emp:[
        {empno:"000999",
          lastname:"Harrison",
          edlevel:"16"},
        {empno:"370001",
          lastname:"Davis",
          edlevel:"12"}
      ]
 proj:[
        {projno:"397",
          projname:"Site Renovation",
          respemp:"370001" },
        {projno:"397",
          projname:"Site Renovation",
          respemp:"370001"}   …
      ]
}
```

Chance for data redundancy

Requires application-side join

Using references



If you need normalization and database-enforced constraints, JSON may not be best choice

Using Schema-less Model In a Traditional Environment

- **During prototyping, early development, validation**
  - As system stabilizes, stable schema parts could be moved to relational columns
- **For cases where web message or document will be retrieved as-is**
  - Yet retrieval on internal fields is needed
- **When parts of the schema will always be dynamic**

JSON and Informix – Complementary Technologies

- **Does NoSQL mean NoDBA? NoInformix?**
  - Definitely not - the relational database isn't going away anytime soon
  - IBM sees JSON as becoming a complementary technology to relational
- **Transactional atomicity is essential for mission critical business transactions**
  - Informix JSON solution brings commits, transaction scope
- **Informix was the first to store relational and non-relational datatypes side by side 16+ years ago and allow hybrid apps.**
- **Choice for developers.**

JSON and Informix – Complementary Technologies

- **What do we offer a developer – Think Hybrid:**
  - Hybrid database,
  - Hybrid applications,
  - Multi and single statement transactions,
  - Partial transactions,
  - Compression,
  - **Free** backup/restore utilities (with or without a free provided storage manager to disk, tape or remote storage),
  - Sharding,
  - Scale in/out,
  - Replication,
  - Shared disk,
  - Connection manager,
  - User Definable Data Types
  - Spatial, Time Series, Basic Text Search
  - Full JSON/BSON support natively
  - Configurable Autonomics
  - Free Informix Developer Edition

- **We are the hybrid applications solution … again. Right now.**

JSON and Informix – User Experience Quotes

- **“This is just the beginning of the possibilities for this hybrid development structure. I believe this structure will be more common in the future of app creation than simply choosing only SQL or only NoSQL. The more tools are given to developers, the more creative they can be and more choices they have for tackling their problems. That is what developers want, reliable and varied tools to solve their problems. Informix is working hard to provide that with the JSON Listener. This technology will help drive the next generation of web based applications in an industry where time is precious, rapid prototyping is preferred and scalability is key. With NoSQL capabilities infused to IDS, Informix is off to a great start.” –**
  Hector Avala – Software Developer and College Student