Sahil Jindal
2013CS10253

COL 362 Assignment 1

3 Level Architecture

1. Does 3-level architecture solve all the problems?

Ans. The 3-Level architecture solves several problems in Database Management, and offers several advantages:

- The main advantage of 3-level architecture is Data Independence. Schema at one level of database can be changed without changing the schema at other levels.
- Redundancy can be reduced.
- Inconsistencies can be avoided.
- It makes the database abstract, hiding the details of physical data storage from the user, making it easier to understand and use, and allowing user to concentrate on data rather than worrying about how it should be stored.
- Improved Security: Client cannot directly access database.
- The 3-level architecture allows migration to be seamless. The database appears same on different systems, even if the physical storage is changed.
- The model allows database admin to change storage medium of the database without disturbing the user currently on the system.

However some disadvantages that still persist with the 3-level architecture:

- Increased Complexity/Effort.
- Investment in upgrading the architecture from previous model.

2. How can 3 level architecture be implented? What transformations and data structures are needed?

Ans. To implement 3-Level Architecture, we first implement the 3 different level of the model and then implement the mappings between different levels.

Implementation of levels:

- **External Level:** All end user languages include a data sublanguage – a subset of the total language that is specifically concerned with database objects and operations. The Data SubLanguage (DSL) is embedded within the corresponding host language, which is responsible for providing various non-database facilities. Each external view is defined by external schema, which consists of definitions of each of the various external record types. The external schema is written using the DDL portion of the user's data sub-language(external DDL).
- **Conceptual Level:** The conceptual view consists of many types of conceptual record. The conceptual view is defined by means of the conceptual schema, which includes definitions of each of various conceptual records. The conceptual schema is written using another data definition language , the conceptual DDL.
- **Internal Level:** Internal view consists of occurrences of many types of internal records. The internal view effectively assumes an unbounded linear address space. It is described by the means of internal schema., which not only specify various record types but also specifies what indexes exist, how stored fields are represented, what physical sequence the stored records are in.

Mappings:

- The **Conceptual/Internal mapping** defines the correspondence between the conceptual view and the stored database. It represents how conceptual records and fields are stored at internal level.
- The **External/Conceptual mapping** defines the correspondence between a particular external view and the conceptual view.

Various data structures used in implementing 3-level architecture include B-Trees, and hash maps.

3. Does 3 level architecture support more than 1 programming language?

Ans. The 3-Level Architecture does support various programming languages. In external view, for application programmer, language used will be either a conventional programming (e.g. Java, C++, PL/I) or a proprietary language

that is specific to the system in the question. For the end user, the language will either be a query language(SQL) or some special-purpose language, tailored to that user's requirements.
All these languages include a data sub-language – that is, a subset of the total language that is concerned specifically with database objects and operations. The Data Sub-Language is embedded within the corresponding host language. The host language is responsible for providing various non-database facilities, such as local variables, computation operations, branching logic, and so on. A given system might support any number of host languages and any number of data sub-languages.

4. How is data independence achieved?
Ans. There are two kinds of data independence:

- **Physical Data Independence:** Physical data independence indicates that physical storage structures or devices could be changed without affecting conceptual schema. Physical independence is achieved by the presence of the internal level of the database and the transformation from the conceptual level of the database to the internal level. Conceptual level to internal level mapping therefore provides a means to go from the conceptual view to internal view. To achieve physical data independence, conceptual DDL definitions must not involve any considerations of physical representation or access technique.
- **Logical Data Independence:** Logical Data Independence indicates that the conceptual schema can be changed without affecting the existing external schemas. Logical data independence also insulates application programs from operations such as combining of two or more records. This would require a change in the external/conceptual mapping so as to leave the external view unchanged. The logical data independence is difficult to achieve than Physical data independence as it requires the flexibility in design of database and programmer has to foresee the future requirements or modifications in the design.