Establishing an Information Architecture
Integration with an Open Systems Environment

Gerry Bernbom
Assistant Director, Data Administration and Access

Dennis Cromwell
Manager, Information Technology and Standards

University Computing Services
Indiana University
Bloomington, Indiana

Abstract

This paper describes the conceptual basis for a structured data architecture, and its integration with the deployment of open systems technology. The key strategic initiatives which brought these efforts together are discussed: commitment to improved data access, migration to relational database technology, deployment of a high-speed, multi-protocol network, and orientation to workstation-centered computing.

Details of Indiana University's data architecture are presented, focusing on the relationship of operational to information-detail databases, the creation of synthetic data structures and higher order "data collections," and the integration of the university-wide architecture with local and departmental data sources. Discussion of the underlying technology paradigm addresses the integration of proprietary components with "open" systems solutions in a strategy of technology coexistence that addresses: mainframe/server, workstation/client, networks, gateways, and database management systems. Finally, some next critical events are outlined for both the architected data environment and the open systems technology environment.
Establishing an Information Architecture
Integration with an Open Systems Environment

Background and Context

The ideas presented in this paper about data architecture and open systems derive in large measure from trends and ideas that are moving through the entire information technology community. Beyond that, the paper discusses how these ideas have been shaped and implemented in the specific computing environment of Indiana University. Two aspects worth noting about this specific environment are the overall university organization, and the organization of university computing services.

Indiana University is an multi-campus university system. It enrolls more than 90,000 students, and maintains a very high degree of integration and interdependence among its eight geographically dispersed campuses: a single academic school (division) may span multiple campuses, offering the same degree program in multiple locations across the state; faculty regularly teach in more than one location, often during the same academic term; similarly, students enroll concurrently on more than one campus; the university experiences a high degree of inter-campus transfer on the part of students between semesters; and so on. To support this degree of multi-campus interdependence, in most areas of administrative data processing there is a single set of information systems, centrally managed by one computing and information services group, that supports the entire university in all its geographic locations. Thus, size and geographic diversity, balanced against the expectation of close integration, are among the constraining factors that must be accounted for in any technology planning and decision-making.

University Computing Services is a merged academic/administrative computing organization whose mission includes: providing administrative computing services to all eight campuses of the university; providing inter-campus network services among all eight campuses; and delivering or enabling the necessary support structure for these state-wide services. In addition, University Computing provides more intensive computing service support to the Bloomington campus in the areas of instructional and research computing, student computing, and intra-campus network management. A very small set of key strategic initiatives has provided much of the cohesive and motivating power to the computing organization. These initiatives, initially developed in 1989, during the structuring of the current organization, and refined in the time since then are: to develop a high-speed data network; to promote workstation-centered computing; to deploy relational database management systems; and to promote widespread access to data. As with the facts of life about the university, the mission and strategic initiatives of University Computing provide some of the constraints and boundaries that inform the technology plans and decisions of the organization.

Data Architecture

As used in this paper, data architecture is a design and analysis technique based on the macro-level analysis of data within an enterprise. The result of this analysis is a general purpose model for the structure and deployment of all of an enterprise's data, regardless of data subject area.

One part a data architecture model is primarily descriptive in nature, beginning with recognizing variations in the general nature of data within the enterprise: detail data vs. summarized data, real-time data vs. point-in-time ("snapshot") data, etc. The data architecture model also takes note of the variations in intended uses of data: transaction processing, front-line customer service, management reporting, decision support, institutional research, etc.

A second component of the data architecture is more prescriptive in nature. It specifies, for the entire enterprise, standards for information synthesis: how summarized data are derived from detailed data, how standard points-in-time are determined for creating "snapshot" databases, etc. The data architecture model also assists in technology selection and evaluation. By knowing the general form of data, its intended use, and some of the associated performance requirements, the data architecture can help specify the standard or recommended technologies for deployment of different categories of data.
Put otherwise, data architecture is modeling the process of information synthesis, the creation of useful information from lower-order forms of data. The general model produced by data architecture analysis helps to identify or establish authoritative data sources, and to define stable relationships among multiple instances of the same data (frequently in different forms or stages of synthesis) that are used for different purposes in the enterprise. Data architecture also helps assure predictable transformations of data throughout the process of information synthesis. Finally, data architecture helps match the technologies used for data deployment to the intended use of the data.

There are in the information technology industry three or four threads of discussion that are directed toward this same set of data architecture issues. The first to come to our attention was the work of Bill Inmon on "data architecture" and, more recently, the "data warehouse" concept. (1,2) Subsequent to much of Bill Inmon's pioneering work, IBM unveiled its "Information Warehouse" framework. (3) And, in partnership with IBM, Information Builders, Inc. developed their product offering of Enterprise Data Access/SQL (EDA/SQL). The final industry thread that we have followed, and one that looks especially promising in terms of where we want to take the information architecture of Indiana University, is Apple Computer's VITAL (Virtually Integrated Technical Architecture Lifecycle), a conceptual model of information systems design and information access. (4)

An Approach to Design of a Data Architecture

Having put in place a few basic concepts and principles, the clearest introduction to the concept of a data architecture is to look at one. Figure 1 is the basic data architecture model developed for institutional data (also called administrative data or management information) at Indiana University.

The first key point to note is that the model divides data into three strata: Operational, Information Detail, and what are called "Collections." At the lowest level (Operational) are those data used in the day-to-day operations of the enterprise: business transaction processing, data capture/data entry, real-time customer service, etc. At the second level (Information Detail) are the idealized representations of an enterprise's data, minus the redundancies and anomalies that are a natural part of any set of operational information systems, but still having a nearly one-to-one correspondence with the business objects represented by operational data and still maintained at the finest level of granularity. At the highest level (Collections) are found synthesized information products, derived from lower order and more detailed data sources. These collections may be summarizations of detail data, selected extracts, aggregations from multiple detail sources, etc. The metaphor applied to this three-level model is:

- Operational $\Rightarrow$ Manufacturing
- Information detail $\Rightarrow$ Wholesale
- Collections $\Rightarrow$ Retail

Three other points to note about the model in general. First, the movement of data within the model is one-directional; the process is one of refinement and synthesis, and there is no downward flow of data in this model and no back-loading of data from higher-order forms to lower. Second, all business transaction (update) processing occurs at the lowest level in the model, the operational; information detail and standard collections are read-only data and are never updated directly. Finally, the model recognizes the need to integrate enterprise data with local data, both as a point-of-origin (in local operational data) and as a final destination (in collections of information for local analysis and decision support).

Data Architecture and Indiana University

The development of the data architecture concepts and their implementation has been, and will continue to be, a long term process for Indiana University. From our current vantage point we see the process as having four steps: 1) separating operational data from information detail; 2) refining the information detail; 3) developing standard collections; and 4) integrating enterprise data with local data.
Step 1: Separation of operational data from information detail.

In this step separate physical storage locations and formats are created for operational data and information detail data. Separate storage is justified, even required, because of the way these two classes of data differ from each other in terms of typical use and performance requirements.

Operational data are typically retrieved one record at a time; retrieval is of a specific record or set of records, identifiable by key; and records are frequently inserted or retrieved for update processing. Additionally, operational data are typically being retrieved interactively -- for on-line transaction processing or real-time information display -- and so require response time in seconds (if not sub-second response).

Information detail data are typically retrieved in sets of many records; defined sets of records are typically retrieved based on secondary (non-key) characteristics; and records are virtually never retrieved for update processing, all access is read-only. For most information detail purposes -- analysis, reporting, and decision support -- response time in minutes, if not hours, is sufficient.

Given the different uses and performance requirements, separate physical implementations can make use of data design techniques appropriate to the needs of each class of data: for example, operational data stored in highly navigational databases tuned for on-line performance and single record retrieval, and information detail stored in databases having multiple secondary indexes for searching and set-selection.

Aside from structure and performance, one more reason for separate physical implementations is the need for information detail to have a stable set of values over some period of time. Operational databases are in a constant state of flux; they are the place where the business transactions of the enterprise are recorded in real-time. Reporting, analysis, and decision-support require some stability of data so that comparative analyses can be performed, iterative analysis can pursue a line of inquiry on a stable set of data values, and the same results can be replicated by multiple users of the same data. Accordingly, one requirement of the information detail is that it be a time-variant replication of its operational data counterpart, having stable data values from a defined point (or points) in time.

At Indiana University, a two-part environment of transaction-oriented, operational data and fixed-value, replicated information detail data was first implemented in the early 1980s. This initial investment in data design has had great value over the past ten years providing a widely-used end user reporting environment. It also forms the foundation on which further stages of the data architecture model can be built.

Step 2: Refining the information detail.

The second step, at least for Indiana University, is a refinement of the information detail data environment that has been in place for the past ten years. In an organization just beginning to implement a data architecture model, most of these refining tasks would be accomplished in the initial design of information detail data.

Our chief tasks of data refinement are: development of a university-wide, conceptual data model; migration of the information detail environment from indexed and sequential storage formats to relational database formats; and resolution of anomalies and redundancies in the data structures by determining authoritative sources for each atomic data item.

A project to develop a university-wide conceptual data model, in four component pieces, was begun in 1991. The first two components, a conceptual model for student data and one for university financial data, are complete. Remaining to be done are models for employee data and physical facilities data.

Also begun in 1991 was the migration of existing information detail data from older storage formats to relational database format, using DB2. Thus far only 5% of the information detail environment is converted. (Though we know, too, that the critical mass of data that we absolutely must migrate to relational format -- to meet the majority of the university's access and reporting needs -- is far smaller than 100%.) The strategy now in place for further data migration is to follow the progress of the university-wide data model projects,
focusing initially on the student data model, and to address those data subject areas identified in a user survey as the greatest need for access: class schedule data, student enrollment and advising data, financial account status and budget data, and others.

Step 3: Developing standard collections.

Using the three-stage model described earlier, this step in a data architecture is intended to allow information delivery to move from wholesale to retail, to organize and package data in ways that are more ready to use. The process is one of synthesizing information from the information detail, and the chief tools of synthesis are summarization, aggregation, selection, and point-in-time representation. All these are transformations of data that one typically finds in user-designed reports, queries, or data selection/extract programs.

The term "collection" refers to any selected or synthesized set of data, derived from the underlying information detail. Collections may be implemented as physical entities, where the results of selection, summarization, and so on are saved and stored as a physical database. Or collections may be virtual entities -- data views -- where the same rules of selection, summarization, etc. are applied to a relational database at query-time using stored SQL.

The chief intent behind "standard" collections is to establish shared definitions for certain transformations of data within the enterprise: the calculation of grade-point-average based on hours and honor points, the set-definition of a full-time student or part-time employee, the accepted intersection of faculty data and course data for use in measurement of teaching load, and so on. Shared definitions, and their implementation in a data architecture, benefit the data user by providing commonly needed information, already synthesized and ready-to-use; they benefit the institution by helping assure that all reports and analyses have access to the same interpretation of the underlying detail data.

Moreover, a standard collection, implemented as a physical database, is desirable when some set of summarizations, selections, or aggregations is repeated so often that significant computing resources are being spent, many times over by many different users, executing the same program code. This is especially true of complex and often-used set-selections like: find all students enrolled in the current semester, find all accounts with outstanding balances greater than thirty days, etc. Finally, implementation of a collection as a physical database in the data architecture is simply a practical requirement when a collection is meant to reflect a "snapshot" point-in-time.

Our progress toward formalizing this step has come slowly. Within selected data subject areas Indiana University has had a long history of using standard collections, in particular for point-in-time snapshots (e.g., student census reporting files). Since 1990, with the implementation of relational database technology, we have also started using logical data views as a means of implementing standard collections.

Step 4: Integrating with local data.

The data architecture as representation for enterprise-wide data needs to be seen as an open architecture with respect to local stores of data that exist throughout the organization. These local data can be a primary source for enterprise-wide operational data; they feed the operational information systems of the enterprise. They can also be the final destination for synthesized information as used in decision-support and analysis at the local level. The enterprise cannot pretend to manage data centrally that is really managed locally. The purpose for placing local data within the architecture is to show the points of interface and to identify the need for these to be managed.

As source for enterprise-wide operational data, the key requirement for interface to local operational data is that these data be passed through steps of validation and authentication as prerequisite for their acceptance as enterprise-wide data. As final destination for data, the key requirements are to encourage use of standard collections as the basis for building local collections (i.e., encourage use of shared definitions, where such exist) and, if this cannot meet the local need, to assure that information detail (and not operational data) is used to populate local collections.
Open Systems

In light of its mission and strategic initiatives, University Computing at Indiana University has chosen an "open systems" approach as the best possible environment for deploying information technology. In the environment we are developing, five key points are used to define "open systems." First, the technology must be standards-based, whether these are national or international standards, such as are sanctioned by ANSI and ISO, or whether they are de facto standards based on industry acceptance and widespread use. Second, the technology must support a multi-vendor environment, providing interoperability between platforms and, in some cases, portability of applications across platforms. Third, the open systems environment must make use of available technologies, those that are on the market and generally available. Fourth, the technologies chosen should show signs of being sustainable over time within the industry as evidenced by such factors as market share, degree of penetration, vendor viability, and vendor commitment. Finally, the technologies should be oriented to the workstation, enabling the integration of information at the desktop. Indiana has committed to support a heterogenous (multi-vendor) desktop environment and selections of open systems technologies must support this with a user interface that is native to each of the supported desktop devices.

By way of quick comparison, two published definitions of "open systems," both of which provided some basis for our own:

From Hewlett Packard: "Software environments consisting of products and technologies which are designed and implemented in accordance with standards - established and de facto - that are vendor independent and commonly available." (5)

And from Gartner Group: "An information processing environment based upon application programming interfaces and communication protocol standards which transcend any one vendor and have significant enough vendor product investment to assure both multiple platform availability and industry sustainability." (6)

Examples of Open Systems Technologies

Three technologies, from three different technology families, can illustrate how the definition of open systems can be applied. Not every technology can score highly on every factor, but each of these three technologies is among the best in its class for meeting the criteria of an open systems environment.

In the area of networking, TCP/IP is a good example of an open systems technology. It is a de facto industry standard, is based on available technology, and supports the implementation of interoperability among multiple technology platforms.

In the area of relational databases, DB2 is a key player in an open systems solution. The product is based on the Structured Query Language (SQL) standard; it is an available and robust technology; and it is a sustainable technology based on IBM's investment and direction. In addition, DB2 is a platform to support interoperability (often based on third-party vendor products) because of its large market share.

In the area of local area networks, Novell is a piece of the open systems picture. It is a de facto standard for network operating systems; it shows very good signs of being a sustainable technology in the market place; and it offers interoperability among technology platforms and some degree of portability.

Because of their intrinsic merits, their fit with an open systems architecture, and based on other decision criteria, each of these three products was chosen by Indiana University as part of its overall open systems environment.
Strategy for Open Systems: Technology Coexistence

The strategy that Indiana University is pursuing for implementing its open systems environment is one best described as "technology coexistence," and is based on just a few guiding principles. Chief among these is that the network is the key to deploying open systems. It is the integrating and communicating force in the environment, providing access to multiple services and routing multiple protocols in support of these services.

A second principle is concentration on the user view of the technology: each workstation performs multiple functions in our view of an open systems environment; the functions are performed with an interface that is native to the users desktop device; and the desktop represents the integration point for services to the user.

Migration of data to relational database format is a third principle. In order for a coexistence strategy to work -- to enable gateway and network access to data across multiple technology platforms -- a common relational format for the university's data is a requirement. And a related principle is the deployment of gateway technology that links proprietary systems with the open systems environment. Especially as pertains to IBM's role in an open systems picture, gateways are the path to more open communication with MVS-ESA (the operating system of our largest central server mainframe); they also permit us to take advantage of DB2 as a database engine and its outstanding ability to manage and serve large volumes of data.

A final principle, which encapsulates much of the rest, is that a mix of proprietary and non-proprietary technologies is required for a successful open systems environment. This is the heart of the coexistence strategy. It builds on current strengths, it leverages past investments without being bound by them, it makes use of available or obtainable skills, and it tries not to induce trauma while introducing change to the organization. In particular, it tries to protect user investment in technology (the purchase of desktop devices, design and implementation of traditional information systems, training in tools of information access and analysis, etc.), again while trying not to shackle the computing organization to unlimited and open-ended support of all conceivable technologies.

Three Other Open Systems Strategies

Technology coexistence is not the only path to an open systems solution. Two other strategies outlined in work by the Gartner Group (though carrying slightly different names) are: "Freeze and Rebuild" and "Clearcut and Reseed."

Freeze and Rebuild (also called Fade In, Fade Out) is a fairly flexible transition strategy, at least in terms of the pace at which it is implemented. All information systems and applications based on "old" technologies are frozen; no new investment is made in their maintenance or enhancement. All new investment is made in applications based entirely on open systems technologies. This strategy works best in situations where there are a full suite of well-defined open standards and a clear direction toward their implementation. The strategy does carry with it a relatively high cost, both in terms of lost opportunities, as legacy systems fail to change with the changing needs of their users, and in terms of resource outlay for redevelopment of a completely new set of enterprise applications.

The second transition strategy, Clearcut and Reseed, is the most extreme. The plug is pulled on the old system and the new technology is rolled in. One day there are legacy systems and the next day, nothing but open systems. This strategy can work at the level of an individual application or closely related set of applications, or in a small and relatively simple organization. But it is very expensive, and probably impractical, to think about as a viable alternative for the revolution of technology on an enterprise-wide basis.

A third strategy is to simply Wait and See. If open systems are not well enough defined (if the problem with standards is that you have too many to pick from), just wait until the picture clarifies itself. In truth, this strategy may not be as safe as it sounds. One of the benefits in an open systems environment is the leverage it provides in negotiation with hardware and software vendors; even with some proprietary components, there are choices and alternate paths to almost any implementation. This leverage is lost to an enterprise that is not pursuing some type of multi-vendor approach to open systems. Moreover, the organization that waits too
long may end up being forced into open systems anyway, at a time and pace not of its own choosing, and will ultimately be forced to play catch-up.

Technology Suite at Indiana University

In establishing its open systems strategy of coexistence, Indiana University has selected and deployed products and technologies at four levels: the workstation, the network, the central server, and the database management system. Figure 2 shows the mix of these technologies and their interrelationships.

At the workstation level, Indiana is committed to supporting a heterogeneous mix of desktop computers. Machines running DOS or Windows 3.x are the most prevalent systems on our campuses. There is also a strong Macintosh presence on more than one of our campuses, and the last two years have seen a growing base of Unix workstations deployed as individual desktop devices.

Indiana University operates a high-speed data network among its eight campuses and has promoted the deployment on each campus of a high-speed intra-campus network. The network carries multiple protocols, most significant among them: TCP/IP, IPX, and AppleTalk.

At the level of central servers, Indiana operates a large scale MVS/ESA processor which, although proprietary in nature, does support TCP/IP access using Interlink Access MVS. The university maintains and is deploying more VMS and Unix platforms as application and data servers. VMS offers a slightly more "open" architecture than does MVS in terms of interoperability and portability, while retaining several of its key advantages such as structured operations and security management. Unix servers are making great progress in security, operations and performance and will play an increasing role, though they are not seen as an absolute prerequisite or requirement for an open systems implementation. Finally, Novell is a critical component today as a network file server, and as we watch developments in the arena of Novell/Oracle database technology we remain aware of its possibilities as a database server.

Regarding central server database technology, our basic investments today are in Ingres as the relational database on the VMS and Unix platforms, and in DB2 as the relational database on the MVS platform, with continued support for VSAM as a data storage and access technology where needed.

Mapping Technologies to the Data Architecture

In its evaluation and selection of technologies for its open systems future, and especially as regards the use of technology to support a data architecture, Indiana University faces questions at three levels of specificity:

1. What will be the suite of technologies we support, and how will we choose them?
2. How will we select one (or more) of these technologies to deploy for one of the functions within the data architecture? (Example: What database(s) will be supported for enterprise-wide operational information systems? Or, what technology might be recommended for local data collections?)
3. And finally, how will select from among the chosen and supported technologies (assuming more than one is supported for any general function) the correct technology for a specific application?

We have made good progress on questions at the first two levels of specificity. A suite of technologies has been chosen for most key technology families: networks, central server databases, etc. As each technology is chosen (Question 1) it is also evaluated and slotted for deployment in one or more of the general data architecture functions (Question 2). The criteria we have used to select and assign a technology include the basic criteria for any technology in an open systems environment: 1) standards based; 2) workstation oriented; 3) supports a multi-vendor environment; 4) based on available technologies; and 5) based on sustainable technologies. In addition, we have applied criteria related to resources and constraints: 1) user needs and functionality; 2) performance and cost; 3) established expertise within the organization, or its availability; 4) degree of integration with existing technologies required, and the degree actually possible; and 5) user preference.
One of the next critical steps, as mentioned below, is our need to develop better decision rules for the selection from among available technologies, where more than one technology is deployed in support of a single data architecture function.

In the area of database technology, Figure 3 shows how DBMS products have been mapped to each component of the data architecture:

<table>
<thead>
<tr>
<th>Component</th>
<th>DB2, Ingres, and VSAM</th>
<th>DB2 and Ingres</th>
<th>Ingres, DB2 (and data views)</th>
<th>Ingres, Paradox, FoxPro, etc.</th>
<th>Lotus, Excel, etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operational Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Information Detail</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard Collections</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Local/Operational</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Local Collections</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Next Critical Events

The data architecture and the open systems environment continue to evolve in their implementations at Indiana University. Some next critical events in the evolution of data architecture are summarized as follows. Information Detail data needs to continue its migration from sequential and hierarchical file formats to relational format. A methodology is needed for specification and creation of Standard Collections -- how they are defined, by whom, and at what stage of an information systems lifecycle. Decision-rules are needed to select from among available technologies, when more than one exists within any component of the data architecture. User access to data needs to move from a mode of terminal login to central servers to a mode of desktop delivery and integration. And, finally, the preferred source of data needs to move from the detail level to the collections, where significant synthesis will have already occurred and where shared definitions are accessible.

There are similar critical events in the evolution of an open systems environment. Developments in Novell/SQL database technology need to be tracked and evaluated; even if no central support for a single technology is envisioned, integration with user-selected technologies in this field will almost certainly be a requirement of our environment. Technologies for gateway access to and from the MVS world will also continue to be evaluated, both Remote Procedure Call (RPC) gateways for predictable and static interactions between platforms and dynamic SQL gateways for data interaction and interchange. A key goal for the next year or so will be analysis and resolution of issues related to security in a client/server implementation. And, finally, we will need to monitor and seek opportunities for the integration of "standard" Distributed Computing Environment (DCE) functions in our open systems environment: time servers, name servers, authentication servers, and so on.

Notes

(3) IBM Corporation, An Introduction to IBM's Information Warehouse Framework (Corporate Presentation Material, 1991)
(4) Apple Computer, Inc., Introduction to VITAL: Designing Information Systems for the 1990s (Corporate Publication, 1992)
(6) Gartner Group, Open Systems: Cutting Through the Confusion (Conference Material, 1991)
Figure 1: Data Architecture Model

Data architecture

- Standard Collections
- Local Collections

Information Detail

- Operational (Enterprise)
- Operational (Local)

Figure 2: Technology Suite at Indiana University

Technology Map of Indiana University

- DB2
- VSAM
- MVS
- APPC
- TCP/IP
- IPX
- APPLETALK
- NOVELL
- VMS & UNIX
- DOS
- Mac
- UNIX

Ingres
Figure 3: Mapping Data Technologies to the Architecture

Data Architecture/Technology Map

- Standard Collections: Ingres, DB2-views
- Local Collections: Lotus, Excel, ...

Information Detail: DB2, Ingres, ...

- Operational (Enterprise): DB2, Ingres, VSAM, ...
- Operational (Local): Ingres, Paradox, FoxPro, ...

CAUSE92: Information Architecture and Open Systems - Indiana University
Figure 1: Data Architecture Model

Data architecture

Standard Collections → Local Collections

Information Detail

Operational (Enterprise) ← Operational (Local)

Figure 2: Technology Suite at Indiana University

Technology Map of Indiana University

[Diagram showing various technologies and systems, including DB2, VSAM, MVS, VMS & UNIX, NOVELL, TCP/IP, IPX, APPLETALK, DOS, Mac, UNIX, and Ingres.]
Figure 3: Mapping Data Technologies to the Architecture

Data Architecture/Technology Map

- **Standard Collections**: Ingres, DB2-views
- **Local Collections**: Lotus, Excel, ...
- **Information Detail**: DB2, Ingres, ...
- **Operational (Enterprise)**: DB2, Ingres, VSAM, ...
- **Operational (Local)**: Ingres, Paradox, FoxPro, ...

CAUSE92: Information Architecture and Open Systems - Indiana University