**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **08-July-2020** | **Name:** | **Bhuvanesh M** |
| **Course:** | **Coursera** | **USN:** | **4AL16EC015** |
| **Topic:** | **The Evolution of Data Models** | **Semester & Section:** | **8th sem & ‘A’ section** |
| **Github Repository:** | **Bhuvan** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| **Report –** In my first session today I have studied about -The Evolution of Data Models Motivation Relational data models with SQL as data definition and data manipulation language are a quasi-standard for commercially available data base systems, but research in data models and interfaces to database systems seems to diverge into different directions rather than converging to a new, higher-level object model which could serve as a new platform for the non-traditional applications as well as for the classical ones. Object-oriented databases often start from object-oriented programming languages and add persistency as well as other database functions such as transaction management. Objects (instances of abstract data types) are encapsulated and only "accessible" through a well-defined interface. Object manipulation is by invoking type-specific interface functions (methods). The advantages are twofold: first, the structure of objects is hidden (which provides a higher level of abstraction) and second, the methods can implement integrity checks that are specific to the object type.  Typically, OODBMS have a complete programming language environment, but provide little means of descriptive set-operations [GM88]. This reflects their origin from the programming language realm. As there are many programming languages, we expect as many database systems and the integration aspect of databases across several applications written in different languages seems to be lost or at least more difficult. Complex Objects have evolved from relations in that they are constructed by repeated application of tuple and set constructors. Nested Relations as a special case of Complex Objects have been studied 136 in detail in their theoretical as well as practical issues during the past few years [AFS89]. One of the major strong points is the fact that they preserve the descriptive set-oriented style of relational query languages.  Like in many other models, a variety of query languages have been proposed for nested relations and complex objects, such that it might not seem justified to consider this direction as one approach. However, the essential ingredients in all of them are actually the same. Therefore, like in the classical relational model, we can assume one - may be hypothetical - algebra as their common basis. Furthermore, it seems that shared subobjects or recursively structured objects are incompatible with the hierarchical organization and hence can not be brought under the same unifying umbrella. Adding generalization or specialization known from semantic data models [HK87] or similar notions from knowledge representation schemes ultimately seems to increase the divergence.  While such mechanisms facilitate the definition of new types and gives rise to inheritance of attributes, methods, or functions, making query formulation easier, they also need new constructs in queries, such as type predicates and update operations for changing the type of an object. Again, many proposals exist that look pretty incompatible. Furthermore, a number of problems with the formal foundations still lack solutions, for instance how to handle polymorphic types, and how to deal with the higher-order syntactic constructs in a first-order logical framework [Bee89, KL89]. Therefore, in view of all this it appears hopeless to obtain a higher-level unifying data model as a new platform which contains the existing data models as special (degenerate) cases.  However, in this paper we will put together some evidence that the seemingly different concepts are not really incompatible by their very nature. While there exist some differences with respect to formal definitions and in the terminology, we try to show that the essential features are very similar with regard to their behaviour for a user. We think that the differences are not exactly" ... just silly exercises in surface syntax" [SRH90], but we do argue that, apart from the diverging development, we can - in retrospect - interpret them in a consistent way. In order to explain this we describe two evolutionary ways: the first starts with the relational data model and encompasses nested relations, complex objects and object networks ("structural" object-orientation [Dit86]).  The second one starts with the DBTG network model and leads directly to the structural part of object networks. As a result of this exercise we see how recent object models can be obtained as a synthesis of wellestablished concepts, namely  (1) set-oriented, descriptive query and update languages from relations and nested relations,  (2) recursive schema definitions from the network model to allow for object sharing,  (3) behavioral abstraction from abstract data types providing encapsulation (methods), and  (4) inheritance through the specialization of types. Notice that object methods also provide extensibility and adaptation to application classes.  The practical aspect of such a synthesis is that classical data models are contained as special cases. For example standard SQL can be extended to SQL for complex objects. This, in turn, can be generalized to SQL for object networks, and opened to object-specific methods. Thus, a kind of upward-compatibility is achievable which facilitates cooperation between different systems. We do not describe a specific object model here, although we use our notation of [SS90b] and [SS90a] where appropriate, for explanation purposes and as an example.  Our principal goal, however, is to point out how the salient features of recent research proposals are obtained as an evolution from well-known classical ones, and what features have been added to them. In fact, it turns out that apart from all terminological and syntactical differences, a substantial body of core concepts can be found in all of them. The research proposals we have in mind include the ORION query language described in [BKK88, Kim89], the OSQL language of the IRIS project [Bee88, WLH90], the PROBE language [MD86], the 137 object-oriented extension OODAPLEX of the Functional Data Model [Day891, the EXCESS language and EXTRA model of EXODUS [CDV881, the data model used in the O2 project [D+90, LR891, the LauRel project [Lar88], the MAD model [Mit87] with its MQL query language, and the HDBL language [PT86] of the AIM prototype system. Previous work on evolutionary aspects includes [Bee88] and [SS90b]. |