A Fast Database Generator

Shih Lin

Computing Center, Academia Sinica
Jan 15, 2000
Why Fast Database Generator (FDG)

- Increasing requirements for databases
- Increasing complexity from
  - Better understanding of database capability
  - Deeper insight of data from metadata
  - XML/XML Schema for describing complexity
- Higher possibility of change
Why Is FDG Possible

• Limited types of composing elements -- no matter how sophisticated the data schema looks like
• Limited data management and query patterns for each elements and specific combinations of elements -- in some scope
Composing Elements of Schema

- Units of structure
- Data types
- Data validating conditions
Units of structure

- **Entity** -- collection of related fields
- **Field**
  - Primitive data element
  - Nested entity -- forming hierarchy
  - link -- referring to other entity
- **Scale of root entity and field**
  - unary -- may or may not be null
  - multiple -- fixed or variable number of items
Scale and Relation

- Entity to unary sub-entity -- one to one
- Entity to multi-value field (except link) -- one to many
- Linked entity to unary linking entity -- one to many
- Linked entity to multi-value linking entity -- many to many
Data types

• Primitive type
  – Numeric -- integer, decimal and floating point
  – String -- fixed and variable length
  – Time -- date and timestamp
  – Boolean
Data types (continue)

- Enumeration
  - Listing keywords and associated codes
  - Single or multiple levels
  - Static or dynamic
- Entity or enumeration definition with type name
- Link
Data types (continue)

- Full text
- Application type -- e.g. user ID
- Multimedia -- image, audio, video, etc
Scope Fitting FDG

- Frequently used logic
- Data entry and maintenance
  - Integration of insertion, deletion and update
  - Data validation
  - Data integrity and consistence
- Query by example
- Internet resources from databases
  - e.g. bibliography, fish or plant
System Components

- Schema language and parser
- Database creation script generator
- Data management/query interface generator
- Report language and report generator
- Core system
  - Toolbox on browser
  - Intermediate server
  - DBMS
System Features

- The schema language is expected to be easy to learn and easy to use, and yet powerful enough to describe complicated schemata.
- User requirements as well as the associated schemata may change frequently and heavily before the data is entered.
- The schema language, generated interfaces and report language are DBMS neutral.
- FDG utilizes DBMS, but it does not have to be restricted to any special DBMS.