

NEW DATABASE DESIGN FOR THE APS SURVEY AND ALIGNMENT DATA*

*Jaromir M. Penicka, Horst W. Friedsam
Argonne National Laboratory, Argonne, Illinois, U.S.A.*

ABSTRACT

The conceptual design of the new APS Survey and Alignment Database, based on relational database technology, is presented in this paper. Authors outline the process of developing the logical database structure from data modeling to final quality check of the database design. The data used by the APS Survey and Alignment Group is analyzed and modeled in the entity-relationship model (E-R model). The E-R model is then transformed into a DBMS-independent relational design, and the quality of the database design is examined against the normalization theory. Authors also evaluate the relational design in the context of possible queries that might be posed to the database implemented on this design.

1. INTRODUCTION

Since its inception in 1990, the APS Survey and Alignment Group (SAG) depends on Geonet software [1] for its data storage and management needs. Geonet was originally developed under the DOS environment at SLAC in the 1980s as an all-inclusive software package for the accelerator alignment community. Geonet served us well for over a decade through the times of fast technological advancement not only in the area of MIS, but also in metrology. With the arrival of laser trackers, digital levels, etc., some Geonet utilities, like data collection and data reduction programs, slowly became obsolete and were phased out or replaced by new tools. Although we still keep Geonet data on our server and use it on PCs under the Windows operating system, the need for a new database concept replacing the obsolete DOS-based Geonet structure is imminent. The original data model, on which the Geonet database was built, does not accurately reflect our working environment any more. The capability to track new information, and greater flexibility in information retrieval and record update are just a few of those factors prompting the design and implementation of this new database concept. At the same time, we try to avoid any radical restructuring of existing data tables as we intend to migrate the old Geonet data to the new DBMS and would like to minimize the effort associated with this task.

After reviewing the current state of information technology, we have decided to choose the normalized relational model as an architectural standard because of its conceptual elegance, simplicity, mathematical foundation, powerful retrieval capability, flexible transaction protocols, and constructive design approach. The relational model also provides maximum independence between programs on the one hand and data representation on the other. Additionally, it supports a variety of very high-level data manipulation languages. The fact that the relational model is the foundation of the majority of today's commercially successful DBMS products also played an

* Work supported by U.S. DOE, Office of Basic Energy Sciences, under Contract No. W-31-109-ENG-38.

important role in our decision. The implementation of our database design in a well-supported software product with easy maintenance is an equally important goal of this project.

2. DATABASE DESIGN

The APS Survey and Alignment Group is responsible for precise positioning of beamline components for the APS accelerator systems. SAG fiducializes beamline components, aligns components in the accelerator tunnel, and maintains all necessary geodetic control networks to achieve required placement tolerances. The objective of this database design is to organize important information for the APS SAG in association with its tasks. Although the intention of this database is not to store raw survey data, the references and hyperlinks to the important unprocessed field measurements will be maintained within the database.

2.1 The Entity-Relationship Data Model

The reasonable beginning for database development is to build a data model that documents the entities to be represented in the database, establish relationships among them, and determine the characteristics of those entities that need to be stored. We utilized the E-R model as a modeling tool to express this logical organization of the APS SAG data. In the E-R diagram (Fig. 1), the entity classes are shown by rectangles with the name of the entity shown inside the rectangle, and relationships are indicated by diamonds with the name of the relationship shown near the diamonds. The maximum cardinality of the relationship is indicated inside the diamond, and the minimum cardinality is shown across the relationship line near the entity.

The model outlined in the E-R diagram in Figure 1 can be described as follows. There is an entity class COMPONENT and an entity class FIDUCIAL. The relationship between these two entities COMP-FID is one-to-many (1:N), indicating that each COMPONENT can have many FIDUCIALs, but every FIDUCIAL belongs to one COMPONENT only. The minimum cardinality of this relationship indicates that every COMPONENT must have at least one FIDUCIAL and each FIDUCIAL must have a COMPONENT to justify its existence in the database. The FIDUCIAL is a weak ID-dependent entity, and its identifier has to include an identifier of the COMPONENT. The FIDUCIAL has a many-to-one relationship with TARGET, meaning that each FIDUCIAL can accept only one TARGET at most, but one TARGET can be used on many FIDUCIALs. The minimum cardinality of the TARGET-FID relationship points out that FIDUCIAL does not need to have any TARGET and vice versa. The practical application of this allows theoretical and calculated points, like centers or vertices of magnets, to be included in the database. The term fiducial is used loosely to describe any component point. The entity COMPONENT has a many-to-many relationship with entity LATTICE, which suggests storage of historical records in the database. This aspect is especially valuable to the APS SAG because the APS storage ring is currently being realigned to a new lattice, and the majority of the components are being shifted to new locations. Booster realignment is also expected in the near future. Furthermore, it is worth noticing that the minimum cardinality of the ASSIGN-COMP relationship is zero, which suggests that a COMPONENT may not be assigned to any LATTICE point. This feature, combined with the selection of a serial number as an identifier for the COMPONENT, enables us to store spare fiducialized components in the database. There is also a one-to-many relationship between COMPONENT and ROLL, allowing for many instances of ROLL to exist for any instance of COMPONENT.

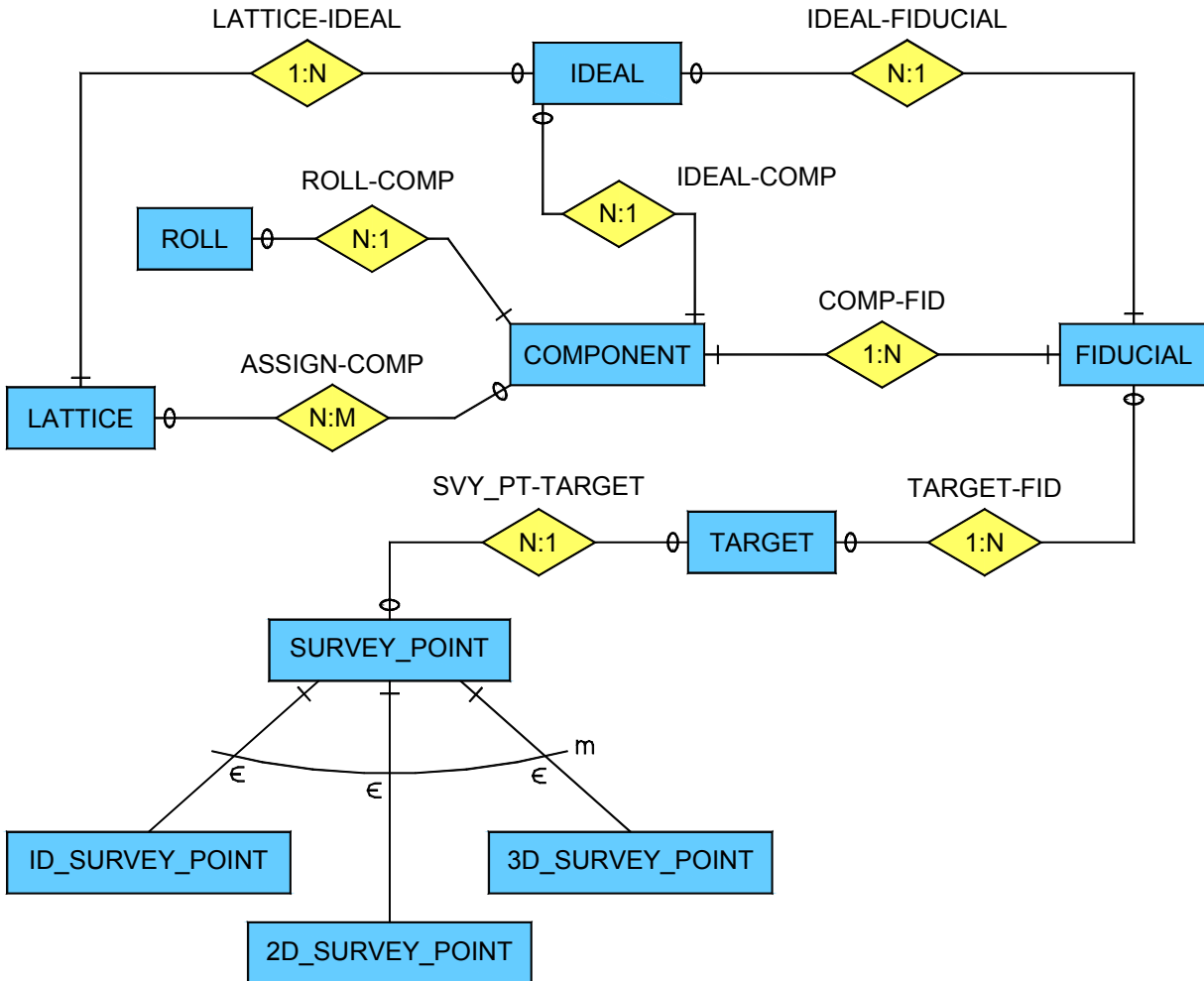


Figure 1 - Entity-Relationship Diagram

Continuing interpretation of the E-R diagram, one can see that the relationships LATTICE-IDEAL, IDEAL-FIDUCIAL, and IDEAL-COMP reflect the philosophy of allowing historical data to be stored in the database. The LATTICE can have many IDEALS, but each IDEAL belongs to exactly one LATTICE. In similar fashion, every FIDUCIAL can have several IDEALS, but every IDEAL belongs just to one FIDUCIAL and to one COMPONENT. There is also an entity class SURVEY_POINT for all measured points. The SURVEY_POINT has three defined subtypes: 1D_SURVEY_POINT, 2D_SURVEY_POINT, and 3D_SURVEY_POINT. The “m” indicates that SURVEY_POINT may belong to anywhere from one to many subtypes. The subtypes are not mutually exclusive, because one specific instance of a SURVEY_POINT can be measured with a level instrument as a 1D_SURVEY_POINT or with a laser tracker as a 3D_SURVEY_POINT. This generalization hierarchy contains an IS-A relationship, which implies that the subtypes have the same identifier as the supertype SURVEY_POINT, and they can also inherit many other attributes of the SURVEY_POINT. Notice that all other relationships in this model are of type HAS-A. HAS-A relationships connect entities of different types. The SURVEY_POINT can also be equipped with a TARGET in a similar fashion as the FIDUCIAL. The E-R diagram indicates that both SURVEY_POINT and FIDUCIAL can accept, at most, one

TARGET. This is a policy established by the APS SAG and, in most cases, enforced by the design of monuments and fiducials, which accept exclusively either 1.5”-diameter or 3.5”-diameter spheres as targets. The only potential problem is created by 0.25”- diameter bushings, sometimes used as fiducials by vendors. In this case, the policy of using only one type of tooling ball, or corner cube reflector, with a 0.3125” offset is enforced.

2.2 Relational Design

The entities and relationships identified in the E-R diagram were mapped to the relational design consisting of eleven normalized relations¹. The relations, including keys and all attributes, are listed below. Primary keys are underlined, and foreign keys are printed in italics.

LATTICE (*LatticePoint_ID*, LatticeType, Z_LAT, X_LAT, Y_LAT, Pitch, Roll, Yaw, Date)

SUBSYSTEM (*LatticePoint_ID*, Machine, Sector)

COMPONENT (Component_SN, Description, AlignSys, Comment)

FIDUCIAL (*Component_SN*, Fiducial_ID, z_comp, x_comp, y_comp, *Target_ID*)

ROLL (*Component_SN*, Roll_ID, Date, RollValue, Method, Comment)

IDEAL (*LatticePoint_ID*, *LatticeType*, *Component_SN*, *Fiducial_ID*, Z_APS, X_APS, Y_APS, AssignDate, Comment)

TARGET (Target_ID, Description, LevelOffset, TrackerOffset, SurfaceOffset, Image)

SURVEY_POINT (Point_ID, Machine, Sector, Description, *Target_ID*)

1D_SURVEY_POINT (*Point_ID*, Date, Y, SY, Method, NetworkOrder, LinkToFile)

2D_SURVEY_POINT (*Point_ID*, Date, Z, X, SZ, SX, Method, NetworkOrder, LinkToFile)

3D_SURVEY_POINT (*Point_ID*, Date, Z, X, Y, SZ, SX, SY, Method, NetworkOrder, LinkToFile)

The LATTICE relation contains three-dimensional geometric coordinates of the lattice points and the rotations for the beam components at these points. The Accelerator Physics Group generates this data. The primary key is a combination of a lattice point ID and a lattice type identifier. The combined key is necessary for distinguishing between two different lattices of the APS storage ring; the original lattice and the modified “Decker-distorted lattice.” The shifted lattice points retain their old IDs and therefore lose their capacity to uniquely identify tuples and determine the attributes in the tuples by themselves alone.

The purpose of the relation SUBSYSTEM is to quickly identify the location of a lattice point in terms of the accelerator subsystem and sector.

¹ A relation is a two-dimensional table with certain restrictions. We prefer to use rigorous relational mathematics terminology *relation*, *tuple*, and *attribute*, which are analogous to less strict terms *table*, *row*, and *column*. Some programmers favor similar terms *file*, *record*, and *field*.

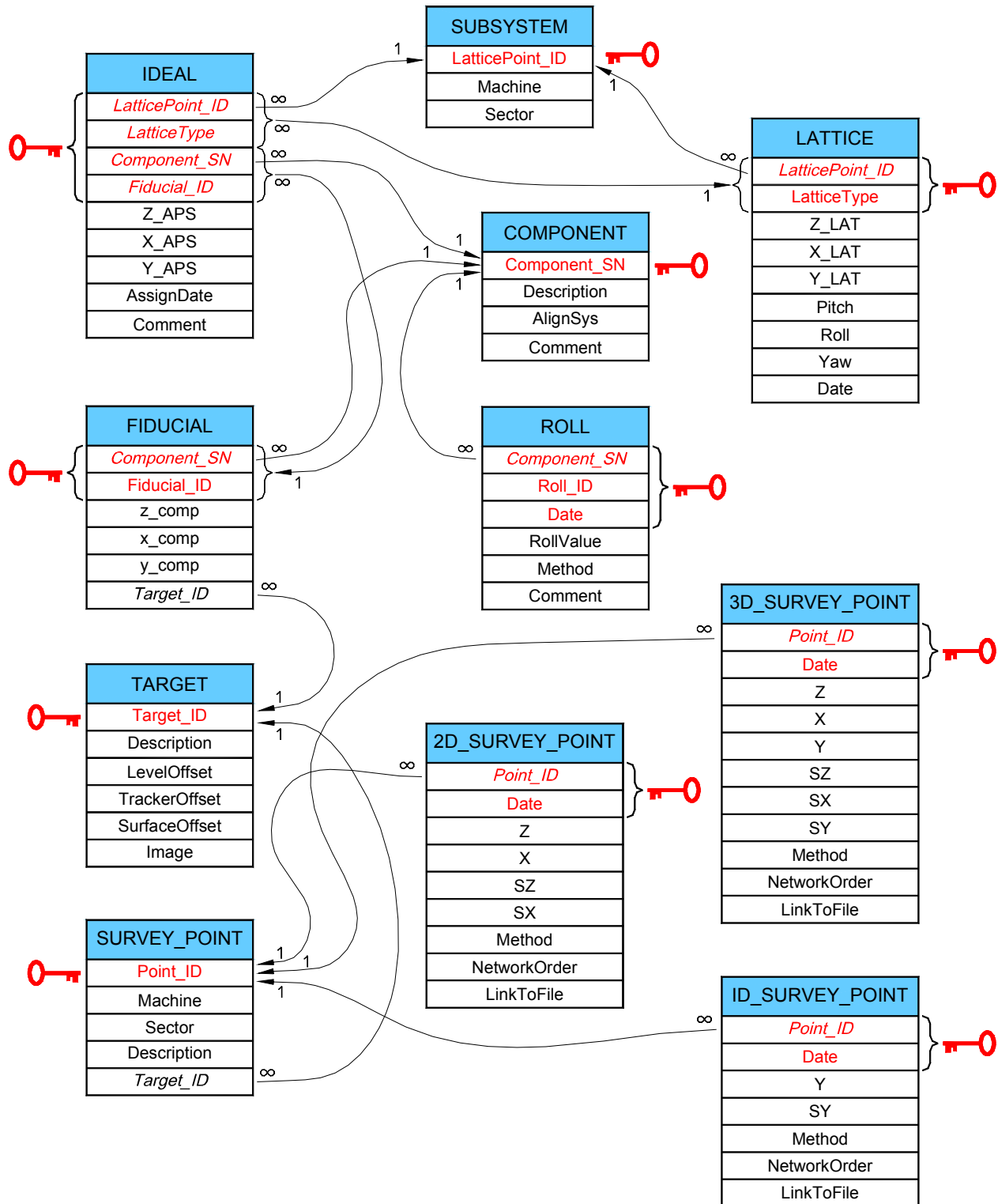


Figure 2 – Relational design

The objective of FIDUCIAL is to store the local component coordinates of a fiducial. The primary key is a composite of a fiducial ID and the serial number of the component to which the fiducial belongs. The foreign key, Target_ID, indicates what target is associated with this fiducial.

The relation COMPONENT, with a serial number of a component as a key, holds the common information about a component. For example, the fact that the component is a sextupole magnet with a three-point support push/pull alignment system, could be stored in this relation.

The ROLL relation contains roll values associated with some components. The primary key is composed from the component's serial number, a roll ID, and a measurement date. A roll ID is required because some components, for example dipoles, have multiple roll measurements associated with them for monitoring the twist. The date attribute added to the key again permits the storage of historical data. A method is included to specify how the roll value was determined.

The relation IDEAL has a dual function. First, it serves as an intersection relation to assign components and their fiducials to the specific lattice points. This assignment creates a unique primary key for IDEAL by using a combination of the primary keys from the LATTICE and FIDUCIAL relations. Second, the IDEAL relation is a good place for storing ideal geodetic coordinates of fiducial points, because IDEAL's primary key is a determinant of these coordinates.

The relation TARGET contains information related to the target used for point measurement. It defines the virtual point associated with the measurements or coordinates and specifies miscellaneous offset constants needed for reduction of different survey data to the same virtual point. Images of the targets are also stored in this relation.

The SURVEY_POINT relation maintains the common data for any measured point. It contains information such as location, description, and link to a target associated with the point. As a superclass relation to subtypes 1D_SURVEY_POINT, 2D_SURVEY_POINT, and 3D_SURVEY_POINT, it maintains all the universal information that can be inherited by the subtypes. In accelerator alignment processes the same points are measured repeatedly many times by many different methods, which justifies the existence of this relation separate from measured data.

The 1D_SURVEY_POINT, 2D_SURVEY_POINT, and 3D_SURVEY_POINT differ only in the number of coordinates and respective standard deviations they contain. The primary key is point ID in combination with measured date. These relations cover measured point coordinates, measuring methods, and order of survey networks. In addition, they hold hyperlinks to measurement data files, which will be stored in a hierarchical directory on a server.

The relational design showing the referential integrity is summarized in Figure 2. The primary keys in Figure 2 are indicated by red text, and foreign keys are printed in italics.

2.3 Database Design Evaluation

The relational database design is supported by well-thought-out formal theories of functional dependencies (FDs), normal forms (NFs), normalization, information-loss verification, dependency-loss verification, etc. We will briefly examine our design under the aspect of normalization, functional dependency, and functionality.

2.3.1 Functional Dependencies and Normalization

Analysis of the functional dependencies (FDs) within a database is one of the essential tools for working with relations and the process of normalization. Functional dependency is a term describing a situation, when one attribute or a set of attributes uniquely determines another attribute. More formally: let A and B be arbitrary sets in some relation R(ABCD). Then a functional dependency (FD) from A to B, denoted by $A \rightarrow B$, exists in R if for every A value that appears in R, the corresponding B value is unique. A functional dependency is a constraint on the possible relations in a relational schema.

Normalization is a process of lossless decomposition of larger composite relations into an equivalent set of smaller, simpler relations. The purpose of normalization is to eliminate or reduce redundancy and inconsistency of data, allow for a flexible update capability in the database, and identify the fundamental information building blocks for the construction of larger information objects. In the early years of database technology, extensive research went into establishing formal rules for checking the quality of a relational design. These rules were summarized in the definition of normal forms of relations: 1NF, 2NF, 3NF, BCNF, 4NF, and 5NF [2]. The normal forms have increasingly restrictive requirements and are nested, meaning that every relation in 3NF is also in 2NF, and every relation in 2NF is also in 1NF. The normal forms are basically restrictions on a relational database schema that prevent redundancy as well as anomalies in the database. If a relation is in one of these normal forms, we know that certain types of problems cannot happen. We also know whether we have a well-designed relational schema or whether we need to decompose it into smaller relations. Now let's examine our design.

A relation is in the First Normal Form (1NF) if each domain from which the attribute values are drawn contains only atomic values, not lists or sets. Our design satisfies this requirement.

The Second Normal Form is based on the concept of full dependency. To clarify the term full dependency, let A and B be attribute sets in a relation R. B is fully dependent on A if there exists a FD $A \rightarrow B$ and there is no other FD $A' \rightarrow B$ such that $A' \subset A$. Now a relation is considered to be in the Second Normal Form (2NF) if it is in 1NF and each of its non-key attributes is fully dependent on the key of the relation. All relations in the relational schema presented meet this condition. There is no attribute in any of our relations only partially dependent on a key.

A relation is in the Third Normal Form (3NF) if it is in 2NF and none of its non-key attributes is transitively dependent on any key. In other words, all the non-key attributes of a relation in 3NF are mutually independent. There is no transitive dependency in our relational schema.

With the Boyce-Codd Normal Form (BCNF) the terms “determinant” and “candidate key” have to be introduced. A determinant is an attribute, or a set of attributes, that determines some other attribute of a relation. A candidate key is an attribute, or a set of attributes, that can be a key of a relation. Now a relation is in BCNF if and only if every determinant in the relation is a candidate key. Consequently, if a relation is in BCNF, every attribute of every tuple keeps a bit of information that cannot be inferred from the values in all other attributes in the relation by using only FDs.

Our objective was to construct a database design with all relations in the Boyce-Codd Normal Form (BCNF) in order to eliminate all of the most common cases of insertion and deletion anomalies. Relations in the BCNF have no anomalies in regard to functional

dependencies. The BCNF guarantees that no redundancy can be detected using FD information. We believe that we have achieved at least the BCNF in our design.

Q1: “Where is the survey point “WALL127” located and what is the proper target for it?”

```
SELECT SURVEY_POINT.Point_ID, Machine, Sector, TARGET.Description
FROM TARGET, SURVEY_POINT
WHERE TARGET.Target_ID = SURVEY_POINT.Target_ID
AND SURVEY_POINT.Point_ID = "WALL127";
```

Q2: “Find all elevations and their standard deviations measured by the digital level after 1/1/2000 in Storage Ring sectors 10 thru 15.”

```
SELECT [1D_SURVEY_POINT].Point_ID, Date, Y, SY
FROM SURVEY_POINT, 1D_SURVEY_POINT
WHERE SURVEY_POINT.Point_ID = [1D_SURVEY_POINT].Point_ID
AND [1D_SURVEY_POINT].Date > #1/1/2000#
AND [1D_SURVEY_POINT].Method = "digital level"
AND SURVEY_POINT.Machine = "SRING"
AND SURVEY_POINT.Sector > 9 AND SURVEY_POINT.Sector < 16;
```

Q3: “When was the last time the 2nd order elevation network in the Booster, quadrant 4 was measured?”

```
SELECT MAX([1D_SURVEY_POINT].Date)
FROM SURVEY_POINT, 1D_SURVEY_POINT
WHERE SURVEY_POINT.Point_ID = [1D_SURVEY_POINT].Point_ID
AND [1D_SURVEY_POINT].NetworkOrder = 2
AND SURVEY_POINT.Machine = "BOOSTER"
AND SURVEY_POINT.Sector = 4;
```

Q4: “Give me the latest leveling data for any floor monument(s) in Storage Ring sector 3.”

```
SELECT [1D_SURVEY_POINT].Point_ID, Date, Y, SY
FROM SURVEY_POINT, 1D_SURVEY_POINT
WHERE SURVEY_POINT.Point_ID = [1D_SURVEY_POINT].Point_ID
AND SURVEY_POINT.Machine = "SRING"
AND SURVEY_POINT.Sector = 3
AND SURVEY_POINT.Description = "floor monument"
AND [1D_SURVEY_POINT].Date IN
  (SELECT MAX([1D_SURVEY_POINT].Date)
   FROM SURVEY_POINT, 1D_SURVEY_POINT
   WHERE SURVEY_POINT.Point_ID = [1D_SURVEY_POINT].Point_ID
   AND SURVEY_POINT.Machine = "SRING"
   AND SURVEY_POINT.Sector = 3
   AND SURVEY_POINT.Description = "floor monument");
```

Example 1

2.3.2 Functionality

For the evaluation of the database functionality, our design was analyzed in the context of possible queries that might be posed to the database. Or in other words, what questions could be answered from the database with the given relational design. To illustrate this process, four potential questions for the database are presented in Example 1. The four simple SQL queries indicate how to obtain answers from the underlying relational structure [3]. For a meaningful test of the logical design, one has to mentally pose a large number of such queries. But even this small example demonstrates the great data access flexibility of the relational model. Once the conceptual database is implemented in a DBMS, more frequently used queries would be parameterized and permanently saved in DBMS or embedded in application programs. The user will always have the option to type in SQL statements for some ad hoc queries, like the ones shown in Example 1, or query the database by example (QBE).

3. CONCLUSION

The proposed logical database concept will significantly increase retrieval and update flexibility of our data and eliminate a few redundancies and anomalies we still have in the current database. The new design captures more accurately the current stage of our ever-evolving working environment and new technologies. We believe that the presented logical database schema has no redundancies and no potential for anomalies in regard to functional dependencies.

The schema was developed without any particular Database Management Systems (DBMS) in mind. In the near future we will review commercially available DBMSs and implement our database design in a selected system. In addition to providing means to implement the database and to manipulate data efficiently, we expect the chosen DBMS to deliver physical data independence, concurrency control for access to shared data, automatic security enforcement, automatic integrity enforcement, transaction recovery, crash recovery, and other important features. Portability is also a very important aspect to the implementation. The selected DBMS should be portable from one platform to another with very little overhead. The successful realization of this database design in a suitable DBMS should bring us one step closer to our overall goal of developing a modular system for accelerator alignment that is easier to maintain in phase with technological progress.

4. REFERENCES

- [1] GEONET, *Stanford Linear Accelerator Center Survey and Alignment Workshop on Data Processing Using Geonet for Accelerator Alignment*, SLAC-PUB-395, February 1992.
- [2] W. Kent, *A Simple Guide to Five Normal Forms in Relational Database Theory*, Communications of the ACM, Vol. 26, No. 2., pp 120-125, February 1983.
- [3] J. Melton and A. Simon, *Understanding the New SQL: A Complete Guide.*, Morgan Kaufmann, 1993.