Building azdistancelearning.org

John Campbell
Northern Arizona University
Flagstaff, Arizona 86001

Abstract:
In June 2000 the azdistancelearning.org web site was launched to showcase courses taught at a distance by the three Arizona universities. The Arizona Board of Regents office coordinated the work while staff from Arizona State University (ASU), Northern Arizona University (NAU), and the University of Arizona (UofA) built the actual web site. The site's design is surprisingly interesting; the developers made extensive use of leading-edge technologies to solve non-technical problems related to data ownership, local control, and institutional status. Key technologies include Oracle replication, WebDB and round-robin DNS. Staff from ASU, NAU, UofA, the Arizona Board of Regents and Oracle created a "virtual university" schema to handle differences in data field values. Oracle donated a WebDB programmer to assist in implementing the graphical user interface. This paper discusses the key technologies and how well they worked to bring together three very autonomous institutions.
Building azdistancelearning.org

Arizona's three state universities, Arizona State University (ASU), Northern Arizona University (NAU) and the University of Arizona (UofA), have long been interested in distance education. Each of the three state universities offers courses taught at a distance using various modes of delivery. Relevant examples include correspondence courses, broadcast TV courses, interactive TV courses, and, more recently, web based and web streamed courses. The presidents of the three universities began to see the total collection of these combined distance education course offerings as a statewide resource. In 1998, they launched a modest project to build a single web site that would serve as a statewide course catalog and class schedule for current classes available at a distance through their respective Arizona universities.

In October 1999, an Arizona Board of Regents (ABOR) staff member brought representatives from each of the three state universities together to form a technical team charged with building this new web site. After a couple of initial meetings the team also brought in Oracle as a development partner. The result is the new azdistancelearning.org web site that allows users to search for “non-traditional” classes taught by any of the three Arizona universities.

This paper discusses how the web site was built--including a look at the site's unique architecture as well as the equally unique collaborative challenges presented by this sort of joint project.

Site Objectives

Before the technical team got started, a working group had already been discussing and negotiating details for a combined on-line schedule of courses on the web. A number of difficult goals and objectives were discussed and negotiated, leading to a very high-level design specification for a searchable web site. The working group had oversight responsibility and was available to resolve issues if necessary. After agreeing to the proposed site design, however, this group was not directly involved in the implementation.

The implementation team designed and built the web site subject to the general guidelines already established:

- The web site would list distance courses from each university based on an agreed upon list of modes of delivery.
- The web site would be neutral and not favor one university over another.
- The web site would be built and maintained using university and not Board of Regents resources.
- The web site would be searchable on delivery mode, course title, keywords, year and term.
- The web site would be limited; the university where the course is taught would handle detailed course and program information, admission information and enrollments.
- The total web site budget would not exceed $30,000.

Political Challenges

Arizona’s three state universities are quite separately administered with different student information systems, different host architectures and even, to some extent, different missions. Cooperation occurs between the three universities regularly, but there is also a lot of healthy competition among the three. In the first kick-off meeting, one of the implementation team members introduced us all to the concept of “coopetition”—a term indicating that the web site might be neutral but the goal was to give each university more distance learning students. Hence it was critical to consider how best to design a system that would send students back to the home campus and give everyone an equal chance to capture the student’s attention and attendance.

From the outset it was clear the web site needed to be data driven. Hence, a database of some sort was clearly required. And, a strong argument was made to keep data loading and data management under the control of each university. For readers not familiar with Arizona, it’s also important to know that Northern
Arizona University in Flagstaff is 150 miles from Arizona State University in Tempe and another 110 miles from the University of Arizona in Tucson. Hence, collaboration efforts required some travel and lots of phone and email conferences. The final system design also required technology robust enough to work over a wide area network.

The solution to data loading and ownership resulted in turning to Oracle (already a strong presence at NAU and UoA) and Oracle's data replication capability. By having three servers, each capable of hosting the data and the web site, and by using Oracle's replication services, we could neatly sidestep issues such as who would be responsible for each other's data and how to assure that data would remain timely and up to date. Also, as planning progressed, it seemed a good idea politically to bring on a corporate partner. Since the budget was tight and the project was very high profile, we hoped to partner with a prominent higher education provider to our mutual benefit. Oracle agreed to be this partner.

In the current production system, data are loaded locally at any time and, through database replication, show up soon at each of the other sites. One university loads data from the student information system through a nightly data feed; the others load data as they deem necessary. Figure 1 demonstrates how the servers interact with each other and the web.

Figure 1: Three Server Architecture

Note that if a single server hosted the web site it might send the message that one of the three universities was more capable or more favored as a "virtual host" than the other two. We resolved the "ownership" of the site by creating three functionally identical Apache web servers that respond via "round-robin DNS" to randomly give each client one of the participating servers. The result puts the arizonadistancelearning.org web site near the state of the art in terms of redundancy and reliability.

In fact, any two of the servers can go off line (either for expected maintenance or unexpectedly) without bringing down the service. Further, three servers can handle nearly three times the traffic compared to a single server solution. And, through the "magic" of replication, any one site can manage their data without needing permission from, or waiting on, the other two sites.

Collaborative projects are hard. Working with five agencies (NAU, UoA, ASU, Oracle, and the Arizona Board of Regents) required commitment, trust, and a tolerance (if not an appreciation) for group
differences. Technical folks can often take solace in technical problems and the approach we took certainly was technically challenging. One view is that this helped bring people together and keep them in good spirits. Another view is that the site is “too complicated” and a simpler approach would have served just as well.

Design Challenges

From the outset the $30,000 budget was clearly not going to pay for the staffing, software licenses, or machine capacity needed to build and maintain this web site. Thus it was understood and agreed that each university would donate their local staff and equipment to support the project. We also applied to Oracle for assistance and were generously given an “Oracle scholar” to help develop the graphical user interface. The implementation team recommended that the money budgeted be used for marketing.

Each university manages its own distance education course information and each is in a different format with different tables, fields, and codes. Part of the challenge was to design a common entity-relation diagram and then a schema to support a “virtual university” view of all this data. The model produced is extensible beyond the three current universities and consists of 22 replicated tables along with 6 support tables. Parent-child relationships where used where possible to do the actual “roll up” of each university’s data into a virtual university. Representatives from each university spent considerable time and effort reviewing the entity relations and schema to assure that their data would fit into the schema and that data differences could be handled in the final search engine. Figure 2, for example, illustrates the final parent-child design for the “mode of delivery” field:

Figure 2: Mode of Delivery Parent-Child Relationship

<table>
<thead>
<tr>
<th>VU Parent</th>
<th>ASU Children</th>
<th>NAU Children</th>
<th>UofA Children</th>
</tr>
</thead>
<tbody>
<tr>
<td>VU1 Web Delivered</td>
<td>ASU1 Web Delivered</td>
<td>NAU27 Web Streamed</td>
<td>UAWC Web-based classes</td>
</tr>
<tr>
<td>VU2 Cable TV</td>
<td>ASU2 Cable TV</td>
<td>NAU13 Cable-Individual</td>
<td>NAU14 Cable Overflow</td>
</tr>
<tr>
<td>VU23 Public TV</td>
<td>ASU23 Public TV</td>
<td>NAU16 Satellite TV</td>
<td></td>
</tr>
<tr>
<td>VU3 Satellite TV</td>
<td>NAU16 Satellite TV</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VU4 Video Tape</td>
<td>NAU11 ITV-Origination</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VU5 Correspondance Mail</td>
<td>NAU12 ITV Receiving</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Since each university maintains and manages their own data, it seemed only fair to expect each university to load their legacy data into the common schema. Each team eventually selected different tools for loading the data. What we probably didn’t appreciate was the difficulty loading the data would cause for one of the teams. We freely shared scripts between sites, but in hindsight a user’s guide on data loading might have helped in this area. Frankly, since the schema is normalized and contains many dependencies, data loading turned out to be one of the most difficult aspects of the project. Still, numerous phone calls eventually got everyone’s data loaded.

Division of Resources

The implementation team effort was initiated and subsequently led by a staff member from the Board of Regent’s office. Figure 3 shows the seven teams listed in the initial project plan with leadership roles shared between the partners. Data loading was assumed to be each member’s responsibility and not listed as a team effort. (In retrospect, this was probably a mistake.)
Some of the work proceeded in parallel, but the Data Discovery, Schema, and Data Replication efforts clearly seemed to be on the critical path for this project. A weekly phone conference led by the Arizona Board of Regents’ project lead was critical in order to coordinate the different efforts.

Each institution agreed to donate a Sun server running similar versions of Solaris, Apache web server, and Oracle. Initially, all database instances were built at NAU and replication was first tested in a fairly controlled LAN environment. Moving to our final WAN environment (spanning the whole state of Arizona) definitely exposed the design to more frequent and longer network outages. To date, however, this has not created a large problem; the replication queues have not grown overly large while waiting for network connections to be restored.

Working with a corporate partner was a new experience for most of us. Oracle has a “visiting scholar” program whereby an Oracle sales engineer visits a campus and gives talks on new technologies. At times, these Oracle scholars also help build a small pilot or demonstration project to showcase some hot new Oracle product. At the time we were building azdistancelearning.org, the Oracle WebDB product was new and hot. Oracle generously agreed to provide the project with an Oracle scholar. Before development began, however, the first Oracle scholar assigned to the project left Oracle. A substitute was forthcoming, but it required not only getting this person up to speed but also convincing his boss that the sales force had not given away the store and that the WebDB developed code would not end up being supported by Oracle.

**Technical Spotlights**

According to at least one student reporter, the azdistancelearning.org web site is easy to use. It was not, however, easy to build. From properly configuring the round-robin DNS server, to data loading, to building a migration path in order to put the data and WebDB scripts into production, each step had its own unique challenges and required its own set of experts to resolve the problems. Perhaps the most critical and most successful piece of the puzzle was getting data replication working in a stable and production environment across a wide area network.

**Bootstrapping the servers**

As mentioned earlier, the initial databases were housed at NAU and were built as the schema began to be finalized. As data was loaded and used in early search engines, schema changes were proposed and voted on by the implementation team members. Over time, fewer and fewer changes were required.

The final project uses a total of nine separate databases. Each university has a development, test, and production database to work with. The NAU database administrators (DBAs) approved and managed the table layout, migrations, and replication efforts necessary for the project to work. Without strong DBAs this project would not have succeeded.

Before the site went live and into production, each of the databases was exported and reloaded at their home sites. At the same time, the Apache servers were configured so they were functionally equivalent as “virtual hosts” and the WebDB scripts were loaded separately into each system. Replication, once it’s set up properly, makes it easy to administer the data. However, the server configurations, Oracle database
management, Apache web server, WebDB scripts and various static web pages all need to be duplicated precisely on each separate server. While this sounds very bad, in practice it has not proven to be much of maintenance burden.

**The Virtual University Abstraction**

Each of the three Oracle servers hosts a replicated copy of the schema used to implement the shared database. Each server, however, is unique by virtue of a prefix (ASU, NAU, UA) designed to separate and identify data belonging to the host university. To implement the unique prefix, a *virtual university* table exists at each site and contains the prefix for the given university. This table is not replicated and contains just one record. Any time values are added to one of a handful of tables, a trigger fires and adds the university prefix to the primary key before writing the record. This assures that data from different universities will peacefully coexist in the same tables. Examples of tables that have this trigger include instructor, section, section location, credit type, mode of delivery, and term information tables.

One of the greatest challenges was to combine data from disparate systems and allow these data to exist in the same tables and have some common meaning across the different universities. The virtual university parent-child relationships allowed us to roll up different university values into common sets. Figure 1, for example, shows that VU1 (Web Delivered) mode of delivery is parent to ASU1, NAU28, and UAWC modes of deliveries. To support application programmers, database “views” were created that contain both parent and child columns.

For example, to find all the web course class sequence numbers, the application programmer would create a SQL statement searching for a parent type of VU1. Figure 4 shows an actual query that would select all web delivered sequence numbers from all universities along with the actual mode of delivery description used by that university for that particular course. (Note that a real query would also want to include the year and term.)

**Figure 4**

```
SELECT sequence_number, g_child_desc FROM gv_mode_deliv_type_r_section WHERE g_parent_type_id = 'VU1'
```

The parent-child relationship solves the problem of searching across different, and sometimes quite complex, types of data. The “term information” roll up, for instance, allows us to gracefully handle different length summer, fall, mini-course, and other terms by grouping those that are related into a smaller set of recognized virtual university terms. Thus a student can search for a “VU Summer” parent and find all the individual university’s summer terms.

**Protecting Shared Data**

A total of 26 tables are needed to implement this schema. All but two are replicated: the virtual university table and the instructor “external id” table. In the first case, we simply need the university code to be unique to each local site. In the later case, many of the instructor id’s are actually social security numbers. As such, these cannot be shared between institutions. Hence, to list instructors in courses requires loading the external id’s into a private table and then, through a trigger that combines a sequence with the university prefix, a unique identifier is generated as the primary key used by the rest of the tables. Thus, no university has any other university’s social security numbers on their server. Also, note that an architecture relying on just a single server would have to solve this problem differently.

**Round-robin DNS**

Actually, round-robin DNS is now called “random DNS” and is supported on most modern Domain Name System (DNS) servers. DNS is the Internet’s way of resolving a host name to a physical IP address. The “random DNS” trick is to list multiple hosts for a given host name in order to do some crude load balancing and provide redundancy for important web services.
Technically it’s relatively easy to set up: you simply add each of the servers’ IP’s into the “A” record list on a DNS server that supports random DNS. It’s usually also a good idea to reduce the “Time To Live” field to a small value in order to stop other DNS servers from caching the data if you really want to do load balancing. It’s also necessary to set up reverse pointer records on whatever local DNS servers need to do reverse name translations. While these steps are easy in theory, in practice most automatic DNS tools do not give this level of access to the DNS tables. So, as we learned, the tables may have to be edited by hand.

In our case, the DNS server is at ASU and we had to assure the system administrators that each of the other institutions did not mind having their IP addresses in the ASU server. Good system administrators are very reluctant to administer someone else’s IPs.

Once the DNS server was set up to use random DNS, we found it very difficult, as developers, to tell which web server had actually responded to our web client. For that reason, during testing, we added an HTML comment at the bottom of the page to tell us which server responded. The reader can see this by connecting to www.azdistancelearning.org and then viewing the html source for the page.

Finally, when the client requests a `gethostbyname`, the server returns the whole list. Most web clients will try the next server in the list if the first server times out. This increases the reliability and robustness of the overall azdistancelearning.org web site. If one server is down the others will still work, subject to a delay, even if the DNS table is left unchanged. For prolonged outages, of course, the DNS table should have the errant server temporarily removed.

**Replication**

The unsung database administrators were vital to this project. Without replication the whole design would fail and the only solution would be to bring up one server at one site and face the practical and political consequences. In such a scenario, there would be no redundancy and the whole service would be offline during server maintenance. Without replication the overall web site would handle fewer connections and it would be much more difficult to load and manage each university’s data.

Oracle replication works by generating internal system triggers to run stored procedures whenever an insert, update, or delete would modify a record in a replicated table. These transactions are pushed onto a queue and processed in the order they arrive. Ours is a “multi-master replication” design, which allows each of the three hosts to have a queue for the other two hosts so that each host is master for some part of the data. For this to work, we rely upon the unique keys generated through the university prefix triggers mentioned previously. These keys segregate the data into disjoint sets so that each university’s host data are clearly unique and separate from the other universities.

Our experience has been that when things work, they work very well. Even though we have regularly had short network outages between the universities, the transactions are queued and simply wait to be processed. Replication thus gives the data loading teams freedom from worrying about transferring data over a sometimes flaky wide-area network.

When things don’t work, the story is very different. Fixing problems is a challenge and requires expert database administration skills and a good understanding of replication. At least once we’ve had to manually turn off replication, disable triggers, remove parent-child constraints and then fix a data synchronization problem to allow the transactions stored in the queues to complete. Once a problem is fixed, all the triggers and constraints need to be put back again in order for things to work properly.

Having the development and test databases has been a necessity and not a luxury. At various times the database administrators have had to develop administration procedures in the development database and test them again in the test database before finally applying the procedure to the production database. While the goal is to hold each site responsible for administering their Oracle databases, the reality has been that only one team really has the expertise to solve significant problems when they arise.
Still, there are local tools each site uses to insure that tables are not exceeding available disk space or that
the queues are not growing unreasonably. And, performance has not been a problem, despite the fact that
the three servers are all very different in capacity (ranging from the smallest to the largest of Sun’s
Enterprise series).

Replication remains the core of our design. There have been very few problems and, despite the need for
highly skilled database administrators, Oracle replication has worked very well since the site went live in
June 2000.

**WebDB**

WebDB, which is now called “Oracle Portal”, is a graphical development environment designed to make
developing data driven web sites easy. Given that this site had a predetermined graphical user interface and
some very complex data access and searching rules, using this tool did require developing some custom
Oracle scripts. The product has evolved since we used it in June 2000. Today it has new features and has
taken on a new direction.4

Both the WebDB toolset and the WebDB application had to be loaded independently on each server. At
one time we explored replicating the WebDB application so that it too could benefit from data replication.
However, most of the application is in stored procedures, which do not seem to be candidates for
replication.

The Oracle developer working on this project did an amazing job of developing while on the run. Much of
the product was built at 30,000 feet while traveling from one customer’s site to another. The final version
was knocked out in a three-day meeting at ASU involving a core of the implementation team. While no
compromises were made in the design requirements or expected functionality of the graphical user
interface, much of the delivered functionality had to be implemented as Oracle stored procedures. This is
not an indictment of the product but it does point out that tools designed to make jobs easy should have
“back doors” to allow professional developers access to underlying features when they are needed.
Because WebDB is implemented as stored Oracle procedures, not only was the Oracle developer able to
build what we wanted, but our maintenance team, familiar with Oracle, has been able to go in and make
changes as needed.

**The Future**

Success is in the eye of the beholder. On the one hand, the implementation team met many functional,
technical and political challenges and built a robust and successful web search engine. On the other hand,
this paper has ignored the larger question of an Arizona Virtual University, which has been discussed in
statewide planning sessions and even written about in some press releases. While the current
azdistancelearning.org architecture is certainly robust and scalable to the extent that it could handle many
more users or even support other universities through its “university code” design, the reality is that a real
virtual university would need to solve tremendous policy, admission, and registration issues. Students
enrolled in a future Arizona Virtual University would expect to find degree programs built from the
complete catalog of courses offered by the state. An early article in the University of Arizona student
newspaper suggested this was “just around the corner” and that students would soon be able to take courses
at any of the universities and apply it to their degree.

The reality is that the current system cannot easily scale to include student transactions. In other words,
student admissions, registrations, financial aid, and transcript management are well beyond the scope
considered by the implementation team and are probably beyond the capabilities of a multi-master
replication architecture. While seeing all these courses listed side by side easily raises the expectation that
the next phase will be to allow co-enrollments between the three universities, the reality is that there are
significant policy, procedure and technical challenges to solve well before any new system requirements
can be developed.
One of the biggest, and most obvious, lessons learned by the implementation team is that even an unfunded project takes resources. This project, with its seemingly limited scope, took significant time and effort to develop and continues to take some time and effort (not to mention machine resources) to maintain. Success relied upon the willingness of each campus to donate resources from many different areas. The team has been unanimous in recommending that any future phases be fully funded and staff permanently allocated to the project.

End Notes

1 The Oracle WebDB product is now called “Oracle Portal.”
2 See Ryan Gabrielson’s Arizona Daily Wildcat article (http://wildcat.arizona.edu/papers/93/154/01_1_m.html).
3 Many people use different acronyms for DNS. See http://www.faqs.org/rfcs/rfc1035.html for more information.
4 For information on Oracle Portal, see http://technet.oracle.com/products/iportal/listing.htm.