TECHNICAL TRANSACTIONS | **CZASOPISMO TECHNICZNE**
MECHANICS | MECHANIKA
3-M/2016

DARIUSZ KARPISZ*

DESIGN OF MANUFACTURING DATABASES

PROJEKTOWANIE PRZEMYSŁOWYCH BAZ DANYCH

A b s t r a c t

The article presents specific cases appearing in the design of manufacturing databases. In particular, a decomposition of entity model to the physical data model is presented. Understanding of significant differences between them makes it possible to build such structures of industrial databases that include special cases which occur in them. Different strategies for the transition from hierarchical entities model to targeted database tables are described. Additionally, an overview of problems associated with unary entities is outlined in this document.

Keywords: manufacturing databases, *ERD*, *PDM*, *super-entities*, *sub-entities*

S t r e s z c z e n i e

W artykule przedstawiono specyficzne przypadki spotykane w projektowaniu przemysłowych baz danych. W szczególności zajęto się dekompozycją modelu encji ERD do fizycznego modelu struktur danych PDM. Zrozumienie istotnych różnic między nimi daje możliwość zbudowania takich struktur przemysłowych baz danych, aby uwzględniały szczególne przypadki w nich występujące. Przedstawione różne strategie przejścia od modelu z hierarchią encji do docelowych tabel bazy danych

Słowa kluczowe: przemysłowe bazy danych, *ERD*, *PDM*, *hierarchia encji*, *encje podrzędne*

DOI: 10.4467/2353737XCT.16.123.5734

* Ph.D. Eng. Dariusz Karpisz, Institute of Applied Informatics, Faculty of Mechanical Engineering, Cracow University of Technology.

1. Introduction

Modeling relational databases is based on a well-defined theory [1, 2]. Despite the existence and availability of a number of graphical tools for database design, creation of the correct model is difficult and time consuming.

In every industry area, there is a variety of custom cases which require specific design mechanisms. Such cases can be found also in databases for manufacturing systems. The Entity Relationship Diagram (ERD) is a basic tool for database design [3], but its logic elements – entities representing a fragment of reality do not necessarily correspond to the target database tables. This way it is possible to result in flexible logical description of the topic database separate from the implementation (usually in the SQL language). Transformation of the ERD diagram into correct data storage structures is realized by means of the Physical Data Model (PDM). Importantly, the number of entities and tables after the transition may vary significantly.

2. Methods

No particular difficulties from the perspective of relational theory are encountered when using classical methods of database design, including normalization, up to the 3rd Normal Form. However, there are special cases in which the details of the types of entities relationships have a significant impact on the implementation and operation of the system.

2.1. Data Flow representation

In the case of designing entities based on data flow responsible for the order of operations, it is possible to use classic model with unary (recursive) entities relationships when both the Participants in the relationship are the same entity. Such an approach is shown in Fig. 1 and 2.

In the case shown in Fig. 1, subsequent process steps follow each other, each operation is always preceded by only one other operation. The figure shows a simplified entity *Workpiece Operation* (state of the workpiece in manufacturing process as an example in [4]) with

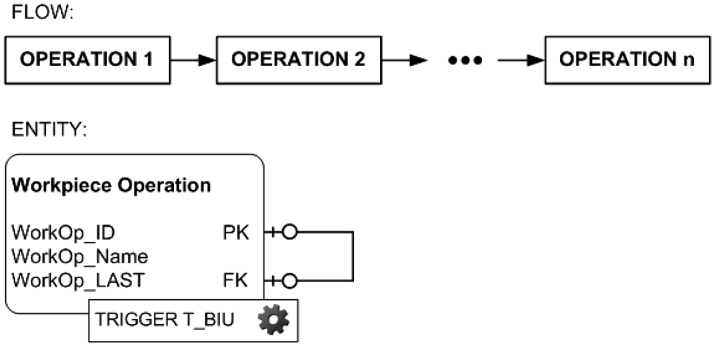

Fig. 1. Example of simple process flow represented by unary relationship entity

WorkOp_ID Primary Key and Foreign Key *WorkOp_LAST*, but the Foreign Key is the same entity (unary relation). Martin Notation is used in the ER model (Crow's Foot notation) to show the participation of entities in the relationship. As default in unary relations must be used in correlations *0:1*, *1:1* or *1:many* (*1:* as perpendicular line, *many:* as Crow's Foot) with both side optionality (indicated by the open circle). The optionality is required in order to be able to insert and modify data in targeted database table.

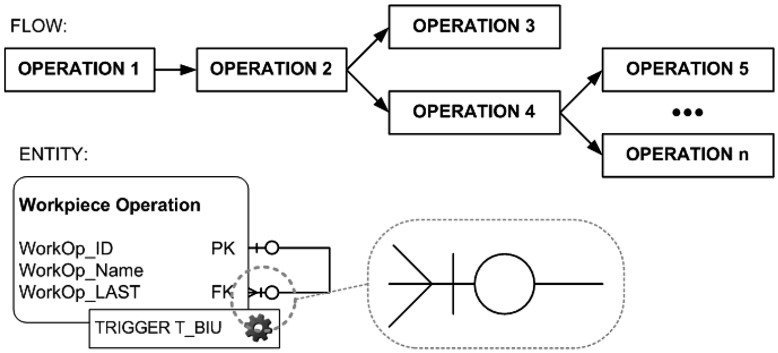

Fig. 2. Example of process flow with branchiny represented by unary relationship entity

In the case shown in Fig. 2, the following operations can be due to branching of the process. Operations can be executed parallely, allowing the case of any number of parallel operations. The ER model from Fig. 2 differs only in the entity relationship *1:many* from the ER Model in Fig. 1. To implement this functionality, database procedures activated by events called triggers [5] are to be used. In the given problem it is recommended to use a trigger type *Before Insert* or *Update* (example in [6]) to check whether there is a previous or next operation defined in the process.

2.2. Supertype and Subtype Entities

A unique design case is a description of reality in the main entity (Supertype Entity) and its subtypes (Subtypes Entities). An example of such an approach is presented in Fig. 3. The *Machine* entity case (e.g. *n*-axis CNC) is described by a set of factors represented by the *Machine Factors* entity.

In the example, the factors were divided into 4 classes:

– *Nominal Factors* − nominal monovalent,
– *Range Factors* − range from/to,
– *Fuzzy Factors* − described as fuzzy values (e.g. set: low, middle, high),
– *Timed Changing Factors* − time-varying values.

The *Machine* entity remains in a constant relationship *1:n* with the *Machine Factors* entity. The additional *Sample* entity designed to store data from the measurement of specific time-dependent factors is in relationship only with the subtype *Timed Changing Factors* entity. This is one of the standard cases in databases used in the Total Productive Maintenance Systems (TPM as example in [7]).

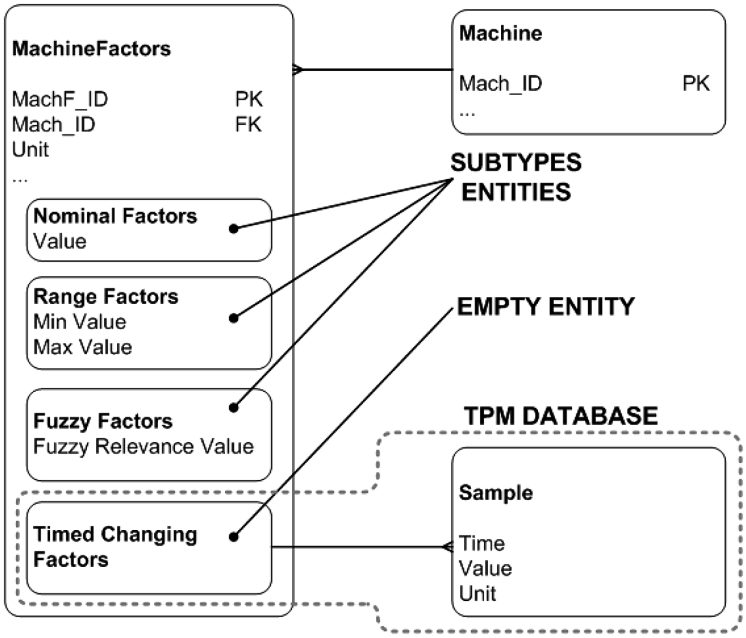

Fig. 3. Example of Supertype and Subtype entities in the Manufacturing Database

The presented example also shows the use case known as the Empty Entity (*Timed Changing Factors*) without any private attributes. Such an entity signals the need to use a different context of the subtype entity, in this case, with time dependencies.

3. Discussion

Presented in Fig. 1 and 2 unary relationships of the entities are often present in design practice. In fact, the shown cases are not difficult to implement, but usually full implementation to the PDM model with dedicated triggers is skipped due to the insufficient knowledge of developers generating data model of the ICT system from application classes created in the Object-Oriented Programming (OOP) style. The use of such practices as generating data model from the Object-Oriented Application with the Object-Relational Mapping is a wrong approach because it does not guarantee data consistency and compliance with the design of correct data layer.

In the case of Subtypes Entities, there are three solutions for transitioning from the ER Model to the PDM model:

1. For all Subtypes Entities, separate database tables are created which contain foreign key from the parent entity.
   Advantages:
   – Database structure is transparent,
   – Ease to define data integrity restrictions of the FK by the SQL code,

– Ability to add tables corresponding to subtypes entities without interference with existing data.

Disadvantages:

– Search data will require many join operations between tables which can significantly affect system performance.

2. For each Subtype Entity a separate table with a set of attributes from the parent entity is created.

   Advantages:

   – Fast searching when data is collected only from a particular table corresponding subtype,
   – No Foreign Keys are necessary to build a table for the subtype.

   Disadvantages:

   – High redundancy in the case of the parent entity with a large list of attributes and the lack of distinction between subtypes entities,
   – Ambiguity in relations with other tables of database that have previously been in relationships with subtypes entities.

3. Creation of a single table with a set of attributes of the parent entity and subtypes entities.

   Advantages:

   – Fast data searching – everything in one place,
   – Lack of Foreign Keys necessary for table creation for subtype entity.

   Disadvantages:

   – The table raw may require a large amount of storage space,
   – Potential issue with a high number of NULL values impacting performance.

There is no guidance as to the preference of methods to be chosen to decompose the Entity Relationship Diagram to the relational database data model (the Physical Data Model). It all depends on the design assumptions adopted in the analysis of the functionality of target system and the amount of information that is needed to be saved, modified and searched in the system.

References

[1] Codd E.F., *A Relational Model of Data for Large Shared Data Bank*, Communications of the ACM, **13** (6), 1970, 377-387.

[2] Elmasri R., Navathe S.B., *Fundamentals of Database Systems (3th Edition)*, Addison Wesley, 2000.

[3] Date C.J., *An Introduction to Database Systems, 8 edition*, Pearson, 2003.

[4] Gawlik J., Kiełbus A., Karpisz D., *Application of an Integrated Database System for Processing Difficult Materials*, Solid State Phenomena, **223**, 2015, 35-45.

[5] Widom J., Ceri S., *Active database systems: triggers and rules for advanced database processing*, Morgan Kaufmann, 1996.

[6] Karpisz D., *Implementation of assertion for relational databases with ORACLE examples*, Technical Transactions, 1-M/2011, 179-186.

[7] Gawlik J., Kiełbus A., *Application of artificial intelligence in supervising the technological equipments and quality of products*, [in:] Sikora T., Giemza M. (eds.), *The Practice of Quality Management of the 21st Century*, Wyd. Naukowe PTTŻ, Kraków 2012, 508-534.