Iterative Database Design
Challenges and Solutions for a Geomechanics Database

By James Ding and Cary Purdy
Presented at Petroleum Network Education Conference (PNEC) 2011
Abstract
Pressworks™ is an award winning, relational database designed for the purpose of storing and managing geomechanics and pressure-related data. The design and development of this database must accommodate and support geophysical data, petrophysical data and drilling data and related workflows. The database is tightly integrated with the Drillworks™ geomechanics and pore pressure prediction application.

Over the life of such a database it is challenging to anticipate future data requirements, emerging data formats, the breadth of possible new cross-domain workflows, or real-time communication demands. This paper provides a retrospective view of what was learned over the past eight years, and how we plan to leverage these lessons in future Pressworks development. We also explore how these database design principles might be applied to the development of new relational databases supporting the digital oil field of the future.

Iterative and database development best practices are suggested, which include, for example, a methodology for making incremental changes to the database design, making the database easier to understand and easier to modify, evolving your design via iterative steps, and general recommendations for improving database structure to enhance performance and ensure data quality, usability and integrity.

Using iterative development principles the Pressworks database can more effectively respond to changing needs and an inevitable evolution of data types. The iterative, incremental design practices used to develop the Pressworks database resulted in a robust geomechanics database capable of bridging the gap between geoscience, reservoir engineering and drilling workflows by allowing collaboration through the use of each other’s analyses, data, models, formation parameters, etc., thus improving well planning and drilling operations.

Introduction
Pressworks™, an award-winning geomechanics database, was designed and implemented over the last eight years. This paper provides a retrospective of what was learned and how to apply these lessons to the future development of Pressworks specifically, but also how these principles might be applied to the development of other relational databases. Iterative and incremental database development best practices are suggested, which include, for example, a methodology for making incremental changes to the database design, making the database easier to understand and easier to modify, evolving your design via iterative steps, and general recommendations for improving database structure to enhance performance and ensure data quality. Also included is a description of how the database was real time enabled as well as a discussion on how connectors to OpenWorks, EDM and some 3rd party databases were implemented so that all data appears to the user to be federated in a single data repository.
Background

Drillworks™ is an integrated pore pressure and geomechanics application that allows operators worldwide to drill deeper, faster, and safer with fewer surprises. Pressworks™ is a relational database designed to work with the Drillworks application for the purpose of storing and managing pressure-related data including geophysical data, petrophysical data and drilling data. The technology combines new Drillworks and Pressworks software to deliver an integrated pore pressure and geomechanics solution. The technology is enhanced through the connection to DecisionSpace Desktop where data and results from geomechanics are combined with the data and earth model from geology, geophysics and well planning modules.

Pressworks has an intuitive data browser and visualizer that allows engineers, geoscientists, and managers to quickly and easily find, select, and view wells and pressure-related data in a customizable form that is meaningful to each individual user. The interface also has an option to search and query via a map display powered by ESRI GIS. Connectors are available which federate OpenWorks, EDM and some 3rd party databases providing streamlined and seamless access to well logs, seismic data, well and wellbore operations data, etc.

Through their respective interfaces, Drillworks and Pressworks enhances team collaboration by allowing asset team members and management to view, comment and contribute towards completing pore pressure and wellbore stability plans.

Both Drillworks and Pressworks are real time enabled through a WITSML compliant transmission application called ConnectML. Using ConnectML, operators can now stream real time data directly into Pressworks allowing wells to be analyzed with remote support by experts in real time centers. Pressworks helps bridge the gap between geosciences, reservoir engineering and drilling by allowing collaboration through the use of each other’s analyses, data, models, formation parameters, etc., thus improving well planning and drilling operations.

The Process

Over the life of such a database it is challenging to anticipate future data requirements, emerging data formats, the breadth of possible new cross-domain workflows, real-time communication demands, and emerging requirement to connect to multiple databases with seamless access, as well as client specific multi-user, simultaneous access to data.

Under most circumstances, we don’t have a fully formed plan to anticipate all these uncertainties up front. So it is unrealistic to expect that we can build the Database all at once, or even solely rely on incremental design to build the Database just like building a car, one part at a time, to reach a final version in one release. So designing a database early on, and sticking with it by forcing development team following a Change Control Process won’t work because clients are demanding new features and improvements on existing features all the time.
We can’t expect an up front design, or even subsequent commercial releases/updates to anticipate all future data requirements or emerging data formats. Therefore we need to embrace approaches where the designs evolve throughout the life of the product. Incremental and Iterative Database Designs let us move from idea to full implementation, without a fully formed plan up front about every detail. Incremental designing is about building up a footprint to support more application functionality. Iterative Design is about improving database design. This paper’s main focus is on iterative design as it gives us a formal process for inspection, getting feedback, measuring progress, and adjusting, to iteratively reach the desired detail and completion level.

This is achieved through:

- Improving design of the database
- Making database easier to understand and modify
- Enhancing performance
- Ensuring data quality and security

To take full advantage of this methodology, we will need a shift in our mind set. In the past, we took comfort in believing we had all the details during the planning cycle, thus making it relatively easy to come up with tasks and estimation of effort, etc., before the project was started.

Here are some mental adjustments that will be required for moving to the new methodology:

- Don’t try to offer a complete implementation too soon
- Prioritize what you want
- Add detail and fine tuning implementation iteration by iteration, release by release
- Build up quality along the way
- The best way to deal with uncertainty is to iterate

We will need database design changes and a formal process through each cycle of iterative design in order to ensure its success. The process we recommend for design change is as follows:

1. Confirm the requirement for Design change
2. Select the most appropriate change
3. Write test cases
4. Decide on data migration strategy
5. Implement Database change
6. Script for both new Database install and upgrade existing Database
7. Testing and documentation
8. Deployment
In the case studies, we will discuss why we need a particular database design change, our development and deployment process, and customer data migration. We will cover the following case studies:

Case 1:
Add lookup table (for lithology to improve data consistency and referential integrity, to provide user with code lookup instead of free text input)

Development and Deployment Process:
- Create Lithology Table
- Add LithologyID column to LithologyColumn table
- Get lookup data and put them in table
- Add referential constraint
- Update application code

Customer Data Migration:
- Make sure data in LithologyColumn have corresponding value in Lithology
- Update LithologyColumn set LithogyID = ## where LithologyName = 'yyy'

Case 2:
Add Index (Unique Well Identifier, or UWI, to make it easier to find all related well data from different sources and/or vendors and to enable client companies to match wells in Pressworks against their own data repository)

Development and Deployment Process:
- Add a new index on UWI (Unique Well ID)
- Update application to build query based on the uniqueness of Well Identifier

Customer Data Migration:
- Select Count(*), UWI From Well group by UWI having count > 1
- Export duplicates
- Merge back into database

Case 3:
Split table (for cube data to improve performance since application reads cube attributes more often than cube data, which is much more expensive to load)

Development and Deployment Process:
- Create CubeData table
- Update application code accordingly

Customer Data Migration:
- Copy CubeData from the original table to the new table
Case 4:
Replace one-to-many relationship between Project and Well with a mapping table, and migrate it into a many-to-many association (for individual well to be used in different projects)

Development and Deployment Process:
- Add a mapping table ProjectWell
- Remove referential constraint based on ProjectID and deprecate the ProjectID column in Well Table
- Add index on ProjectWell table (ProjectID, WellID)
- Update application to allow well’s association to multiple projects

Customer Data Migration:
- Populate mapping table (ProjectWell) with existing one-to-many association
- Sync with trigger (for transition period)

This paper provides a retrospective view of what was learned over the past eight years, and how we plan to leverage these lessons in future Pressworks development. We also explore how these database design principles might be applied to the development of new relational databases supporting the digital oil field of the future.