Database Engineering Focusing on Modern Dynamism Crises

Luiz Camolesi, Jr.
Methodist University of Piracicaba, Brazil

Marina Teresa Pires Vieira
Methodist University of Piracicaba, Brazil

INTRODUCTION

Researchers in several areas (sociology, philosophy, and psychology), among them Herbert Spencer and Abraham Maslow, attribute human actions resulting in continual environmental changes to the search for the satisfaction of individual and collective needs. In other fields of science, this behavior represents a challenge in ethical researches on concepts, methodologies, and technologies aimed at optimizing and qualifying the actions involved in these continual changes to obtain better results.

Specifically in computer science, software engineering is a critical sub-area for these researches and their application (Lehman & Stenning, 1997), since it involves the construction of models and orientation for their use in the development of resources, such as software, to support the user’s needs (Perry & Staudenmayer, 1994). Databases are included in this context as a component for data storage.

Considering the premise of continuous changes (Table 1) and the human needs involved (Khan & Khang, 2004), the consequences for software and for the database used are obvious. In the field of computational science, these changes in the modern world are reflected in evolutionary features for software and databases (Brereton, Budgen & Bennet, 1999), based on database concepts, structures, and processes that allow for rapid, albeit not traumatic, shifts to new industrial, commercial, or scientific systems (McFadden, Hoffer & Prescott, 1999) in new contexts (temporal scenarios) (Camolesi, 2004).

Table 1. Types of changes

| Evolution | actions for an (variant) element’s technological progress, improvement, modernization, or correction. |
| Revolution | alteration actions of an element can influence the element’s purpose in the context. |
| Involution | simplification actions of an element, regression in its conception or content. |

BACKGROUND

Database models must comprise representation elements that are adaptable to the user’s varying and dynamic needs, and contain the taxonomy needed for their manipulation. Thus, traditional (generic) database models such as the Entity-Relationship (ERM) and Relational (RM) models (Siau, 2004) have been expanded with appropriate “profile” for specific applications and requirements. Considering their purpose of supporting changes, “profile models” can be easily referenced in scientific researches as:

- **Version Model:** considering versions as database objects derived (originating from, but containing alterations) from others, models of this profile must be applied to a database characterized by the explicit and voluntary storage of the historical information about object changes (Conradi & Westfechtel, 1998). The features frequently specified in versions models are:
  - **Derivation Structure:** establishes the data structure for organizing versions, for example, stack, tree, or not cyclic digraph, linked by special relationships representing linear or nonlinear derivation actions;
  - **Versionable Element:** establishes which variant elements (database objects) can have versions created and represented in the database;
  - **Property of Versioning:** this is a feature that serves to define a versionable element. This
property must be dynamically established for each element during either its creation or its definition;

- **Version Status**: status set (state or situation) for the versions;
- **Manipulation of Versions**: the creation, update, and deletion of versions can be accomplished implicitly either by the database system or by the user through specific command language;
- **Operation Restrictions**: the manipulation of versions by users can be granted unconditionally, or restrictions may be imposed for each operation (create, update, and delete).

- **Time Model**: in models of this profile, the elements that represent the dimension time are established essentially to control the evolution of the database. The reliability of such time-based models depends on the unambiguous definition of temporal limits to be imposed in any business or scientific database system. In an evaluation of systems using the time representation, database designers find many variations and ambiguous representations that can degenerate the processing of the time value simply because they are ignorant of how time is represented, how it can be analyzed, or how it should be converted. Using a Time Model for the homogeneous representation of the data type `time` and `interval` enables the designer to improve the evolution control performance. Based on many researches about time representation and utilization (Allen, 1991; Bettini et al., 1998), the following features are identified for a homogeneous data type definition:

  - **Moment**: a time instant value;
  - **Granularity**: precision domain of time instant, this feature can be based on the ISO 8601 (International Organization for Standardization, 2000) standard, for example, PnYnMnDTnHnMnS, or any other standard established by the application;
  - **Orientation**: reference system for temporal representation, for example, Gregorian calendar (UTC or Coordinated Universal Time), Chinese calendar, Jewish calendar or others;
  - **Direction**: all orientation has a moment of origin (0), and a time may be the moment preceding or following this origin moment;
  - **Application**: specification of the use of the temporal representation, allowing for the semantic recognition of the type, independently of the context in which it is inserted. The attributes `application` should indicate one value as:
    - **Occurrence**: to specify a moment (time or interval) to carry through an action (either a past situation or a future one);
    - **Duration**: using time or interval to specify the duration of an action (either a past situation or a future one);
    - **Frequency**: to specify a moment (time or interval) used to record repetitions.

- **Configuration Model**: this “profile model” is based on and related to the version model, with version aggregations defined as `configurations` or `releases` (Conradi & Westfechtel, 1998). The `configurations` or `releases` are logical aggregation components or artifacts, selected and arranged to satisfy the needs of applications based on composition abstractions (Sabin & Weigel, 1998). Applications that require the composition of database objects are related with engineering, that is, Computer Aided Software Engineering (CASE) and Computer Aided Design (CAD);

- **Integrity Model**: required in all data models (RM, OOM, and ORM), elements are established in models of this profile to support data consistency and integrity. In certain typical databases, the number of constraints and actions for database integrity may involve hundreds of elements, which require periodical reviews since they are strictly related to continually changing real situations. The elements in these models vary in form and purpose, the most common being rules, `business rules` (Date, 2000) and `database constraints` (Doorn & Rivero, 2002; OMG, 2003);

- **User Model**: necessary in all data models (RM, OOM, and ORM), the elements that represent the users are established in models of this profile. The modeling of user features is critical in the evolution of a database because it involves a diversity of needs and changes in the operational behavior (insert, delete, alter, and select) in the database. The modeling of human behavior to design access privileges is a well-consolidated analytical process (Middleton, Shadbolt & Roure, 2004). However, this process must take into account the static and dynamic aspects of people and their activities (Perry & Staudenmayer, 1994). Analyses based on a static approach define `User Roles` that are appropriate for traditional applications, whose activities change with relative infrequency. In dynamic applications, however, static definitions are insufficient, since the requirements call for temporary and specific activities that are necessary to support the dynamic definition of `User Roles`. 

