Tips for success

Common mistakes in application development with Firebird and how to avoid them

Pavel Císař
IBPhoenix

Firebird Conference 2011

Agenda

- Recipes for disaster
- Mistakes in database design
- Wrong approaches to handle data

Recipes for disaster

- “Agile Knowledge Management”
- Wrong assumptions
- Unjustified trust to SW, HW, tools etc.
- “Thinking only up to the API”
- Preferring the way of least resistance
- No prototyping / No testing

Solution

- Think, don't just copy!
- If it walks as duck, squeaks as duck, it could be a beaver in disguise!
- Don't trust strangers!
- ...especially those who offer you candy!
- You don't know until you verify it, so what you're waiting for?

Database Design

- Artificial vs. Natural primary keys
- VARCHAR vs. BLOB
- Character sets
- Security

Artificial vs. Natural keys

- Artificial keys are always better (at the end)
- ...but there's hidden bloody price...
- ...the JOIN HELL
  - Users want to see anything else but keys
  - Users use anything but keys for search
  - You have to join tables to get in important columns even it wouldn't be necessary
  - You have to add more indices

Solution

- Use natural keys for most used “leaf” tables
- Cache the lookup tables in application
  - Full set for small ones
  - MRU for big ones

VARCHAR & BLOB – The Wrong

- People decide by feel, not reason
- Focus on values, not how they're used
- Oversize just to be safe

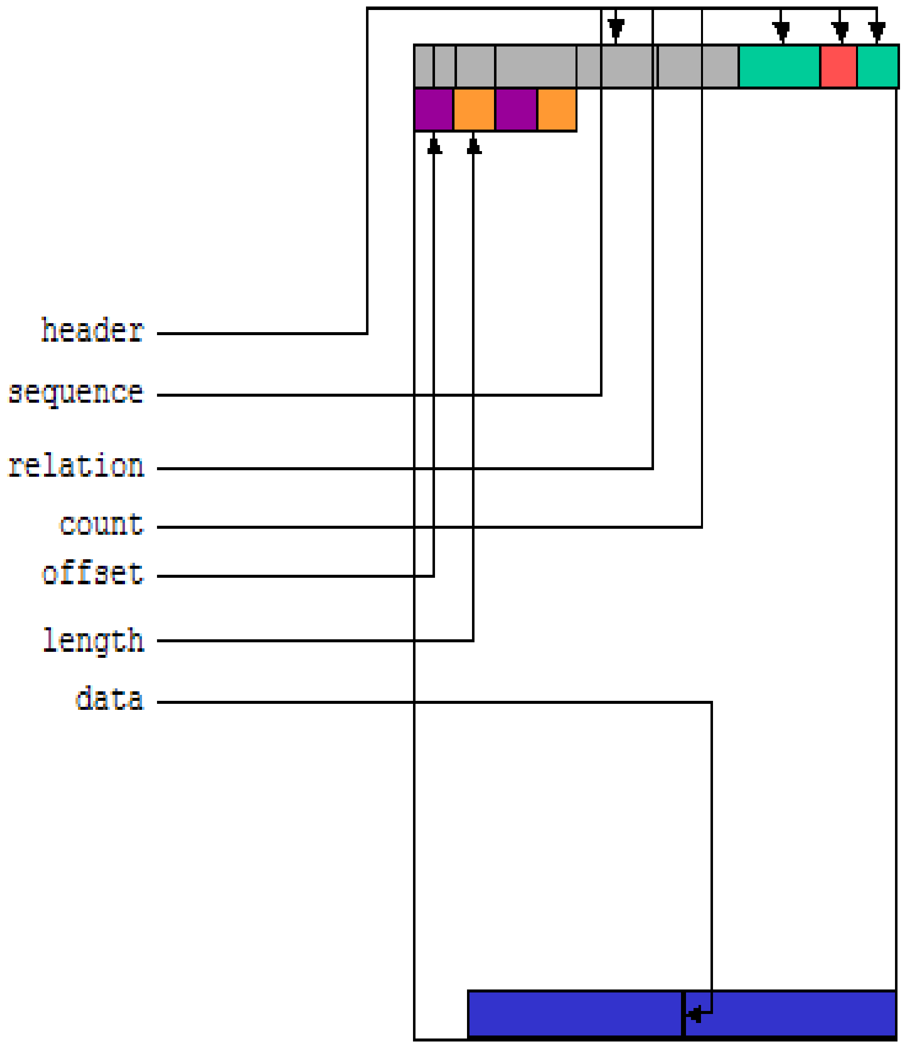
VARCHAR vs. BLOB - Solution

- There is a great guide from Ivan Přenosil http://www.volny.cz/iprenosil/interbase/ip_ib_strings.htm
- *Quick* tips:
  - Anything longer than 150 characters is good candidate for BLOB
  - BLOBs are bad news for search and server-side processing
  - Long VARCHARs are bad news for sorts
  - You can combine several long VARCHARs into single BLOB or refactor them out to 1:1 table
  - More than one BLOB per table is bad idea (Refactor BLOBs out to 1:1 table)

Golden Rule of Database design

Get as much rows on single page as you can

- BIGINT x INTEGER x SMALLINT
- DATE x TIMESTAMP
- Non-empty BLOBs are rows too!
- Pages smaller than 8K are pointless



Character sets

Not that big problem today,
But still some developers are careless
or blindly default to UTF-8

Character sets - Solution

- Always use one!
- Consider conversions!
  - What is native data type/encoding for strings your application uses?
  - Minimize number and complexity of conversions:
    Database <-> Application <-> Input/Output

Security – The problem

Either
NO security at all
or
Extreme security measures

Security - Solution

- Use as little from SQL security features as you can
- Use remote server with restricted access
- Your application is exclusive gateway to the database
- Use OWNER account (block SYSDBA if you want)
- Implement fine-grained security in your application

Handling data – The wrong

- “When I did this in ISQL, Flamerobin, my other app etc., it worked just fine, so what's wrong now?”
- “It worked just fine on my development machine, so why it fails in production?”
- “Damn, it doesn't scale as I expected...”

Handling data 101

- Get intimate with your connectivity library!
- Always manage your transactions manually!
- Always mind the MGA!
- Never fetch more data than you actually need!
- One size doesn't fit all:
  - Interactive vs. Machine processing
  - Native vs. Web applications
  - Embedded vs. Department vs. Corporate

Know your connectivity

- Identify higher and lower level access paths
- Learn the steps the access paths use
  - Path complexity
  - Data conversions
  - Storage requirements
  - Algorithm efficiency
  - Points of failure

Transactions

- Interactive
  - All data read in single transaction READ_ONLY READ_COMMITTED
  - Writes in separate R/W transaction
- Machine processing
  - R/W in single SNAPSHOT or READ_COMMITTED
  - No UNDO log
- Long running “monitoring” transactions
  - READ_ONLY READ_COMMITTED

MGA Implications

- Long running transactions block GC
- Inserts never block other users, update and delete may block
- Changes create garbage
- Correlating inserts with update/delete is VERY bad idea
- Mass update/delete burdens you with GC

Minimizing data transfers

- Interactive
  - Less rows
  - Less columns, fetch the rest on demand
  - Cache lookups on client
- Machine processing
  - Do as much on the server as you can

Beware the “One code to rule them all”

- Except the simplest cases, there is direct proportionality between universality and direct cost
- Universal code that doesn't have an option to cop-out on special case to user code is recipe for disaster

Questions?

Pavel Císař
pcisar@ibphoenix.cz
IBPhoenix
www.ibphoenix.com