Good Morning
9 years ago, in Chicago at LISA X, I gave an invited talk about rather than just managing computer accounts, to trying to expand the project to manage more information about people, and from that drive the computer accounts, ID card systems, library circulation systems, and so on. I also presented a paper on producing and managing the university telephone directories as a problem in systems administration.

Both of these presentations had a very technical focus, explaining why I used a big hammer such as Oracle, and I gave examples and approaches to using a relational database system in solving these problems and other problems in systems administration.
Now 9 years later, I am back to report on how this actually worked out.

Unlike my talks from 9 years ago, although I will touch on some handy bits of technology, I also want to talk about some of the social issues, turf battles, political hot buttons, and other issues you must contend with to make all of this actually work, and some of the approaches I was able to use to get folks willing and in some cases eager to help me!
The problem I was addressing in 96 was finding a way to generate computer accounts, the phone directory and drive the ID card system, based on information from the registrar, human resources and a poorly defined guest category. My point at that time, was that defining that magic process and making it one system could provide some benefit.
The problem we have now, is a lot more complex. We have a lot more data sources (the boxes and the top) and a whole lot more data consumers, the boxes on the bottom. In addition, we had a new challenge that unlike the traditional source (Registrar and HR), these new types did not have elaborate database systems with staff to manage.

In some respects however, this increase in complexity and scope provided some benefit to pulling this together. I will get back to that later.
I first heard a version of this by Ken King, at the time the Vice Provost for Information technology at Cornell who described Cornell as “11 colleges linked by a common steam plant”. While perhaps a bit extreme, it does point out that all of those departments listed on the last slide all have their own problems, and helping me solve my problems just is not on their list of things to do.

The basic problem of providing computer accounts is of interest to us, the IT folks, but it doesn’t really impact the rest of the folks at the institute. But finally an issue came up that grabbed the interest of many people and departments at Rensselaer and gave us the ability to move forward.

I had made some progress along the way, but finally, an issue came to the fore, that everyone could rally around.
That issue is of course, Parking.

This was actually an expansion of our existing card access system (the replacement for the system I described 9 years ago), but this change included a significant number of people from every type of person we had on campus, students, employees and many different kinds of visitors. To get on campus, they had to have a parking transponder. Also, unlike the many different email systems on campus, there was only one access control system, and everyone had to be in it. This got all the critical players to the table. Once I was able to start offering solutions, I was able to get the support I needed to make it all happen.
The Concept

• A “Status” for everyone on campus
• That Status is controlled by the appropriate authority
• The status automatically drives all other systems
  – Including separation.

Everyone on our campus – current folks at least, has at least one status value.

Those status values are controlled by the appropriate authority

Everything else derives from this status.
Selling the Concept

• To Data Providers
  – They are authoritative
    • No Exceptions!
  – Tools available to assist
    • Tools give additional value
• To Data Consumers
  – Resistance is Futile

The first two data providers are the registrar and human resources. At that time, I controlled the computer account process, the campus phone directory and to some extent, the ID Card system, and through that, the parking system. We took a hard line that employees had to come via HR, and students via the registrar. There were some limitation in Banner for employees, so I had to provide another tool to HR. I cut a deal with them – if they would use the tool to mark when new employees were “OK”, we would refuse to issue computer accounts, ID cards, parking access or directory listings until new hire was marked ok by HR. This gave HR the juice they needed to get I9s signed, and ensure at least a moment of face time with every new employee BEFORE they started. As data consumers, we got out of the exception business – we just send the folks to HR (or the registrar).

As smaller data sources come on line – we give them tools and procedures that make it easy for them to work with us – and not only does it make their jobs easier since everything is clearly defined, the tools will give them some value added features like address lists, photos of their people, and the knowledge that one simple task on their part, and their constituents get email, parking, library access, athletic facility access, and everything else in one shot.

For the data consumers – we try to make it very easy for them to get the data they need – that alone sells most of them right away. And the more consumers we have, the more value it provides to the data providers.
You need to identify your primary data sources. In our case, we get all employee data from Human Resources and all student data from the Registrar. If a new student or employee wants their email account, their ID card or their parking transponder, they must be cleared by the registrar or HR as appropriate. A “no exceptions” policy makes life easier for everyone – the process is clearly defined, and when followed, flows very smoothly.

The guest problem is more challenging. We have many different types of guests – but we were able to funnel all of them through our ID card office.
Time to dive into the tech details for a bit

We get a data feed from the Human Resources System and the Registrar. Both of these are Oracle based, and in fact are on the same server (SCT Banner). In addition, we needed a way to handle “guests”, so we developed an Oracle based system to manage guest entries. The ID card office has developed procedures and forms so that the appropriate offices can request that people be added, such as visiting scholars and ROTC cadre from the Provosts office, members of the Lets Go Red club from the athletic office. There are other types of guests who are also entered into the system. The white arrows are direct machine to machine feeds, and the dotted red arrows are essentially paper feeds.

But where does this get us?
These data feeds end up in three different tables, one for employees, one for students and one for the ID guests. They all have names – a good start. Both Employee and Student have a Rensselaer ID number. This used to be the social security number (boo, hiss). Sometimes we actually get a RIN for ID guests for folks who were students or employees, but in other cases we don’t.

But after that, things break down. For employees, we have get a classification, and a department affiliation and in some cases, a termination date. For students we find out their campus, their class year and if they are currently registered, and finally for ID guests, we have a guest type, a sponsor, either a person or a department and a termination date.

As part of processing, we take these 3 tables, and populate a forth, people table. Where possible we try to match up new people with existing entries – like where a staff member is also a student.

We finally have all of the data we need (more or less), and we have list of all people, current and past, but we don’t have a good handle on WHAT they are. We once had columns (student, employee, guest) in the people table, but it wasn’t scaling all that well.
Here is the big piece of technical magic….

The person status view, which is explained in detail in the paper, takes each of the individual tables, with their own fields, student, employee, ID guests (and a few others), matches up some keys in the people status types table, and produces a uniform person_status record based on what it finds.

This is where all the site specific stuff ends, and a common format goes forward.
Each “active” person has at least one status entry – it is possible for them to more than one. The main columns of interest are the Person_Id, which points back to the people table so we can get name, address, and all of that info. Next up is the Status_Id, which points into the People_Status_Types table – which defines each of the different types. One of the important things it gets from here, is a rank – this is used to order status entries when someone has more than one – for systems and things that can only handle one status.

Also in the view is an department (or organization) name and a department code. For students, we don’t get a department code – the name is good enough. In some cases, we also have an end date. I also pass through a flag indicating if that person should be in the online directory – a reasonable idea perhaps, but not appropriate here.

This was a good start at managing WHAT people were – but I wanted more.
So I wrote a process that compares the existing state of the Person_Status view with an almost identical table called People_Status. This gives me some performance gains, as I display status information a lot – it has proven to be very helpful in tools dialogs to select a person. But it gives two other things – the first is a Start_Date – when did we first see someone with a given status, and what is even more important, provides a place to hold history.

In the person select dialog, I list the person’s current status. If they don’t have any valid status, I list their last status and tag it as “Former” and the termination date. This has proven very helpful in identifying records for people who have been separated for some reason.
Here is a pull down selection box that is part of a person selection dialog from one of our web tools. We were looking for someone named Powell, and we have a bunch to choose from – including two named Alan. But along with their names and their Rensselaer ID Numbers, we also have their status and department value which can be very handy in getting the right entry. In this example, we also have a few former members of our community – their last status and their termination date are included.
Grouping Based on Status

- Library Circulation System
- Mailroom Distribution Lists
- Building Access Control
- Directory
- Status Based Services
  - PC Store
  - Trouble Ticket
  - Computer Accounts

Being able to identify all the members of a particular group, or set of groups such as all faculty, all students and employees, etc is a very powerful tool. From this we can feed other systems such as the library circulation system, whereas depending on what you are, you get more or less access. We also use this to maintain the campus mailing lists. We use it for access control to buildings and facilities, especially after hours. This controls who goes into our online and hardcopy telephone directories. We use it to validate who can access certain services such as a campus computer store and who gets loaded into our trouble ticketing systems. It even determines who gets email accounts – which is where this whole story started.

The more consumers we have of this data, the more important it becomes, and gives more power to the data sources. Given the higher importance of the source data, the cleaner and more timely it becomes – if a data source is slow or makes errors, they get all the pressure from the end users. But since the data is getting cleaner and more timely, it becomes more attractive to new data consumers – a nice feedback loop.
Guest Management

- Everyone is in a “Department”
  - Except dependents
- Initial entry via ID Card Desk
  - Ongoing Maintenance Delegated
  - Everyone Expires
- New types are easy to add

Basically, people who were not employees or students of some kind, were guests. And we had to provide some tools to manage these. To this end, I took my earlier work on managing the phone directory, and my first pass at the ID guess problem, and merged them together. This gave me a ready made set of tools and structure to build from.

I already had a organizational tree for all the departments on campus. I added some additional trees to manage new groupings, from incubator companies, to on campus vendors, off campus vendors, tech park companies, and so on. The existing directory tools were ready to maintain these as well.

All guests had to enter the system via the ID Card office. I have worked closely with them for many years and try to provide some reasonable level of support – in turn, they set up all the paper procedures with all the outlying offices – but they can offer the one stop shopping – work with them and everything else comes automatically. But once someone is in the system, the ongoing data maintenance goes back to the source (or department). We no longer allow open ended guests – guest HAVE to be renewed, or they expire automatically. And the person responsible to renew, is the one who will get the complaints when someone expires.
Delegate Data Maintenance

- Departmental Administrators
  - Renew and expire ID Guests
  - Maintain Directory Information
  - Control Computer Accounts
- Service Administrators
  - Set parameters based on status
- Subtype Administrators
  - Member Management

One of the problems we encountered with the phone directory, is that while HR might know the current employment status of an employee – they are not aware of their directory information, their email address, etc. What is more, they really were not interested in maintaining that information. So we identified one or more “directory department administrators” for each department, and gave them a tool so that they could update directory information about their staff. When we added ID Guest management, just about every guest was associated with an existing department, so the departmental administrators got an enhanced tool that lets them maintain information about their guests, controlling if they should go into the directory, having an email account and renewing or expiring the person as appropriate. Once a person is entered by the ID card office, the maintenance of their data goes back to the appropriate department who requested the person in the first place.

Another thing we delegated, was to allow each service administrators to set parameters based on status type, which we then feed back to those service administrators. In this way, the library staff could set the Patron type and limits for people based on their status. Likewise other data consumers could select who was included in their feeds.

We were also able to address a problem with our Emeriti faculty. These are faculty who are retired, but are entitled to all of the benefits of being a regular faculty member (but they don’t have to attend faculty senate meetings). We had previously set up a tool to enable people to manage mailing lists. A few minor modifications to the tool and a tweak to the person_status view, and now the Provost’s office was able to maintain lists of all current emeriti faculty. These lists fed status values, which in turn fed computer accounts, directory entries and the parking system! (You do NOT want to deal with an emeritus faculty member who was just denied access to the parking garage!). Our access control system terminates all access when you loose all of your status.
Futures

- Tools entirely data driven
- Service Flags for individuals controlled by Departmental Admins
- Service Flag defaults set on a per department basis.
- Service Based Ranking

This project has grown – many units of the university are dependent on this working, so I don’t see it going away any time soon. I do forsee some minor changes.

The first would be to make the guest process entirely data driven. At present, a small amount of coding.

The next would be to allow the easy definition of arbitrary flags on a per service basis, that are assigned to each individual. This would allow me to get the In_Dir flag out of the id_guests table. As part of this, we would also allow setting the default values for these flags on a per department basis rather than just on a status type basis.

We also rank order status values – there might be some applications that would want a different ranking – some academic programs might want to identify someone as a student first, and an employee second.
Chicken Versus Egg

• Computer Accounts
• Directory/White Pages
• Mailing Lists (Mailroom)
• Minor Systems (provide data)
• ID Card/Access Control

We got our original data feeds to provide computer accounts. When we later tried to handle the directory, we discovered that the feeds were not quite good enough, so we were able to upgrade them a bit. But as you feed more systems, the importance of your data feeds goes up in the eyes of your data sources.
Ok, what do I think are some of the important things for you to take away from this presentation?

Using the person_status view to standardize all different data sources and formats into a uniform status value is a very powerful concept – and everything after that is generic – not site specific.

Identify your data sources – and find ways to add value to their processes, while at the same time solving your problems.

Figure out all of your “none of the above” types, and identify who is really responsible for them.

Foisting off work on others is always good – but call it empowering – it sells better.

Identify existing tools that can be re-used to solve other problems – our mailing list manager tool was quickly adapted to handle the emeriti issues, so when the topic came up – I was able to quickly give the Provost’s office a tool to solve their problem, and also solve mine.
Thanks…

• Program Committee
• Tom Perrin
• Rob Kolstad
• Joyce
? Questions ?

Manage People, Not Userids

Jon Finke
Rensselaer Polytechnic Institute
finkej@rpi.edu
www.bedework.org

• **Bedework: an open source institutional calendar system**
  
  – Bedework v.3.0 is a Java-based, enterprise-wide calendar system (server and clients) for higher education that supports both public and personal calendars. Bedework is the standards-compliant fully rearchitected version of UW Calendar built on Hibernate.
Objectives

• Simplify Intake
  – Single Entry point per person
  – All services provided automatically
  – Simplify Separation Process
• Eliminate Duplicate Data Entry
  – Reduce administrative overhead

Before I dive into what we did and how we did it, (and how you might do something like it at your site), lets first touch on some of my objectives for this overall project.
I could solve this problem using a relational database, such as Oracle.

Ok – is anyone here surprised that I came to that conclusion?

How about this,