CS61: Database Systems - Dartmouth College

This is a true story.

Aaron was well-known as a superb algorithms guy who had solved some of the most complicated, messy network design problems there were. He had even applied his skills in other kinds of combinatorial optimization, such as … but making sense out of large scale complicated masses of interdependent data, occasionally of dubious veracity, were his favorite.

So it came as no surprise when the researchers at a neighboring University approached him for help.

They were working on finding drugs to fight certain kinds of cancer. They were primarily chemists and geneticists, and not programmers or very sophisticated computer users. They were trying something new: crowd sourcing their lab data. They were a small informal consortium of like-minded researchers scattered around the world. Everyone knew and respected each other, so they didn't have to worry much about the quality of the contributed data. Having started small, with just one other research group, they had tossed together some simple tools for collecting a member's contributed lab data. They also had built some rudimentary tools for searching the data for matches against some simple templates representing genes and chemical compounds that they had found to have some effect on them. As their work progressed, they needed new ways to search for new templates they wanted to try. Every time they needed a new search, they had to find a grad student somewhere to write special code. But grad students were plentiful across the consortium, especially as it grew larger, so the consortium just continued to focus on their research.

All the members contributed code from time to time, using a growing collection of user interfaces and not always fully debugged. Typically it would work fine for the particular group that had developed it ... the code found what they wanted it to find ... But the code didn't always work as well for others. But no matter, they had their own growing libraries of search code that was more tuned to their needs.

As the consortium had some early successes, the membership grew. Accordingly, the data began pouring in from everywhere. The crowd sourcing of lab data data was working! Soon they had to abandon the two PC's running their project and move up to mid-range servers with terabytes of storage and real backups in the cloud. Soon after that, their membership in Russia and Asia spiked, and so they had some more grad students cobble together a mirror of their data servers in Beijing. Of course, the servers there were fantastic machines from a Chinese manufacturer, but they ran a different operating system from the original ones and some accommodations (and more code) had to be made.

That's when things began to fail.

- There were synchronization problems between the sites.
- The searches were taking longer and longer to run because of the huge amount of data.
- The same search didn't always produce the same result. Data that had been updated in one part of the network sometimes existed in other parts which still had the old values. Similarly, some data that had been deleted by one team still persisted in other parts of the network.
- The vast collection of customized search tools being used by the dozens of researchers would sometimes conflict with each other, resulting in more errors or full system failures.
- As the membership grew, the conflicts over data updates also increased. One researcher's application might be trying to update some shared data while another was trying to read it or even update it differently.
- As the number of researchers grew, the complexities of managing Access Control to the data also grew. There were times when researchers needed to have access to some, but not all, of the data. As the number and type of data files exploded, this became unmanageable.
- Some members lost their graduate students to graduation, and so some of them began to try to use other members' search tools ... inconsistent interfaces resulted in frustration and missed paper deadlines.
- Some members even began withdrawing from the consortium.

That's when they called in Aaron

What followed was weeks of painful discovery. Aaron discovered all of the inconsistencies and poor practices mentioned above, and other problems that even raised doubts about some of the consortium member's published results.

• There was no database, per se, but really just several enormous sequential "flat" files built directly within a system's file system. No consistent software or hardware was used.

• There were different representations for the same kinds of data. Different chemical compounds had their constituent parts listed in different orders, depending on the researcher's academic background.

• There was no way to lock access to a particular set of data during a program's execution even if it was updating the data.

• There were HUGE data redundancies across the project's servers. The same results duplicated many times over, stored and sorted in a variety of ways to suit individual researcher's needs.

• As a result, there were data inconsistencies everywhere. One researcher would create a copy of a data file, make some "what if" changes to the data, and rerun his/her research. If the results were promising, the researcher would continue to work with the updated copy.

If a researcher found an error and corrected it, it was typically only corrected in his/her own copy!

• Sometimes a researcher would find an error in his/her lab work and then want to remove some related data he had contributed. He would simply go to where he had put the data and delete it, unaware of any copies or updates that might have been made to these 100-MByte files.

• The wide variety of programs written to access all this data were of wildly variable quality and didn't always play nice with the other programs. This resulted in system hangs, deadlocks when accessing common data, and all sorts of data integrity problems.

Aaron had never seen such a disaster of a system.

Like many projects, this one was based on a brilliant idea: crowdsourcing of quality lab data from like-minded researchers. It was the execution and scale-up that bit them.

A database system would have saved them a whole lot of trouble. A modern database…

• has a consistent user interface for queries and updates

• can easily handle ad hoc queries

• provides built-in consistency checks that insure the ACID properties of transactions:

- **A**tomicity - either all the effects of a transaction persist after completion, or none of them do.
- **C**onsistency - every transaction leaves the database in a consistent and correct state - i.e., the resulting data doesn't represent a system-defined impossibility (like a negative molecular weight for a compound).
- **I**solation - transactions cannot interfere with one another.
- **D**urability - the result of a successful transaction must persist after the transaction completes, and even across system crashes or other events.

• often provide extremely efficient query optimization capabilities.

• are designed with multi-user and highly distributed systems in mind.

• provide data models at the *logical* level, hiding away the physical implementation underneath, improving portability and reducing errors, while enabling sophisticated optimization techniques "underneath".

This is why we study database systems in CS61 !