Color coding ER-diagrams, one approach to modeling semi-structured data. Entity-Relationship diagrams used for aggregate oriented databases

Anders Lassen[1,2], Carsten Ejstrup[1]

[1] Datalogisk Institut, University of Copenhagen (DIKU), Copenhagen, DK-2200N, Denmark

2 Institute for People and Technology (IMT), Roskilde University, Roskilde, Denmark

Abstract: Aggregate oriented databases[26] support collections of documents with unstructured data. The unstructured markup means that anything can be stored anywhere. Notations for conceptual modeling of entity relationship diagrams have been extended with notation for aggregation, which satisfy the nesting of structures in aggregate oriented databases [11]. Conceptual modeling of ER-diagrams [5,27] has the benefit that a generic diagram of the whole database is sufficient for moving forward to a design activity, while conceptual modeling of aggregate oriented databases has established a notation for modeling [2]. However the model is specific and not generic. Solutions that prefer a different nesting needs a separate aggregate data model. Our approach to conceptual modeling does not violate the existing relational constraints and provides an extra conceptual layer where document-structure can be modeled by color coding. We will show that modeling decisions can readily be made and materialized in document views within this notation.

In relational database theory, the study of entity sets is based on the understanding of functional dependencies and normalization. It is this theory of the fundamental properties where every attribute is scalar, not composite. To reach first normal form of a relation, all attribute values must contain atomic values only [Date 1990, p 526-527][8]. No composite attributes; no multivalue attributes, found in JSON or XML. By functional decomposition such representations would be split into a number of relations to reach progressive higher normal forms. The outcome of ER diagramming for relational databases is in many cases a database schema of third normal form or BCNF. But for semi-structured databases, not even the first normal form is reached, underlining the necessity for a notation.

Keywords: *ER-diagram conceptual modeling aggregate -oriented databases aggregate-oriented data model· entity-relationship modeling*

Introduction

Conceptual modeling is used as a 'tool of the trade', when analyzing for structure in relational databases. An early account of databases is given by Date in [8]. The most common use of conceptual modeling for relational databases is the entity-relationship diagrams (ER-diagrams) of Chen 1976[5,6], depicting entities and their relationships. In Chen 1976 he represents entities as boxes and relationships as diamonds. It is a little unclear, as the theory revolves around entity sets and relationship sets. Thus, ER-diagrams could equally well be interpreted as entity-set-relationship-set diagrams. The reason for name giving entities in entity-relationship-diagram in singular is that an entity is but one tuple in an entity set. The box and diamond notation can be seen in Figure 1. In chapter 3 of [5], and in the section covering the network model the diamond notation is left out for the data structure diagram while preserved for the ER-diagram. In [15,25] boxes are shown as entity sets and diamonds are shown as relationship sets. The extended entity relationship diagram (EER-diagram) [25,7,11] extends the ER notation with object orientated notations known from UML: Aggregation/Specialization, including function methods; multiple inheritance. The higher-order entity relationship model diagram (HERM) attempt a diagrammatic form that is not bound to the relational model [28]. The circle with a cross is used in [28] in modeling. Relational databases are well described in [15,25,27] introducing relational algebra, decomposition using functional dependencies, normalization, SQL and conceptual modeling, among others.

In modeling, the concept of functional dependencies and normalization guides reasoning about selection of attributes in relations. The conceptual modeling is manifested in ER-diagram that may take any form according to the modeling tool at hand (crowfoot-notation[9], box-diamond-notation [15,25,21] and UML class diagram style notation [10,21]; data-diagram style [1]. Chen [5] works with four levels of abstraction: Modeling entities and their relationship which exists in your mind is level one; Organizing the information structure by normalization or functional decomposition is level two; modeling access-path-independent data-structures, e.g. modeling tables and their constraints, is level three; and modeling access-pathdependent-data-structures is level four.. The data-diagram style is used in [9] for modeling level 3 diagrams with the intended database schema objects. The best guideline for designers is to explain their use of notation in their reporting. The fundamental constructs in an ER-diagram: is identification of entity-sets(entities); and the relationship between entities, drawn as one-one,

one-many or many-many - relationships according to the notation used. These are the fundamental properties. In addition, Entity-sets can further be modeled as an is-a hierarchy, in which case entity-sets of similar type are modeled. Weak entity-sets model close owner ship to a parent entity-set suggesting cascading delete properties between the two.

The conceptual modeling is then used by the designer as a starting point for creating a set of tables and constraints: primary keys, foreign keys; suggested indexes; and domain constraints, in SQL data definition language (DDL). A foreign key draws an attribute on the many entity set for a one-many relationship; and a new mapping table is created for each many-many relationship. Even cascading policies can be considered. In particular the weak entity-set is interesting when modeling for semi-structured databases [15]. Semi-structured databases are exemplified as XML, JSON, BSON text-file format found in document stores (Mongo, Couchbase, Casandra). The underlying structure of a semi-structured database is a undirected graph, but constraints modeled as foreign key relationships can be defined (XMLSchema for XML). In this case the mapping to an ER-diagram is readily identified. To outline the problem, the use a of diagramming to reasoning about databases is well established and the notations for ER-, EER- and HERM-diagrams so rich that most modeling features are included.

Consider modeling for a semi-structured database. In this case lists and composite data structures must be represented as well. This model is now not adequate to describe the datamodel in terms of overall normalization. Maybee locally. In this case a level of detail approach could preserve the high level ER-diagram in observing rules of normalization and at a lower level, on a subdiagram, model for semi-structured representations. And the XML structures not readily extracted from the ER diagram, can readily be extracted from sub diagrams.

It is counter intuitive to use a conceptual modeling tool to reach a certain database model and then break it down by modeling for semi-structured documents. No structure is upheld in a semi-structured database and yet, as a document database grows, some structure is advised. An ER-diagram upholds the theory of the relational data, and thus is a modeling tool for the chosen relationships between data in the domain. In itself, an ER-diagram is a diagram and not a database.

Color coding is a method used in computer science as a tool for analyzing written text. It is well known for programmers to color code source code printout, when analyzing and reasoning about program structure when debugging the authors own code or debugging existing code [personal work experience]. Color coding is used by design team analyzing the existing work practices for a future system implementation. Interviews and observations recorded on tape can be transcripted and the transcription analyzed with a thematic analysis. In this process color coding represents different themes [4,16,19,18].

In this paper we aim to present one color coding scheme as a prototype for a more general method of using conceptual relational modeling as the data analysis tool for both relational databases and semi-structured databases. This is accomplished by mapping decisions for semi-structured layout with color code.

Supporting Literature for Conceptual Modeling Languages

In Ng (2010) [23] a formal definition of the entity relationship model is given. A note to this work is that attention is given to value set. The relational theory of normalization says that to reach first normal form every attribute must be single valued. But semi-structured databases do not obey relational theory and multivalued value sets (lists) and composite fields (mappings into cartesian products of value sets) are highly relevant to preserve when modeling for semi-structured databases.

[23] states that a relationship relation is normal if only attribute values of involved entities are involved in the primary key of the relationship relation. This means that the relationship relation has local scope to the involved entity sets. A relationship relation is weak if any part of the primary key of the relationship relation is identified by another relationship. This is phrased like: ... Since a relationship is identified by the involved entities, the primary key of a relationship can be represented by the primary keys of the involved entities. We also have two forms of relationship relations. If all the entities in the relationship are identified by their own attribute's values, we shall call it a regular relationship relation. If some entities in the relationship are identified by other relationships, we shall call it a weak relationship relation. ... [23, p224] This modeling distinction limits the modeling of relationships to weak entity sets by not automatically interfering that the relationship to a weak entity must be a weak relationship. An entity set can likewise be defined as regular if the primary key is identified by the entity's own attributes alone or as weak if the primary key must include a relationship. ...If relationships are used for identifying the entities, we shall call it a weak entity relation. If relationships are not used for identifying the entities, we shall call it a regular entity relation. ..[23, p224] This is the basic distinction.

In [22] the extended entity-relationship model is used along with the symbols for aggregation and generalization. The diagram is used to formulate business rules for a generation tool. In [24] the extended entity-relationship operators, generalization and aggregation, are include. In addition the notation for ER- modeling used in [27] is used here: boxes for entity-set and diamond shapes for relationship-sets.

Weak entity-sets are marked with double-line-notation. From [24] definitions of generalization and aggregation are citet: ...Generalization of non-overlapping entity-types. If each instance of a (specific) entity-type S is also an instance of a (generic) entity-type G, then a generalization hierarchy exists between S and G (relationship S is-a-G): If S1 and S2 are specific entity-types of G, an instance of G is at most an instance of S1 or S2 but not both.... [Torey 1986] in [24], p371. ...Aggregation of entity-types. It allows a relationship to be considered as an aggregation that groups its participating entity-types as an entity-type.... [Torey 1986] in [24], p372.

The entity relationship diagram was originally outlined by Peter Chen [6]. In this paper the entity-relationship model is compared to the relational model and the network model. A pointer driven, linked list model described by Backmann and the Codasyl report. Chen goes through the definition of physical entities, entities, entity sets. Then through relationship, relationship sets and the role and the relationship paired. A relationship is a list of collaborating entities, each with its role. Then through the description of value and value sets. Now this may be inherit in the existing models (relational model and maybe even the network model) but is is also part of the description of the entity-relationship model from the start. The next point is that Chen is not concerned with the diagram notation of squares and diamonds seen in Ramakrisnan [25] and Garcia- Molina [15]. In Chen 1978 entity sets as circles and relationship sets are arrows. Apart from this a table-notation is included. The first two diagrams are column- wise with ex. Entity sets in first column and value set in the last column. It is also considered if no primary key is found an artificial primary key can be used to establish one-to-one relationship between a relationship key and the entity values.

Conceptual Modeling

One of the fundamental tools for the computer scientist is the ER diagram for modeling relationships between needed entity-sets in a conceptual model. It is important to preserve proven conceptual modeling tools that serve as excellent communication to the product owner and co-workers when analyzing the problem space and usage space [20,3]. Conceptual modeling of ER-diagrams is one such tool, that can provide insight into chosen relationships in the data. The suggestion here is to work with a color scheme to illustrate the scope of semi-structured databases on a local- or full ER-diagram. A full conceptual ER-diagram captures many relational constraints in the domain and all the entity-sets. Decision choices that must be made according to [25] when modeling include fundamental questions. These questions are equally relevant then modeling for semistructured databases:

- Should a concept be modeled as an attribute?
- Should a concept be modeled as an entity or a relationship?
- What are the relationship sets and their participating entity sets? Should we use binary or ternary relationships?
- Should we use aggregation?

Entity-sets are also just called entities and will be used interchanged in the following. Likewise the overused word relation must be defined. It is in-part due to Ullman 1983 [29] describing a universal query language. One of the properties being considering results of queries relations themselves. In this way queries could be formed based on other queries. Relational constraints are database schema objects and define constraints on tables: primary key constraints, foreign key constraints, cascading policies, unique key constraints, null constrains and check constraints. These relational constraints can to some degree be captured in an ER-diagram. The foreign key in particular is modeled. The foreign key relational constraint is the relation part of the ER-diagram (entity-relation). This causes some confusion in the computer science curriculum. In some cases relation takes on the meaning of an entity set or implicit, an entity-set as a result of a query; and in some cases, as in the ER-diagram, relation takes on the meaning as a foreign key constraint. A question like 'What is the relation between entities in the domain?' means: Identify entity sets in the usage domain, and model the entities and how they relate, Including the multiplicity as one- one relationship (1:1), one-many relationship (1:m) or many-many relationship (m:m). This the the primary modeling in an ER-diagram and tells the database designer where to add foreign key constraints and when to add an extra entity-set to hold many-many relationships. The ER-diagram should be termed 'the entity-relationship-diagram' more clearly.

1. Use conceptual modeling of databases to a grip on the data model. what are the entities and what are the relationships between the entities? Entry level ER-modelling.
2. Refine the ER-model throughout the process.
3. Extract sub ER diagrams for user stories, use cases or other design documents.
4. The ER-diagram does not readily project on XML or JSON structures, since XML or JSON are semi-structured databases and attributes may be carried through in part. A graph notation is well suited to illustrate existing relationships, hereby not projecting selected attributes used in the semi-structured database.
5. Present the semi-structured decision in pseudo-code with graph notation.

6. Using color coding project design choices onto the original ER-diagram. avoid cluttering, using duplicate diagrams or partly diagrams to clearly illustrate design choices.

ER-modeling can be directly followed by color coding thus omitting steps two to five, but it must be stressed that ER-modeling is a conceptual modelling step for database persistence, where the color-coding step is anchoring design choices into semi-structured database fragments.

The unique identifier approach popular in relational databases kind defeats the purpose of an unstructured semi-structured database, however in database design conceptual modeling offers insight into possible organization of data. ER- notation and graph-notation are two dimensions of structured approaches to gaining insight. the unique identifier is kind of a primary key approach ensuring access to key information. the unique key approach is a placeholder for composite keys. in normalization both are superkeys, all attributes are prime and possibly the unique identifier the minimal key.

The conclusion here being that proper semi-structured modeling should include both approaches, an ER-data model for a formal relational analysis and graph model to model semi-structured decisions. I fear the UML class diagram has been left of. In [28] the UML association is limited to two-way relationship, as the origin of association is between classes. this is a modeling deficiency in UML, that could be remedied by introducing a multi-way association for UML modeling of entity relationship diagrams.

Semi-structured datamodel

There are a number of properties of semi-structured databases that are relevant to conceptual modeling. First, an overview of semi-structured databases; and second, justification for interpretation in an ER-diagram.

Semi-structured databases have grown popular over the recent years. So much, that relational databases are termed 'old-school' and 'legacy databases'. But importantly enough legacy databases, for example institutional databases persist 10-20-30 years and are maintained because they are in use. It is the job of IT support and in turn computer scientists to provide configuration management and in due time recurrently challenge the persistence model. It will be interesting to observe the role the semi-structured datamodel will play in light of the existing relational datamodel. JSON and XML are examples of semi-structured databases. Especially with XML clearly defines a semi-structured database. The core contrast is the key-value pair and that the structure is human readable. The database is the structure.

One important example of use of XML is transfer of data in shared-nothing distributed databases. Batch programs and FTP have been the cornerstone in distributed processing. Nightly jobs would serve long running jobs, thus avoiding reducing daytime load. These could be viewed as early request/response systems based on files. Today, relational databases can set up tuple-based support at millisecond response to cross-database-tuple-transfer. A schema of tables can be set up to automatically propagated rows(tuples) to destination tables on a foreign system. These systems are known as messaging systems (JMS, Oracle Advanced queueing) and for systems based on Oracle the stored procedure library AQ (Advanced queuing) implements synchronous and asynchronous messaging. A special field contains the data, the message payload, and it is this information that is formed in XML sometimes wrapped with a soap-tag when the SOAP protocol is in use [authors experience]. Long running batch jobs are replaced in time with messaging request-repose systems that immediately propagate information to the targeted database for processing. The core architecture is to generate a request in JSON or XML, wrapped with protocol information. Add this as a message payload; add the message to a message queue (a special table) and wait for a synchronous or asynchronous response, implemented by a special job.

A second example given is the architectural style of mobile application for Android and IOS termed internet-of-things (IOT). Here the persistence model is based on asynchronous http request/response messages [13] seen in figure 3. Contrast this to driver configuration with synchronous connections. In both cases the underlying protocol is based on TCP, sockets and dedicated port numbers. One of the more popular architectural styles for http-request/response is the REST architectural style [31] described in Roy Fielding's dissertation. The REST architectural style can be implemented in any webserver language like php or python as http-requests that are countered by a response message formed like an XML or JSON document. The http request can include a message payload in the data-area of the PUT command; but the REST architectural style is based on tailoring a request URL and not necessarily relying on server state. So one http request URL gives one server answer in form of a JSON- or XML document. In order for a mobile unit to have database persistence, the database communication is through payloads containing document objects - JSON/XML structures that more formally are thought of as semi-structured database documents.

Experiments

Three experiments are given. Case 1 is a classroom question schema supporting a mobile app. Case two is more complex showing a company schema. Case three is training schema color coded for further processing.

Case 1: The first case is from [12] a bachelor thesis at the university of Copenhagen. The point is building an

app that supports questions to the supervisor in a classroom setting. And the persisting database is Firebase supporting JSON-documents. The ER-diagram for the supervisor-app is presented in figure 6 in crowfoot-notation [9], and the suggested mapping to JSON color coded in brown for the question JSON-structure and green for the room JSON-structure.

In figure 7 the 'denormalized' schema is presented in a UML style with aggregation as the collection symbol. Again, with green color coding for the question-room-category collection, and brown for the question-room collection. It is nonesense to evaluate as an ER-style diagram as the ER-diagram has now been broken down in two structures. The color coding on the ER diagram in Figure 4 is sufficient to convey the intended structure of the respective JSON documents.

Evaluating figure 6 green color coding as a UML class diagram. Room aggregates questions and categories. The diamonds are open. When a room disappears, will the questions and categories remain? Answer: diamonds should be filled. When a room is deleted, so are the questions and categories. The diamonds could be filled to denote a composition relationship.

Evaluating figure 6 brown color coding as a UML class diagram. Room aggregates questions. If a room is deleted the questions may not be deleted. If UML class notation is used the diamond could be filled to denote a composition relationship.

The dangling tuples

The category relationship in the ER diagram (figure 6) should be one-to-many relationship against question, so a question has a category. In the pseudocode JSON structure (figure 7) a room contains a list of categories, and a question, initially belongs to a category. Once the list of categories is changed the relationship to the initial category for affected questions are broken. This is called the dangling tuple problem. If the parent tuple is changed or deleted the child tuple will dangle if no relational constraints are enforced.

Could a dashed line denote possible dangling tuples, and full line dependency preservation? In normalization theory a chase test [15] can be used to confirm dependency preservation. In the JSON structure this must be investigated, but the primary partitioning acts as a master and a secondary structure will be left dangling if a) unique id's are not used; or b) any update or delete (modification) is not propagated down the chain. So, in principle modeling for semi-structured databases without constraints is modeling for dangling tuples and thus primary notation could very well be a dotted line.

Here (in figure 9) the color coding reflects the decisions in the JSON structure. Two weak entities are drawn in green color to model a string array for categories and a substructure for the questions. The fact that a question has a category and that the category could be dangling is modeled with a dotted line. Comment: UML aggregation (open diamond) and composition (filled diamond) is suited to denoted modeling decisions for JSON semi-structured database schemas.

Case 2: color coding company database schema

Case 2 is a company database given in Elsmari [11]. We have selected three structures to model. An employee structure including department information with green color coding. A department structure maintains a list of projects list (the controls releationship set) and referencing a depart manager with brown color coding. A projects structure maintains a list of project members (the works-on relationship set) with a pink color coding. The company database schema is described in [11]. We have added color coding to the ER-diagram (figure 10) and used this to derive a document-oriented data model in with Boaventura notation [17], figure 11. The color coding can be carried through.

Boaventura notation is an established notation that reflects modeling document-oriented [3,17]. Design choices on an ER diagram can be transferred intuitively from ER-diagram to document-oriented Boaventura style notation.

Case 3: color coded personal trainer schema

Case 3 is a personal trainer schema given in [32]. We have selected six documents to model. A fitness center document including a list of coaches with red color coding. A coach document with a list of fitness centers (the works_at releationship set) and a list of trainees (the trained_by relationship set) with blue color coding. A trainee document maintains a list of sessions (the performs relationship set) and a reference to the assigned trainee with a green color coding. A session document maintains a list of exercises (the exists_of relationship set along with the adjusted repetitions, weight, sets) and a reference to the assigned coach and trainee with a yellow color coding. An exercise document maintains a list of moves (the takes_from relationship set, along with adjustments) with a brown color coding. A moves document with a purple color coding. The personal trainer database schema is described in [32]. We have added color coding to the ER-diagram (figure 12) and used this to derive a document-oriented data model in with Boaventura notation [17], figure 13. The color coding can be carried through.

Results

Case 1. The color coding on entity set level with straight line coding is adequate to understand the decomposition offered in figure 5 with Bonaventura notation. When UML-class diagram style have been used they quite appropriately model the semi-structured database schemas intended and breaks down the ER-diagram in subdiagram, one for each structure. The sub diagrams could be enhanced with

black diamonds to denote composition and the fact that when a room is deleted, so is the containing questions and categories.

The analysis for dangling tuples showed changing an initial list of categories left affected questions dangling. Although legal for semi-structured databases, the relational constraint is broken. Decision choices leading to possible dangling tuples could be color coded with dotted lines, and decision choices respecting or enforcing referential integrity constraints could be color coded with lines.

Case 2 and 3. Design choices on an ER diagram can be transferred intuitively from ER-diagram to document-oriented Boaventura style notation (figure 10 and 11; figure 12 and 13).

Discussion

Analyzing document-oriented datamodel is all about selecting attributes. Thus color coding should be simple, e.q. with color dots for attributes and color line markings for marked entity sets, to allow color coding for more than one document-oriented datamodel.

It is further work to represent the aggregation using EER-diagram. Aggregation or rather composition is well suited to describe JSON structures with lists, arrays or nested JSON structure. In either case the ER-diagram as a conceptual modeling tool breaks down into sub-diagrams. We speculate if the ER-notation and normal form modeling is counterproductive to extracting semi-structured schemas? To organize the conceptual model it is helpful. To model the decisions for semi-structured data the datamodel must be denormalized and thus normal form modeling might be counter productive.

Conclusion

Color coding of ER-diagrams preserves high level conceptual modeling, while offering an intermediate step to document-oriented datamodeling. A work process for conceptual modeling towards semi-structured coding has been presented containing 6 steps. A color coding notation has been developed using colors depicting extracted documents as line marking added to ER diagram element where relevant. Use of straight line on relationship sets as a normal notation where referential integrity is meant to be preserved and dotted lines on relationship sets to annotate possible breach of initial referential integrity leaving dangling tuples.

References

[1] Microsoft: Microsoft Access. Relational database management system. Initialrelease november 1998. http://office.microsoft.com/access. Microsoft (1998-2019).

[2] Boaventura Filho, W., Olivera, H.V., Holada, M., Favacho, A.A. Geographic da- ta modeling for NoSQL document-oriented databases. GEO Processing 2015. The seventh international conference on advanced geographic information systems, ap- plications and services. ISBN 978-1-61208-383-4, 63–68 (2015).

[3] Bødker,K.,Kensing,F.,Simonsen,J.:ParticipatoryITd esign.Designingforbusiness and workplace realities. MIT Press (2004).

[4] Charmaz, K.: Constructing grounded theory (2nd ed.). SAGE Publications. (2014)

[5] Chen, P.P.: The entity-relationship model: Toward a unified view of data. ACM Transactions on Database systems, 1, 9–37., (1976)

[6] Chen, P.P.: The entity-relationship model: Toward a unified view of data. Reprint of [5] in full length. In: Embley, D.W. and Thalheim, B. (eds.) Handbook of conceptual modeling. Theory, practice and research challenges. ISBN 978-3-642-15865, pp 57– 84. Springer, Heidelberg (2011).

[7] Chen, P.P.: Entity-relationship model approach to system analysis and design. ISBN 0-444-85487-8. North-Holland (1980).

[8] Date, C.J.: An introduction to database systems. vol 1, 5th edition. Addison-Wesley (1990).

[9] Oracle: Oracle Designer 6i to 10g repository. Modeling tool for full project model with code generation. Oracle Designer 6i 2002. Latest release Oracle Designer 10g 2010. Oracle corporation (2002-2010).

[10] The Dia Developers: DIA diagram editor. Generic modeling tool. The Dia Developers. GNU General Public License, the GPLv2. DIA (1997-2014).

[11] Ramex Elmasri, R., Navathe, S.B.: Database systems. Models, languages, design and application programming. 6th edition. Pearson (2011)

[12] Eriksen, S.R., Jensen, E.S.A., Nielsen, M.J.: Q-up: Development of a mobile ap- plication for use in university exercise sessions, using a cross platform mobile development framework. Lassen, A. (supervisor). Bachelor project report. University of Copenhagen. Department of Computer Science (2017).

[13] Fielding, R.T., Gettys, J., Mogul, J.C., Nielsen, H.F., Masinter, L., Leach, P.J., Berners-Lee, T.: Hypertext Transfer Protocol – HTTP/1.1. In: Network Working Group Request for Comments: 2616. 2068; Category: Standards Track. The Internet Society. The Internet Society, 1–176 (1999).

[14] Garcia-Molina, H., Ullman, J.D., Widom, J.: Database Systems. The Complete Book, Second Edition. ISBN 978-0-13-135428-9. Person (2008).

[15] Garcia-Molina, H., Ullman, J.D., Widom, J.: Database Systems. The Complete Book, Third Edition. ISBN 978-1-78399-319-2. Pearson (2012).

[16] Glaser, B. G., Strauss, A. L.: The discovery of grounded theory: Strategies for qualitative research. New York, NY: Aldine de Gruyter. (1967)

[17] Harmouda, S., Zainol, Z.: Document-oriented data schema for relational database migration to NoSQL. In: Proceedings of 2017 International Conference on Big Data Innovations and Applications (Innovate-Data). IEEE Computer Society, 43–50 (2017).

[18] Holtzblatt, K.: Contextual Design, Chapter 43. In: Ed. Julie Jacko. The human- computer interaction handbook. Fundamentals, evolving technologies, and emerging applications. Third edition. CRC Press. Taylor and Francis Group. (2012)

[19] Holtzblatt, K., Beyer, H.: Contextual design, second edition: Design for life (second ed.). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. (2016)

[20] Mathiassen, L., Munk-Madsen, A., Nielsen, PA., Stage, J.: Objekt orienteret analyse og design. Third edition. UML. Forlaget Marko (2001).

[21] Lucid Software: Lucidchart. Online diagramming software. Initial release December 2008. www.lucidchart.com. Lucid Software (2008-2019).

[22] Nagui-Rass, N.: A formal software specification tool using the entity-relationship model. In: Locopolous, P.(eds.). Proceedings of the 13th international conference on the entity-relationship approach. Lecture notes of computer science 881. ISBN 3-540-58786-1.

[23] Ng, P.A., Jean, P.F.: A formal description of entity-relationship models. In: Chen, P.F. (eds.). Entity-relationship approach (ERA) to system analysis and design. North-Holland, 211–230 (2010).

[24] Petit, J.P., Kouloumdjian, J., Boulicaut, J.F., Toumani, F.: Using queries to improve database reverse engineering. In: Locopolous, P.(eds.). Proceedings of the 13th international conference on the entity-relationship approach. Lecture notes of computer science 881. ISBN 3-540-58786-1. Springer-Verlag (1991).

[25] Ramakrishnan, R., Gehrke, J.: Database management systems, 3rd edition. McGraw-Hill (2003).

[26] Sadalage, P.J., Fowler, M.: NoSQL distilled. A brief guide of the emerging world of polyglot persistence. Pearson Education, Addison-Wesley (2013).

[27] VazSalles,M.(eds.).DatabaseSystems.SelectedChapters.CompiledfromGarcia- Molina, H., Ullman, J.D., Widom, J.: Database Systems. The Complete Book. Second Edition [14]. University of Copenhagen. 2014. ISBN 978-1-78399-319-2. Pear- son (2014).

[28] Thalheim, B.: Entity-relationship modeling. Foundations of database technology. ISBN 3-540-65470-4. Springer (2000).

[29] Ullman, J.D.: Universal relation interfaces for database systems. In: Information processing 83. Elsevier, 243–252 (1983).

[30] Anonymous authors, Wiki.: Class diagram. For the free wikipedia. eng.wikipeia.org. (2006-2018). Last accessed 16 Mar 2019.

[31] Roy Thomas Fielding. Architectural Styles and the Design of Network-based Software Architecture. PhD dissertation. University of California, Irvine. 2000

[32] Andrea Charlie Cordes, Carsten Ejstrup og Thomas Schaer. Personal trainer logging app. Logging data using key-value store database. Lassen, A. (supervisor). Bachelor project report. University of Copenhagen. Department of Computer Science (2018).

Figures:

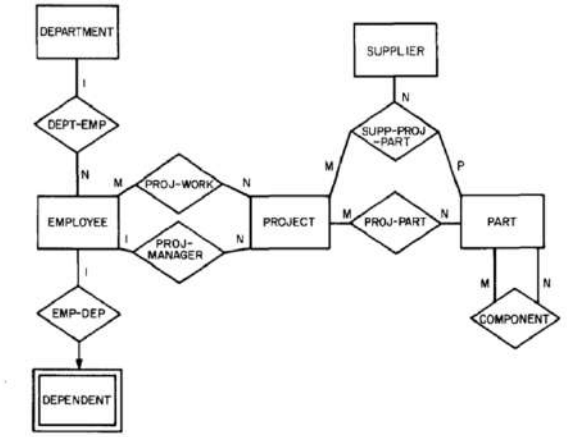

Fig. 11. An entity-relationship diagram for analysis of information in a manufacturing firm

ACM Transactions on Database Systems, Vol. 1, No. 1, March 1976.

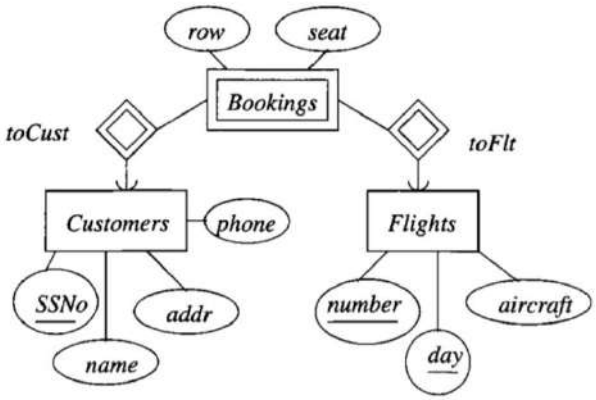

Figure 1. Entity-relationship modeling. Box-diamond-notation from original article [6], figure 12, Box-diamond-notation from Garcia-Molina text book [27], figure 26.

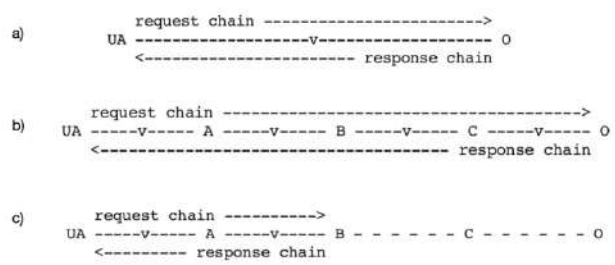

Figure 3. A http-request is sent as a GET or PUT request. The server answers 200 OK and a message body formed as a JSON- or XML document (the http response). The original request/response chain as defined in the http 1.1 standard i 1999 [13]. a. The user agent (UA) post a request to an origin server (O) the origin server eventually responds. A connection (socket on port 80) was used for illustration (v). b. A request/response chain where the message passes through intermediaries A, B and C. c) A request/response chain where the response from intermediary B is a cached response from an earlier request.

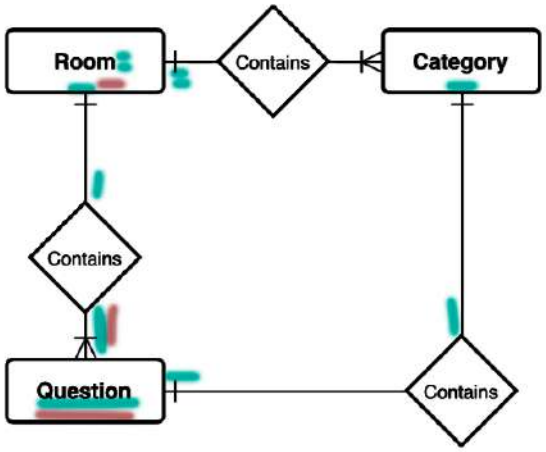

Figure 4. An ER-diagram color coded for two JSON structures brown and green. Rea- ding the ER: A room contains categories and a room contains questions. Figure 24 in [12] color coded.

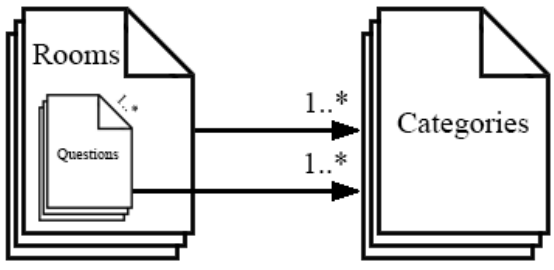

Figure 5. The Boaventura style conceptual model [2] for the two JSON structures brown and green in figure 5. Reading the BV: A room contains categories and a room contains questions. Questions have only one category.

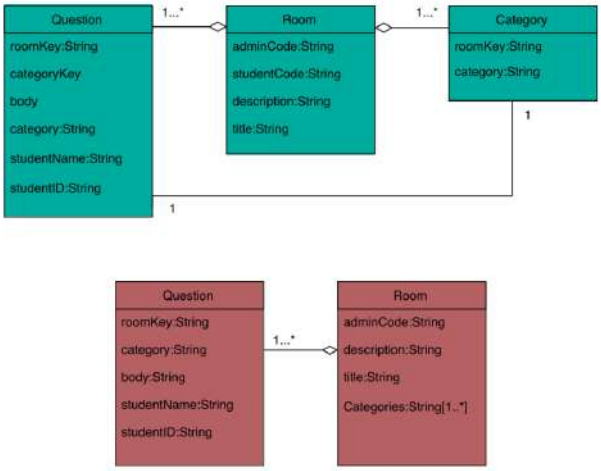

Figure 6. Figure 25 in [12] color coded for the brown and green JSON structure. 'Two denormalized approaches to modeling the database...'. UML style ER-modeling. An aggregate datamodel using an UML class-diagram [26]. The aggregates denotes nested JSON structures

```
Rooms: {
  room_code: {
    admin_code:String,
    title:String,
    description:String,
    Categories:String[1..*],
    Questions: {
      question_id: {
        body:String,
        category:String,
        student_name:String,
        student_id:String
      }
    }
  }
}
```

Figure 7. Figure 3 Listing 2 in [12] Green color coding: Reading the pseudocode JSON structure: A room can have many categories. A room can have many questions.

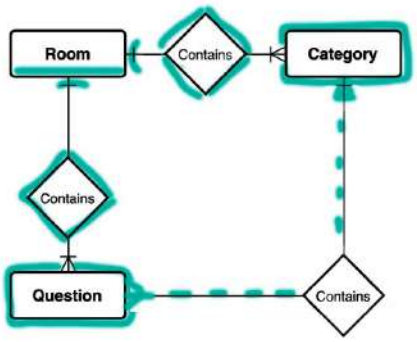

Figure 9. Green color coding on the ER-diagram of figure 24 in [12] .

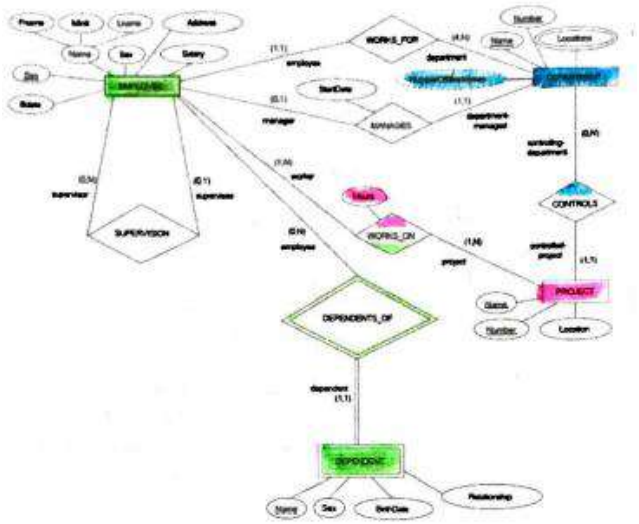

Figure10. The company schema of [11], page 220. The ER diagram has been color coded to reflect the decisions made.

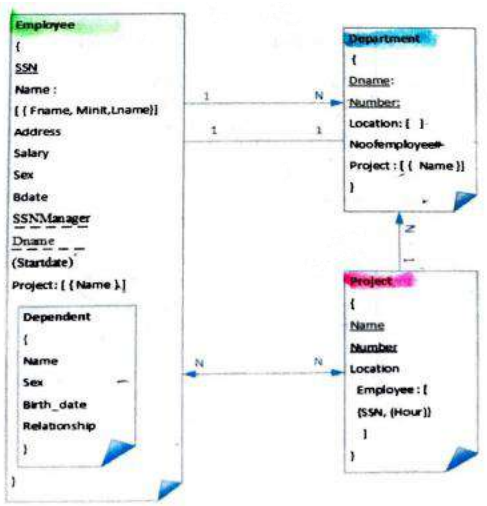

Figure 11. The document-oriented datamodel derived in [17] from the company schema of [11]. The document-oriented datamodel diagram has been color coded to reflect the decisions made in figure 10.

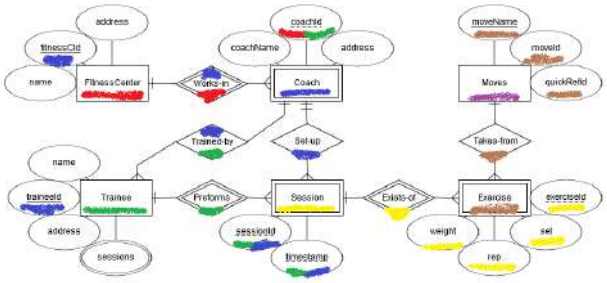

Figure 12. The personal trainer schema of [32].The ER diagram has been color coded to reflect the decisions made.

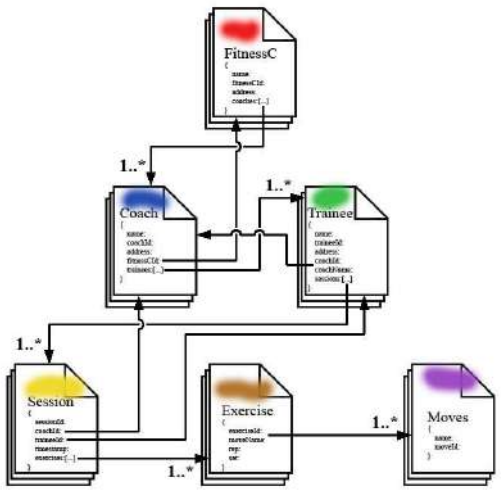

Figure 13. The Boaventura style conceptual model [2] for the ER-diagram in figure 12. The color coding in figure 12 is maintained for each JSON structure.