# Escaping the Database Doldrums

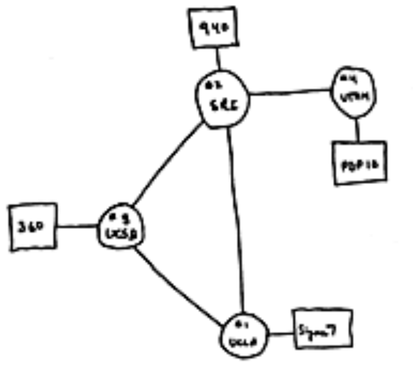
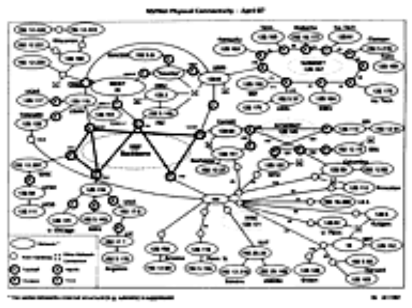
## toward a True RDBMS via free software and IETF methodologies

**James K. Lowden**
**Maintainer, FreeTDS**

## Uncle Codd wants *you*

- Relational technology worthy & interesting
- Proprietary vendors never tried DTRT
- Murray Hill can teach Armonk a thing or two
- Time is ripe for better RDBMSs

# Parallel Universes

| Year | Berkeley | Armonk |
|---|---|---|
| 1970 |  | A Relational Model of Data for Large Shared Data Banks |
| 1988 |  | SQL an International Standard |

## Relational technology worthy

- Many “Database” fads, *Relational* not one
    - VSAM, IMS, OO, XML, NoSQL
    - SQL still dominating after 40 years
    - *Relational* only a theory (*but at least it's a theory!*)
- *Kaching!* Red Hat worth $8 billion, but
    - Oracle $150 billion
    - *Sybase* $6 billion (to SAP on 27 July 2010)
    - even puny *MySQL* $1 billion
- Industry spends $20 billion every year [pdf]
    - Rumors of demise *greatly* exaggerated

## Relational technology interesting

- RDBMSs touch many aspects of Computer Science
    - Relational Theory, UI design, API design, language design, networking, queuing theory, transactions, security, object design, memory mangement, cache management, query optimization, *this space available*
- Many problems unsolved
    - Query optimization
    - `<dbio.h>` and `<db>`
- Many problems actively created
    - Proprietary protocols and language features
    - “Improvements” & “Extensions” to relational model

## Why Do *You* Care?

- Interesting Technical Challenges
    - Language Design & Relational Theory
    - Protocol Design
    - API Design
- Work with Better Tools
- Better Employment Opportunities
- Good of Mankind
- World Domination

[any material that should appear in print but not on the slide]

## Hacking for Fun and Profit

- A free, truly *Relational* RDBMS would be enormously powerful in the hands of an expert.
- A truly Relational RDBMS would make an expert in any one of these areas — Relational, Protocol, or API — likewise valuable.
- “Without the source code, you are up the proverbial tata without a tutu” (Joel On Software)
- “Money follows where value leads” (Eric S. Raymond)

[any material that should appear in print but not on the slide]

## RDBMS: Honored in the Breach

### Vendors Have Not Delivered

- No Relational Query Language
- No Scientific or Graphing Support
- No Standard Wire Protocol
- No Standard Library
- No Progress
- Much Distraction

## The Market Is an Idiot

The customers' cost is the vendors' gain!

- Proprietary languages and libraries create and maintain noninteroperability. Interoperability would commoditize what is currently proprietary.
- Vendors benefit from buyers' ignorance.
- Engineering — language, protocol, API — is work, and not in the vendors' interest.

## Relational Algebra is your friend

**Read**

- Union
- Intersection
- Difference
- Cartesian Product
- Select (a/k/a Restrict)
- Project
- Join

**Write**

- Insert
- Modify
- Delete

## Relational Good, SQL bad

SQL Not Orthogonal

| | constructor | compare | assign | selector | gen expr |
|---|---|---|---|---|---|
| **table** | no | no | only via INSERT - SELECT | yes | no |
| **column** | only as arg to IN | no | no | yes | no |
| **row** | only in INSERT & UPDATE | no | only to/from set of host scalars | (yes) | no |
| **scalar** | N/A | yes | only to/from host scalar | (yes) | no |

Credit: Chris Date A Critique of the SQL Database Language [pdf] December 1983

## Much money, no progress

- Commercial SQL DBMSs cannot
  - Compare relations for equality
  - Define keys for views
  - Optimize queries because language not mathematical
- Poor type support, e.g. cannot express
  - OK: price * quantity
  - Error: price - quantity
  - Error: ID# arithmetic
  - Incompatible ID joins

## Column names matter

SQL

```
SELECT DISTINCT E#, TOTAL_PAY
FROM ( SELECT E#, SALARY + BONUS AS TOTAL_PAY
FROM EMP ) AS TEETH_GNASHER
WHERE TOTAL_PAY >= 5000
```

Tutorial D

```
( ( EXTEND EMP ADD SALARY+BONUS AS TOTAL_PAY )
WHERE TOTAL_PAY >= 5000 ) { E#, TOTAL_PAY }
```

credit: The Importance of Column Names by Hugh Darwen

## Free RDBMS Errors

- Sticking with SQL
- Acting Proprietary in a Free World
  - No attempt at shared protocol or code
  - No thought of common client API
- No adaptation of Good Things from UNIX

# Free RDBMS Opportunites

- Ignore SQL “Standards”
- Embrace Language Based on *Relational Theory*
- Support Interactive Graphing and Linear Algebra
- Adopt One Protocol
- Adopt One API
- Embrace UNIX Ideas (pipelines, namespaces)
- Profit!

## Embrace Relational Theory

- Need a Language with explicit Relational Operators
- More Precise, Less Verbose
- Query Optimization actually possible!
- Rel implements *Tutorial D* by Date and Darwen, cf. The Third Manifesto
- Ingres Implements QUEL [pdf]

## Support scientific computing

- Use SQL to invoke statistical functions
- Statistics are *much data in* and *few data out*. Putting statistics *inside* the server reduces bandwidth requirements.
- No need to install/configure statistical software on clients
- Potential to cache results that would ordinarily be computed on different clients
- Use X client to draw graphs, imagine:
  ```
  SELECT … | ./graph_this
  ```
- Cf. Scenarios for Using R within a [RDBMS]
  *Many ideas above credited to this paper*, and
  Plotting with PL/R on *NIX — A HOWTO

# Adopt One Wire Protocol

- IETF, anyone?
- Any web browser connects to any web server, but
  - MySQL, Firebird, Postgres, Ingres, Rel, MonetDB, SQLite, SAP MaxDB all have their own protocol.
  - Most protocols not documented
- Exception: FreeTDS (but no free server!)
- `ftp` might be a good model
- (Binary protocol required for data fidelity and speed)

## Adopt One API

**RDBMS** — **Bind Function**

ODBC (per column)

```
SQLRETURN SQLBindCol( STMT Handle, int col, int type, BYTE *buf, int len,
int *indicator);
```

Ingres

```
II_VOID IIapi_getDescriptor (IIAPI_GETDESCRPARM *getDescrParm);
typedef struct _IIAPI_GETDESCRPARM
{
 IIAPI_GENPARM gd_genParm;
 II_PTR   gd_stmtHandle;
 II_LONG  gd_descriptorCount;
 IIAPI_DESCRIPTOR *gd_descriptor;
} IIAPI_GETDESCRPARM;
```

SQLite

```
int sqlite3_bind_blob(sqlite3_stmt*, int, const void*, int n, void(*)(void*));
int sqlite3_bind_double(sqlite3_stmt*, int, double);
int sqlite3_bind_int(sqlite3_stmt*, int, int);
int sqlite3_bind_int64(sqlite3_stmt*, int, sqlite3_int64);
int sqlite3_bind_null(sqlite3_stmt*, int);
int sqlite3_bind_text(sqlite3_stmt*, int, const char*, int n, void(*)(void*));
int sqlite3_bind_text16(sqlite3_stmt*, int, const void*, int, void(*)(void*));
int sqlite3_bind_value(sqlite3_stmt*, int, const sqlite3_value*);
int sqlite3_bind_zeroblob(sqlite3_stmt*, int, int n);
```

MySQL (strings only)

```
MYSQL_ROW mysql_fetch_row(MYSQL_RES *result);
```

Postgres (string or unconverted)

```
char *PQgetvalue(const PGresult *res, int row, int col);
int PQgetisnull(const PGresult *res, int row, int col);
```

## ODBC Is Not the Answer

- Works around proprietary protocol problem
- Standard in Name Only
- Cumbersome, not UNIX-y
- Many 64-bit problems
- Notoriously vague error handling

## Use `stdio` as a Model

- `DB * dbopen(const char *url, const char *options)`
- `size_t dbwrite(void * restrict sql, size_t size, DB * restrict stream)`
- `int dbscanf(DB * restrict stream, const char * restrict format, ...)`
- `int dbprintf(DB * restrict stream, const char * restrict format, ...)`
- `int dbclose(DB* stream)`

## Hello Armonk, this is Berkeley

- Adopting UNIX conventions would make existing skills more versatile
- `server.dbname.schema.object.column` comes straight from IBM
- Why not `/server/dbname/dir/.../object` ?
- `use` ***`dbname`*** could be `cd` ***`dbname`***
- `$EDITOR` , `cat` , `more` , `chmod` , `chown` , `chgrp` , and `rm` also good models

## We are not alone

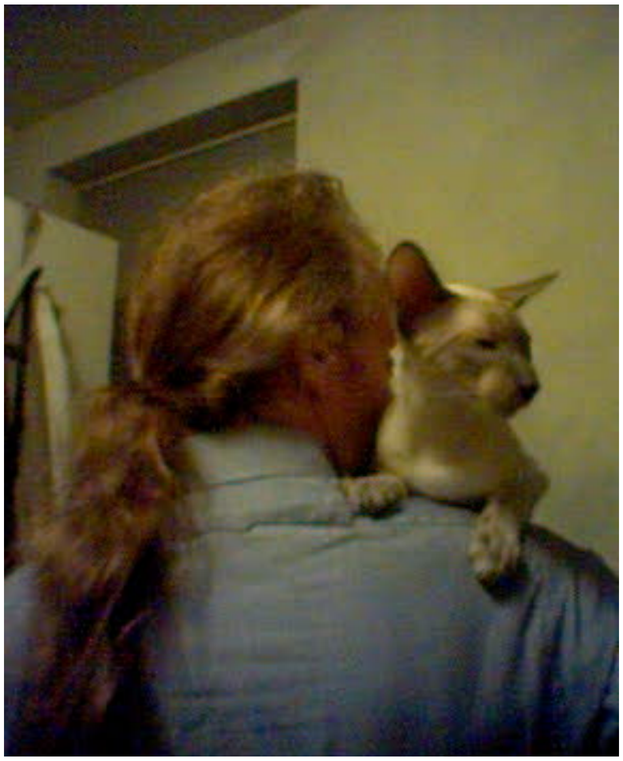
- The Third Manifesto describes *Tutorial D*, a relational language
- Rel implements *Tutorial D*
- Dee implements *Tutorial D* in Python(!)
- Ingres Project D implements D on mature server technology

## Uncle Codd *needs* you

- A *Relational* language would put power (instead of putty) in the hands of the programmer
- Free RDBMSs would benefit (and profit!) from a shared protocol and API
- IETF and UNIX show how to develop working protocols and shared API code
- Better tools are more productive and less frustrating
- The market will reward the experts

## Next Steps

- Read The Third Manifesto
- Port Ingres to BSD
- Adapt FreeTDS to SQLite
- Write stdio-based library for any free RDBMS
- Nudge your RDBMS to Get Relational
- Further reading at freedb.schemamania.org
- We can always talk jklowden@schemamania.org



[any material that should appear in print but not on the slide]