

An Alternative Approach to Integrated Data Repository Design

- Rob Wynden – UCSF
- At UCSF a traditional data warehouse would not be well suited for the design of an Integrated Data Repository which merges Clinical, Health and Genomics Data Sources.
- An approach is required that does not require a centralized schema design.

Special Challenges at UCSF

- The special circumstances at UCSF suggest that an alternative approach to design is required.
 - The UCSF Integrated Data Repository will be comprised of over 50 data sources.
 - Many of our data sources are complex environments comprised of data from a wide variety of disciplines.

Domain Inconsistency

1. The Domain of Discourse for our Integrated Repository system will change over time as new databases are imported and as new research results in an ever updating & expanding ontology. (That would require constant maintenance within a traditional data model).
2. A relation requires that a column means the same thing over all rows. But the meaning of a column in our repository may depend on the Domain of Discourse of the user.

Use Case Examples

1. Two organizations may have different interpretations of the same Privacy Code.
2. All researchers within UCSF may not agree to use the same Ontology and there may not YET be a way to map those Ontologies.
3. What happens when a portion of an Ontology falls out of date?
4. Clinical Research and Health databases often interpret race and ethnicity in very different ways.
5. The ETL (Extract-Transform-Load) process required to map Ontologies (to resolve inconsistent terminology) would require too much manpower.

Design Challenges

- Schema design is difficult when different classes of users (using these normalized source schemas) consider different entities to be important.
 - For example: Public Health is not as concerned with Patient specific data and researchers focus on Cohorts. Yet in other circumstances Encounter would be a better intrinsic identifier.
- When creating a confluence of Clinical, Research and Genomic data there will be no consistent frame of reference from which to create a schema which is optimized for all users of the system.

Schema Abstraction

- An Abstraction layer for access to the database schema can offer a solution to these problems by allowing a design where the schema is constantly evolving.
- Our 2 central challenges are:
 - The domain inconsistency of our users
 - The operational inefficiency caused by providing detailed schema design and custom SQL queries to ease end user usage of our system.

2 Access Methods

- At UCSF we would like to support the creation of custom user interfaces for specific purposes.
- We must also support traditional commercial data warehouse and reporting systems (such as Cognos and SAS).
 - We May Never Alter Data (at UCSF we shall never change source data nor alter its interpretation). The researcher will sometimes require access to that source data.

Expert System Design

- **An Expert System layered above the data can resolve inconsistent domains.**
- **By providing a rules engine, different User Interface designs can be accommodated for users within different Domains.**
 - **For example, Cohort selection and browsing, or the discovery of common traits among unrelated patients for Clinical Research.**
- **Rules which establish the relationships between entities can be supplied for each user Domain (Clinical Research, Hospital Administration, Public Health etc...)**

Data Interpretation in Real-time

- **Data Interpretation only happens if it is deemed to be useful and happens as a result of a specific user request. (Data is not interpreted during ETL).**
- **Rules can drive the application UI to effect the presentation of the data based on Domain.**
 - **For example, Rules can be used to build a cohort discovery system.**
- **Rules type definitions include:**
 - **Fact Extraction (example, is the person a member of the Account?)**
 - **Ontology Mapping**
 - **Archetype Selection (for example, the person is disabled)**
 - **Metadata Rules (when was a particular dataset imported?)**
 - **Quality control (the detection of conflicts for follup via corrective action procedures)**

Data Harvesting

- **Rules can run recursively and be used to Harvest information based on a query of the rules engine.**
- **Repository harvesting can be run as a background process. (A good design for intelligent agent implementations)**
- **Harvested data can be stored within new tables (which would then allow users to later browse that collected data in real-time).**

Any Questions or Comments?

Open Discussion...