In the areas of sciences, engineering and design, the entities experience both structural and state changes. Existing database models allow changes in the structure and state through the techniques of schema evolution and version management, respectively, but fail to capture and represent both of these changes together and at the same time. In this paper we formally introduce the Temporal Object-Oriented System (TOS) which allows simultaneous changes in both the structure and state of the objects. An object in TOS is called a temporal object (TO). An aggregation of different TOs of the system can give rise to a new object which is then referred to as a temporal complex object (TCO). Both temporal objects and temporal complex objects can replace and refresh their knowledge through a process called renovation. The paper also discusses the process of renovations for both TOs and TCOS and enumerate their temporal parameters (i.e., time-span, life-span and life-sequence). A query language for the TOS, called Temporal Object Query Language (TOQL), is also proposed which is a super-set of the relational query language SQL.

During its life time, a database goes through many evolutionary processes. This evolution takes place both at tuple level and schema level. Generally, database systems support the notion of non-temporal evolution of tuples and schema to accommodate changes in the state and structure of an object using the concept of “current view” only (Hurson, 1993; Ling, 1992; Pissinou, 1993; Shoshani, 1984). In these databases, only existing data values are kept using the available schema definitions, and therefore, direct access to past information is not available.

In many database applications such as scientific experiments and studies, statistical surveys and CAD/CAM, the information evolves with time. Therefore, it is not desirable to discard the old information. For example, in scientific experiments and studies it is quite common to have a researcher or a team of researchers working on a specific project and pursue different alternative methodologies and experimental procedures to reach the goal and later decide to choose the best one or to come up with some new method by taking into account many individual experiments and their mutual trade offs. The design environment is another example in which temporal information of the data needs to be preserved. In all such cases, it is necessary to either indicate the time at which data was collected/generated or the time interval for which it is valid.

There is a great deal of research in the area of Temporal Databases (Kline, 1993). The vast majority of research on temporal database systems is on relational and pseudo-relational database systems (Pissinou, 1993). In all such databases, time dimension is added either at the attribute level (Clifford, 1982) or the tuple level (Gadia, 1991) to keep the history of the data objects and is modeled either as a time point or a time interval. Both time models are considered equivalent (Dutta, 1989). The value of time associated with a data object is determined by the system or assigned by a user. If a time value is assigned by the system, then it is referred to as a physical time such as transaction time, while if it is assigned by a user, then it is referred to as a logical time such as user-defined time (Ling, 1990).

Relational databases and their temporal extensions are suitable for simple fixed-format record-base applications. However, scientific and statistical data and design entities are complex in nature, multidimensional, unstructured and possess variable format and length (Hurson, 1993; Shoshani, 1986; Zhao, 1993). Therefore, such temporal data models are
unsuitable for scientific, statistical, engineering and design applications (Hurson, 1993; Shoshani, 1986; Maier, 1986).

Among the database researchers, there is a great deal of consensus that object-oriented databases provide well suited conceptual data modeling constructs for efficient and natural representation and manipulation of such data intensive applications (Beech, 1988; Hurson, 1993; Michalewics, 1991; Shoshani, 1986). Several type of object-oriented data models have been proposed to support the development of such applications and temporal information (see (Kline, 1993) for listing).

In the object-oriented paradigm, an object is defined by two parameters: structure and state. The structure (SR) of an object provides the structural and behavioral capabilities of that object, which is defined by a set of instance variables, methods, and rules (or integrity constraints). The state (ST) of an object assigns data values to the instance variables of the objects and methods which operate on them. A set of objects sharing the same structure is referred to as a class. An object-oriented database is a collection of classes which are organized as a directed acyclic graph (DAG). Figure 1 shows the schema of an Automobile Manufacturing Database. In this figure bold lines represent part-of relationships.

In existing object-oriented database systems, the concepts of temporal evolution of data are not well defined. However, changes in the state and structure (class) of an object are maintained via version management (Agrawal, 1991; Beech, 1988; Katz, 1986) and schema evolution (Nguyen, 1989) respectively. Roddick (Roddick, 1992) provides a bibliography of papers on the subject of schema evolution. Schema or class evolution in databases is generally a less frequent phenomena and since only recent past has got considerable attention of database researchers. Schema or class evolution normally takes place due to changes in database requirements or computational environment. There are three possible scenarios for a class to change its structure. These are:

- type I: Adding new instance variables, methods, and/or rules
- type II: Deleting instance variables, methods, and/or rules
- type III: Changes to an instance variable, a method, and/or a rule

For type I changes, there is no loss of knowledge of a class because the previous knowledge of the class structure is also retained along with the new one. On the other hand, for type II and type III changes, the history of changes in a class structure is not readily available, as it is overwritten or deleted in the latest version of the class structure. In current object-oriented database systems, only the current version of each class structure is retained. After any one of the type II or type III changes, it is necessary to reload a previous version of the database to retrieve any information from a previous version of a class structure. For example, Figure 1 and Figure 2 are two snapshots of the class structure for the class Vehicle at time instances t1 and t2 respectively where t1 < t2. In Figure 2, the instance variable carburetor of Figure 1 in the class Engine is replaced by a new instance variable Fuel-Injector. After this change, all queries about the class Engine can be answered from its current state of the class structure (Figure 2) unless the user explicitly asks for some previous state of class Engine.

As we mentioned, although there is a great deal of research on temporal databases, version management and schema evolution, however, a survey of literature does not indicate a significant comprehensive data model to efficiently capture, represent and manipulate temporal data and changes in the structure and state of such objects, as required in the areas of sciences, engineering, statistics and design. In this paper we introduce a more general system, the temporal object-oriented system (TOS) and its query language for representation and manipulation of data from all of the above mentioned disciplines. This data model is capable of representing both traditional non-temporal as well as temporal information.

In the following section, we describe the temporal object-oriented system (TOS) and its various components. Subsequent sections contain the discussion about the renovation of temporal complex objects and the syntax for a query language for TOS. In the last section, we present our conclusions and provide future research directions.