Relational Database Systems I

Christoph Lofi
Simon Barthel
Institut für Informationssysteme
Technische Universität Braunschweig
www.ifis.cs.tu-bs.de
• **View integration**
• Resolving conceptual incompatibility
• Entity clustering for ER models
• Commercial dimension: The BEA story
4.1 Business Integration

- Business currently is a world of M&A
  - companies need to diversify/enhance their portfolio
  - but it is expensive to develop necessary applications
    - knowledge gathering costs time
    - will the output be worth it?
  - **idea:** rely on people who are *already knowledgeable*
    - acquire small, specialized, and promising companies
    - merge with big players in the field
4.1 Business Integration

• Examples
  – the Daimler-Chrysler merger
  – the Oracle-Sun merger
  – Oracle buys PeopleSoft, Siebel Systems, BEA Systems, …
  • Siebel Sales as CRM tool now part of Oracle’s business intelligence suite
4.1 Business Integration

- Merged (parts of) businesses are administrated by
  - different specialized software systems?
  - one company-wide system?
- Usually there is an historical evolution of separate tools and programs
  - e.g. accounting, sales & marketing, development
  - based on individual requirements
- However, often a unified view is needed
  - e.g. for business intelligence and warehousing
    - Data Warehousing is also a great lecture at ifis…
4.1 View Integration

• Usually, there are several conceptual schemas
  – several designers are part of the modeling process (modular software development)
  – different tasks were modeled within the same organization (legacy systems)
  – several organizations need to be integrated (business integration)
4.1 View Integration

• View Integration
  – several conceptual schemas need to be combined into a **unified global schema**
  – all differences in perspective and terminology have to be resolved
  – all redundancy has to be removed

• But,… what happens, if you don’t integrate?!
• **Example:** Big NASA project **Mars Surveyor**
  – the 1998 missions investigated *Volatile and Climate History* on Mars
    • characterization of climate change and its evolving impact on the distribution of water
    • **idea:** explore the **polar ice caps** of Mars and see whether there is water ice
    • about three months of experiments on Mars were scheduled
• Two vehicles: Mars climate orbiter and Mars polar lander
  – the lander was supposed to probe the layers of ice and dust on the polar ice caps to investigate changes
  – the orbiter was built to monitor the daily weather and record changes in water vapor and dust in the atmosphere
4.1 The Mars Desaster

• Catastrophic failure
  – the Mars climate orbiter approached Mars up to 57 km instead of 150 km, and was destroyed in the atmosphere on September 23, 1999
  – the Mars polar lander crashed during its attempted landing on Mars, December 3, 1999
  – $327.6 million in total for both
    • $193.1 million for spacecraft development
    $ 91.7 million for launch
    $ 42.8 million for mission operations
Why did the **climate orbiter** come too close to Mars’ atmosphere?

- **many organizations** were involved in the development
- there was no global schema
  - navigation software produced by Lockheed Martin used **non-metric** units (i.e. inches, feet, and pounds)
  - NASA used **metric** units
  - a small correction of the course led to the fatally low orbit…
• Happy ending!
  – the next try was the successful 2001 Mars Odyssey
  – the measurements pointed to water ice on Mars
  – confirmed by the Mars Express (ESA) in 2004
    • the polar caps consist of 85% carbon dioxide (CO$_2$) ice and 15% water ice
4.1 The Mars Desaster

– In 2012 Curiosity landed on Mars
  • Investigation of climate and geology
  • Did the surface of Mars ever offer conditions favorable for microbial life?
    – Investigation of the role of water
  • Planetary habitability studies in preparation for future human exploration
4 View Integration

- View integration
- Resolving conceptual incompatibility
- Entity clustering for ER models
- Commercial dimension: The BEA story
• **Schema diversity** occurs when different users develop their own understanding of the world
  – the same reality is not always modeled in the same way due to different information needs or workflows

• **Common principle** for schema integration
  – identify the parts of the input schemas that represent the same reality
  – unify their representations
4.2 Basic Steps

• There are four basic steps needed for conceptual schema integration
  1. pre-integration analysis
  2. comparison of schemas
  3. conformation of schemas
  4. merging and restructuring of schemas

• The integration process needs continual refinement and reevaluation
4.2 Schematic View

- Schemas with conflicts
  - Identify conflicts
  - List of conflicts
    - Resolve conflicts
      - Modified schemas
        - Integrate schemas
          - Integrated schema

- Pre-integration analysis
  - Comparison of schemas
    - Conformation of schemas
      - Merging and restructuring
4.2 Pre-Integration Analysis

- **Pre-integration analysis** takes a close look on the individual conceptual schemas to decide for an *adequate integration strategy*
  
  - the larger the number of constructs, the more important is modularization
  
  - is it really sensible/possible to integrate all schemas?
4.2 Pre-Integration Analysis

• First, an **integration strategy** has to be chosen
• Schema integration can either be performed **many at a time**, …
  – requires only one consistent merge
  – conflict analysis from many schemas is difficult
  – may conserve efforts if done well
… or can be performed **two at a time**

- results are step by step accumulated into a single schema
- how to choose the **right order** of schema comparisons
  - the order can influence the final result
- selecting the first schema
  - mixed strategy: skeleton schema
  - otherwise: most important schema
4.2 Comparison of Schemas

• The **resolution of conflicts** needs a thorough comparison of schemas
  
  – **general question**: how do entities **correspond**?
  
  – **naming conflicts** can be detected, e.g. by scanning the data dictionary

  – **structural conflicts** regarding semantics have to be resolved
    
    • different cardinalities in relationships
    • key conflicts
    • …
4.2 Comparison of Schemas

• The individual *perspective of the world* and the *level of abstraction* are major reasons for conceptual incompatibilities

– Example

• *A product is a unit of sale* for the marketing department, but *consists of parts* in the view of the engineering department.
4.2 Comparison of Schemas

• The **level of abstraction** directly influences the schema design

• **Simple example:** A customer buys a product

• The **marketing view** focuses on how many people buy some product, e.g., for advertising
  – only the characteristics of the customer and product and the connection are needed
4.2 Comparison of Schemas

- The **accounting view** also needs the exact order number for identification of individual customer transactions
  
  – focus is on the purchase, but individual orders have to be distinguished
• The **sales view** needs all individual order details, e.g., for troubleshooting or CRM
  
  – focus is on orders
  
  (which provide the basis for purchase contracts)
4.2 Comparison of Schemas

- **Marketing**

- **Accounting**

- **Sales**
• Different user groups use different names to refer to the same entities (differing terminology)
  – **synonyms**: two terms for the same entity
  – **homonym**: the same term for different entities

• **Rule of thumb:**
  eliminate synonyms, rename homonyms!
The main goal is to make schemas **compatible** for integration.

Conformation usually needs **manual** interaction:
- conflicts need to be resolved semantically
- rename entities/attributes
- convert differing types, e.g. convert an entity to an attribute or a relationship
- align cardinalities/functionalities
- align different data types
• Besides renaming and type conversions, **abstraction** can be useful
  – generalization and aggregation allows to create new supertypes or subtypes

• Also **assertions and constraints** must be generalized or distributed among the type hierarchy
  – for example, checking accounts and saving accounts are both types of accounts, but may differ with respect to the minimum balance constraint
4.2 Conformation of Schemas

• **Example:** Resolving differing terminology

• **Homonyms:** Schema 1 and 2 both contain the term *jaguar*, but mean different entities
  – Rename to *jaguar_car* and *jaguar_animal*

• **Synonyms:** Schema 1 contains the term *jaguar*, whereas schema 2 contains the term *panther*
  – global schema should model *panther is_a jaguar*
  – constraint on the black color should be added
• How to merge schemas into a global schema?
  – copy all distinct entities from the individual schemas
  – apply renaming, overlapping entity integration, abstraction, attribute type conversions, etc.
  – put in the distinct relationships from all schemas
  – again use renaming, cardinality/functionality conversions, etc.
  – restructure the resulting global schema
4.2 Merging and Restructuring

• The final *restructuring of the schema* is driven by the goal of completeness, minimality, and ease of understanding

  – *completeness* mandates that all concepts in the global schema appear *semantically intact*
    • all different concepts of every individual schema are also part of the global schema
    • for each concept, there are no missing attributes, no constraints that cannot be met by all members of a type, etc.
– **minimality** enforces to remove all redundant concepts from the global schema

- e.g. overlapping entities or redundant relationships
- often, the question of minimality can only be decided semantically
– ease of understanding means that the global schema makes sense to the users

• in particular, abstraction and fine granular levels for entities can be very confusing

• Example: Subtype entities have to be clearly distinguishable, and should have only attributes that are not inherited from the supertype.
• **Doomsday Legion (DDL)**
  
  – cooperation of villains from all over the world striving for global domination
  
  – channeling resources, staff, experience and power for reaching their goals
  
  – centralized and coordinated management of all shared assets

  • lairs
  • minions
  • assault squads
  • …
4.2 Example: View Integration

• Doomsday Legion schema
• Justice League (JL)
  – federation of super-powered heroes fighting against global crime and villainousness
    • in particular: opposing the Doomsday Legion
  – central management of joint operations and resources
• Justice League schema
And then, strange and evil aliens invade earth without any obvious reason.

– Justice League wants to save earth (that’s what they do)
– Doomsday Legion wants to save earth (without people, global domination is no fun)
– great idea: Join Forces
  • Defenders of the Earth
– great problem: joining large organizations is not that easy
  • beside the problem of ignoring old hatred, the data schemas need to be integrated for central mission control and planning
4.2 Example: View Integration

• How to integrate?
Integrating the person models

- different structure
  - DDL more general → Merge JL into DDL
    - generalize *Hero* and *Sidekick* into *Person*
  - but: **Attribute Homonym**
    - DDL uses the real name of *name*, the villain identity is *alias*
    - JL puts real name into *secret identity* and hero name into *name*
    - name *Victor von Doom* and alias *Dr. Doom* vs.
      name *Invisible Woman* and secret identity *Susan Storm*
    - attributes need to be **renamed** and **transformed** correctly
Integrating the person models

- **semantic consolidation**
  - *Hero* and *Villains* should be treated the same
    - both are highly skilled and powerful super members of DotE
    - merge classes into *SuperOperator* class
  - *Sidekicks* and *Henchmen* are close Assistants of an operator
    - *Heroes* usually only have **one** *Sidekick*
    - Use more general 1:N association to also capture *Henchmen*
• Integrating the person models

– contains Heroes and Villains, as well as their respective Sidekicks or Henchmen

– Heroes and Sidekicks get an additional id
4.2 Example: View Integration

• Integrating bases
  – *Villains* only have 3 types of bases, explicitly modeled
  – *Heroes* may have any kind of base, given by the type attribute
  – Two solutions
    • merge DDL into JL
      – *geoOrbit*, *city*, and *coordinates* become *location*
      – *type* is given by subclass
      – only possible in a lossless fashion because subtypes don’t have additional attributes
    • merge JL into DDL
      – depending on type, a base is assigned to one of the subclasses
      – 4th subclass necessary for all other types (could also be merged into superclass)
• Integrating bases

– only *Villains* owned lairs; no information of *ownership* for former *Hero* bases
4.2 Example: View Integration

• Integrating skills and powers
  – JL only stores Super Powers
  – DDL stores all Skills (including super powers)
    • more general
  – Merge JL into DDL
    • all old justice league Super Powers become Skills of the type super power
    • name is either null or manually completed
    • No information on Skills of Sidekicks
4.2 Example: View Integration

• Integrated schema
4.2 Outlook

• View integration is a **semantic process**
  – this usually means a lot of **manual work**
  – computers can support the process by **matching** some (parts of) schemas

• There have been some approaches towards **(semi-)**automatic matching of schemas
  – matching is a complex process and usually only focuses on simple constructs like *Are two entities semantically equivalent?*
  – the result is still rather error-prone…
4.2 Outlook

- **Basic methods** (that can of course be mixed freely)
  - **label-based matching**
    - for each label in one schema, consider all labels of the other schema and every time gauge their semantic similarity
  - **instance-based matching**
    - looking at the instances (of entities or relationships) one can e.g. find correlations between attributes
      - Are there duplicate tuples? or
      - Are the data distributions in their respective domains similar?
  - **structure-based matching**
    - abstracting from the actual labels, only the structure of the schema is evaluated, e.g. regarding element types, depths in hierarchies, number and type of relationships
• Sometimes schema integration is **query-driven**
  – the integration is only needed in order to query several different information sources having different schemas

• In that case only a **schema mapping** is needed
  – basically the mapping is a list of correspondences between equivalent entities or relationships of heterogeneous schemas
  – the query can then be **translated** for each different schema using the mapping
  – the mapping can be derived manually or automatically from a respective matching
• View integration
• Resolving conceptual incompatibility
• **Entity clustering for ER models**
• Commercial dimension: ...

The BEA story
4.3 Entity Clustering

- Sample schema integration result

National Contaminants Information System (NCIS)

- © Fisheries and Oceans, Canada
When multiple schemas are merged, global schema can become very large

- many different entities and relationships between them
  - e.g. global view of a company with all its dependencies
- but some parts from different views are entirely independent
  - e.g. accounting does not need technical specifications of products

**Idea:** Cluster semantically coherent parts and abstract from their actual entities in the global schema
4.3 Entity Clustering

- Abstracting complex units allow showing the entire model on a **single sheet of paper**
  - easy to get an overview and easier to integrate units separately
  - **zoom in** for more details

![Entity Clustering Diagram]

- **Shop** sells **Product**
- **Branch** sells **Item** in **Product Line**
- **City** of **Chain**
- **Shop**
• **Grouping** is an operation that combines entities and their relationships to form higher-level constructs
  
  – groups are called *entity clusters*
  
  – can also be performed *hierarchically* from the entire database (root entity cluster) over several levels down to the individual entities
  
  • all original entities are on clustering level 1
4.3 Clustering Concepts

• Usually, **entity clusters** are depicted similar to normal entities in ER diagrams
  – by a **dark-bordered box**
  – numbered according to the **clustering level** and an identifier
  – interfaces for **inter-cluster relationships** have to be annotated
• **Grouping operations** are the fundamental components of entity clustering
  – all operations are **heuristic** in nature

• **Often occurring** operations are
  – dominance grouping
  – abstraction grouping
  – constraint grouping
  – relationship grouping

• They can be applied **recursively** or in a variety of combinations to produce higher level clusters
• **Dominance grouping** focuses on semantically dominant entities in the ER diagrams
  
  – hubs for *otherwise unconnected* or *weak* entities
4.3 Grouping Operations

• **Abstraction grouping** clusters entities of a specific super-type
  – especially helpful, if subclasses have no individual relationships
• **Constraint grouping** clusters entities related by the same constraint

  – e.g. integrity constraints such as XOR constraints
4.3 Grouping Operations

• **Relationship grouping** focuses on ternary or higher-degree relationships
  – the relationship is represented as a whole
4.3 Clustering Technique

- Identify all major functional areas and subareas in a top down analysis
  - functional areas are often defined during the requirement analysis as important organizational units (e.g. HR or R&D) or business activities
  - usually there will be a certain degree of overlap, for example employee data will be administrated by HR, but may also be needed in other departments
4.3 Clustering Technique

• The actual clustering has **four steps**
  – define points of grouping within each functional area
    • locate **dominant entities**, consider **abstraction**, find **n-ary** or **constrained relationships**, etc.
    • if such points do not exist, consider grouping the entire area
  – form entity clusters
    • use the **basic grouping operations** on elementary entities and their relationships to form higher level clusters
    • since entities might belong to several clusters, **define priorities** like *always prefer abstraction grouping, avoid crossing boundaries of functional areas, or leave entities ungrouped, if they belong to two or more groups at the same level of precedence*
4.3 Clustering Technique

– form higher level entity clusters
  • apply the grouping operations recursively to any combination of elementary entities and entity clusters
  • stop, if the diagram’s complexity is sufficiently low: This defines the root entity cluster

– validate the cluster diagram
  • check for consistency of the interfaces (relationships) between entities or entity clusters at each level of the diagram
  • verify the meaning of each level with the intended users
4 View Integration

• View integration
• Resolving conceptual incompatibility
• Entity clustering for ER models
• Commercial dimension: The BEA story
4.4 The BEA Story

- What happens, if you don’t integrate properly?
  - think about the Mars disaster…

- What happens, if you integrate?
  - well, your processes are improved and you become more efficient…

- What happens, if you help others to integrate?
  - short version: you found a company, get insanely rich and are finally bought by Oracle for 8.5 billion USD in 2008
• **BEA Systems Inc.**
  – founded in 1995 in San José, CA, USA
  – before Oracle’s takeover, the company had more than 4000 employees and about **one billion** in revenues
  – **product lines**
    • Tuxedo for **distributed transaction processing** (1995)
    • WebLogic provides a **J2EE enterprise infrastructure** (1998)
    • AquaLogic provides a **service-oriented infrastructure** (2005)
  – **acquisitions** of some companies specializing in middleware and business process management
4.4 The BEA Story

• What are they actually doing?
  – case study

• the DekaBank Group is the central asset manager of the Sparkasse Financial Group managing funds of around 90 billion EUR

• the Bank wanted an access layer to central data sources so that all data for the portfolio structure is available for fund management

• in 2006 DekaBank deployed the BEA AquaLogic Data Services Platform, which models the central data uniformly in a technical context and provides these business objects to local applications in real time
two Challenges

• consolidation of various pieces of information from numerous channels
• provide the information in different formats such that local applications can further process the data

finally, after a lot of integration, data is presented to the outside via a standard access layer in real time

duration: about five month

costs: ???
4.4 The BEA Story

• **BEA AquaLogic Data Services**
  – special Feature: **easy-to-use modeling**
    • *In an SOA environment*, a data model must be flexible so that it can represent any *complex entity* and rich enough to provide information about *data structure*, *relationships*, and services to read or update.
    • *Data services are illustrated in model diagrams* and can easily be shared with others in the enterprise for greater data consistency and reuse.
    • *Mappings and transformations can be designed in an easy-to-use GUI tool* using a library of over 200 functions. For complex mappings and transformations, architects and developers can bypass the GUI tool and use an XQuery source code editor to define or edit services.
What tools are actually given to support integration?

- Data Translation Tool
  - transforms binary data into XML
  - transforms XML to binary data

- Data Transformation Tool
  - transforms an XML to another XML

idea
- transform data to application specific XML
  - transform to other application’s XML or general schema
  - transform back to binary

note: the integration work still has to be done manually
4.4 The BEA Story

- **Data Translation**
  - Metadata Description Language describes how *binary data* (files, DB query results, etc.) is mapped to XML
  - BEA provides *editor* and execution *engine* for MFL
  - *customer* creates MFL

  - or alternatively: **Highly paid BEA consultants**
4.4 The BEA Story

• Data Transformation
  – XSL (eXtensible Stylesheet Language, a W3C standard…) is used to transform XML to different XML
  – BEA provides editor and execution engine for XSL
    • both are rather…simple?
  – customer manually creates XSL
    • or alternatively:
      Highly paid BEA consultants
I cannot afford expensive BEA consultants and the AquaLogic Integration Suite, what now?

– do it completely **yourself**
  • most used technologies can be found as open source projects (data mappers, XSL engines, XSL editors, etc.)

– do it **yourself** with **specialized tools**
  • many companies and open source projects are specialized in developing data integration and transformation tools
    – CloverETL
    – Altova MapForce
    – BusinessObjects Data Integrator
    – etc…
4.4 The BEA Story

- **Altova MapForce**
  - same idea than BEA Integrator
    - also based on XSL and a data description language
  - editors for binary/DB to XML mapping
  - editor for XSL transformation
  - automatic generation of data sources, web-services, and transformation modules in Java, C#, C++
• **CloverETL**
  
  – based on own ETL transformation language
  
  – core tools are open source
    
    • server and GUI tools are sold under commercial license
  
  – can read data from any database
  
  – (visually designed) ETL Script converts data into other data
    
    • XML
    
    • DB with different schema
    
    • etc.
4 Next Week

- Basic set theory
- Relational data model
- Transformation from ER
- Integrity Constraints
- From Theory to Practice