GEMKO Information Group, Inc.

Information Systems Specialists

GEMKO Application Modernization Journey

Data Strengthening - A Technical Perspective

It's all about the data:

The key to your application is your data. It is where the entire history of your business is documented, where most of the lessons are learned and where most of the strategic decisions will be made. Having data that is flexible and that can be shared with applications running on other systems, is mission critical.

Unfortunately, most legacy databases were designed at a time when either disk space or processing power was at a financial premium. In order to accommodate the physical constraints, we unwittingly created "design" constraints. Hence our data files were replete with redundant data, buckets, complex keys, wasted space and unfriendly data types. Horizontal data was king, with record lengths sometimes in the thousands. This resulted in constant program maintenance, painstaking file format changes and frequent inconsistencies, which ultimately contributed to the platform's dinosaur legacy.

GEMKO's approach to data strengthening is predicated on the premise that the more work your data can do for you, the less must be done by your application. The less required by your application, the less resource will be required to maintain it, and thus the more ROI your organization will get from it. Some of the techniques GEMKO uses to strengthen data include:

- Eliminating database file DDS in favor of strictly SQL object development.
- Defining distinct data types consistent with your business needs.
- Using the full breadth of data types provided by DB2, including dates, times, timestamps, binary integers and varchars, plus others.
- Using SQL identity columns in place of complex keys.
- Defining referential constraints between files with parent/child relationships.
- Using check constraints and unique constraints to enforce business rules at the database level that were previously enforced via program code.
- Storing data elements once and only once to prevent conflicts.
- Eliminating buckets in favor of user defined functions over detail data to eliminate inconsistencies.
- Adopting a vertical architecture. A scenario involving more files with fewer fields is far more flexible than a single file with many fields. In some cases the number of fields may be reduced through elimination of redundancy.
- Using triggers to replicate legacy data into its strengthened equivalents real time
- Creating standard I/O conduit functions to eliminate "F specs from RPG. This lifts a number of RPG restrictions and sports a much more object oriented look and feel in your application programs, while making them more self-documenting.

In other words, evolving legacy application data into truly relational databases. When the AS/400 was introduced in 1988, one of its selling points was the integrated relational database. We were led to believe that by simply creating files using DDS, we were developing relational databases. Nothing could be further from the truth.

GEMKO Information Group, Inc.

Information Systems Specialists

You don't know what you don't know:

IBM's System/3X and AS/400 environments were so stable, programmers and analysts could quickly find a comfort zone and reside there for many years. Tools such as RPG and CL were like duct tape – they could be used to accomplish nearly anything, so we thought. However, they also helped us develop high maintenance applications fraught with cloning, redundant code and tooling being used where it was not the best fit. During the 1990's and this decade, professionals on other platforms were making great strides in programming efficiency, reusability and portability. Their talent pool and career opportunities were growing. However, the same could not be said for those on the iSeries – duct tape was still the tooling of choice, and their opportunities were shrinking.

GEMKO's 21st Century Midrange Developer training allows programmers to achieve a proficiency level in numerous technologies and methods, many of which are cross platform, that are used systematically strengthen and evolve existing applications. This enables midrange professionals to craft solutions that interact better with other platforms and react to changes in the business climate painlessly.

Tools such as custom data types, database triggers, referential integrity, user defined functions, stored procedures and SQL scripting help shift much of the business logic to the database level, which is cross platform, as opposed to the RPG and CL level, which is proprietary. Better RPG practice, including taking full advantage of ILE architecture and the rich feature set of free form RPG IV, allows applications still using RPG to adopt a much lighter footprint, as well as a look and feel more consistent with other languages. This greatly expands the talent pool qualified to support your applications.

Someone younger will eventually have to take the ball:

We in the midrange environment have always taken great pride in our design and development. Each program, each menu, each report a unique masterpiece in its own right, with our signature upon it. However, despite our best passions, once the responsibility for the application is passed on to the next generation, it will be likely be retired if its architecture and tooling is not consistent with the skill sets of more youthful professionals. With it goes not only the many years of blood, sweat, tears – but also the user proficiency and intellectual property acquired over the life of that application.

GEMKO's education plan and mentoring assists veteran programmers with introducing better programming practice and more open technology to the application. Emphasis begins to gradually shift toward the "big picture", and consequently, developers discover new problem solving concepts that they had never considered before. This happens gently and consistently over time as opposed to an unpredictable "big bang".

It's always important to remember that someone paid your salary and benefits for years in order to allow you to develop these applications. Others paid for the training time spent by end users. Others may have paid for the applications indirectly. They will expect that investment to be protected, but at the same time, evolved and enhanced. Having a strong application core (data) makes that possible.