DIALOG INTERFACE FOR DYNAMIC DATA MODELS

Mgr. Vojtěch Přehnal

Presentation outline

Introduction

- **New approach to information system development**
- Aimed at:
  - large-scale time-varying data models
  - 3-tier client-server architecture
  - relational DB
- Platform independent methodology
  - various operating systems
  - various application servers
  - various database engines

Terminology

- **Item**: an ordered multi-set of related data values. The rank of each data value is referred to as an **ordinal**
- **Field**: a multi-set of data values with the same ordinal and some common attributes (type, size, nullability, identity, …)
- **Table**: a multi-set of data values organized using a model of **items** (rows, records) and **fields** (columns)
- **Data model**: a set of tables and their relations

Terminology

- **Dynamic data model**: data model with **time-varying** data structure, fully editable at the run-time
- **Dialog interface**: communication protocol between client and server
- **Meta-data**: supplementary information defining the structure and the attributes of the raw data
- **CRUD operations**: standard database operations with the items: **C**reate (insert), **R**ead (select), **U**pdate and **D**elete.

Motivation

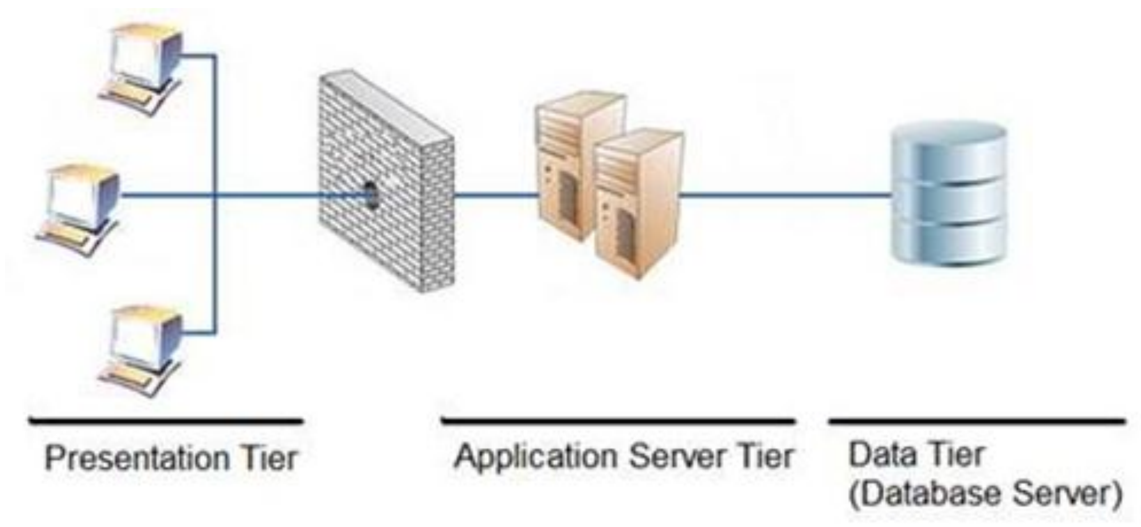
- Information system for ZONER software, a.s.
  - 3 divisions (Internet Services, Software, Zoner Press)
  - 15 branches world-wide
  - over 120 empoyees
  - over **350** tables with common CRUD operations
  - only **8** tables with specialized logic (!)
  - tens of millions of items
  - over 40 GB of (textual) data
  - average time to data-model adjustment:
    only 14 hours (~ cca. 2 adjustments / day)

Objectives

- **Enable editing data model at run-time**
- **Automate** the most common practices
- **Separate** common and specialized logic
- **Lower** the costs of SW development
- **Preserve** performance, scalability and adaptability of traditional approach

3-tier architecture

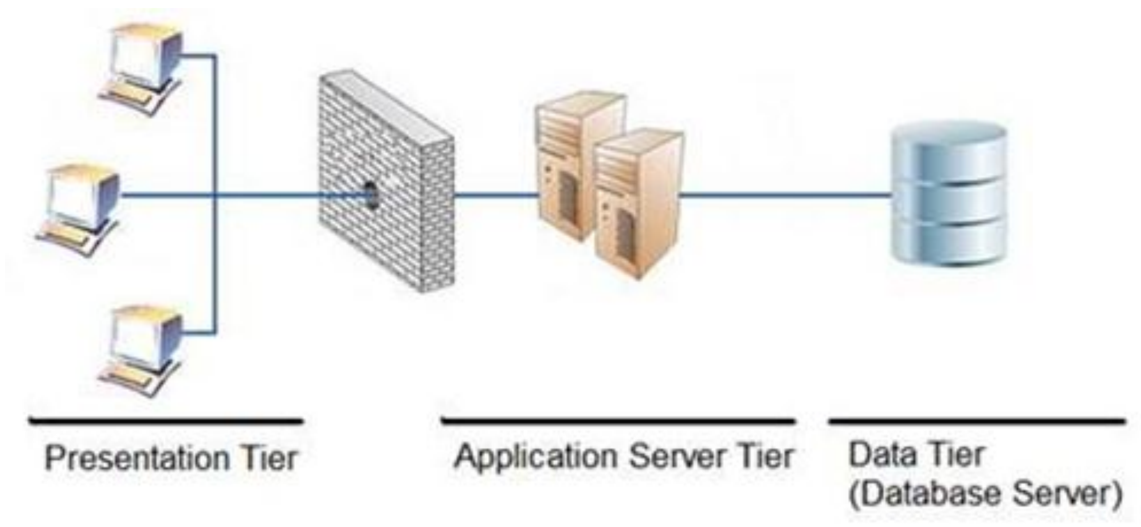
- means **three-tier development**



change ⟵ **change** ⟵ change

3-tier architecture

- How to automate changes in application and presentation tier according to changes in DB?



~~change~~ ← ~~change~~ ← change

State of art

- Compile-time development automatization
  - **automated source code generation**:
    - automated class model generation
      - ORM tools (ADO.NET, Linq2Sql, Telerik OpenAccess, …)
    - automated application logic generation
      - Microsoft WCF RIA Services, Picasso, Habanero, …
    - automated application and UI code generation
      - Microsoft ASP.NET Dynamic Data, Microsoft WebMatrix

👍 auto-generated source code can be extented with specialized logic **arbitrarily**

Automated source code generation

- Limitations:
  - source code based on fixed data model ⇨ **rebuild required** for every change in data-model
  - common and specialized logic is mixed together ⇨ **loss of specialized business logic** after re-generating source code
  - performing **uniform changes** in common functionality for all the tables requires **rewriting source code for each individual table**
  - efficient only for systems with **lower number of tables** and with **majority of specialized functionality**

Goals and gains

- Dialog interface[1] for:
  - interactive data-model exploration
  - automated user interface generation
  - data retrieval, validation and change submission (CRUD ops)
  - automated data log creation
  - automated SQL-injection protection
  - centralized UAC (user access control) management
  - data model and user interface localization

---

[1] Dialog interface = communication protocol between client and server:

- **data model** of messages exchanged between client and server
- **sequential model** of these messages (their arrangement in time)

Conclusions

- New methodology for information system development
- Key point: **retrieving data model at the runtime**
- No brand-new revolutionary techniques
  (instead, utilizing well-known techniques for new purposes)
- Main advantages:
  - ability to **edit data model on the fly**
  - **no need of rebuilding** or restarting the running application
  - **no need of rewriting source code** of common operations for each individual table
  - **suitable** for the vast majority of data-driven apps
  - easily **extensible** for individual user´s needs

Future work

- **Study** of various business scenarios
- **Design** of a meta-data model and a dialog interface to pass the broadest possible scale of user requirements
- Simple **application** for demonstration purposes
- **Presentation** in science community

Discussion

- Comments and questions