3.12 FINDING CONCEPTUAL CLASS HIERARCHIES pdf

1: CSDL | IEEE Computer Society

Informally, a conceptual class is an idea, thing, or object. More formally, a conceptual class may be considered in terms of its symbol, intension, and extension - Symbol words or images representing a conceptual class.

Line diagram corresponding to the formal context bodies of water on the left The above line diagram consists of circles, connecting line segments, and labels. Circles represent formal concepts. The lines allow to read off the subconcept-superconcept hierarchy. Each object and attribute name is used as a label exactly once in the diagram, with objects below and attributes above concept circles. This is done in a way that an attribute can be reached from an object via an ascending path if and only if the object has the attribute. In the diagram shown, e. Accordingly, puddle has exactly the characteristics temporary, stagnant and natural. The original formal context can be reconstructed from the labelled diagram, as well as the formal concepts. The extent of a concept consists of those objects from which an ascending path leads to the circle representing the concept. The intent consists of those attributes to which there is an ascending path from that concept circle in the diagram. In this diagram the concept immediately to the left of the label reservoir has the intent stagnant and natural and the extent puddle, maar, lake, pond, tarn, pool, lagoon, and sea. Applying either derivation operator and then the other constitutes two closure operators: The derivation operators define a Galois connection between sets of objects and of attributes. This is why in French a concept lattice is sometimes called a treillis de Galois Galois lattice. With these derivation operators, Wille gave an elegant definition of a formal concept: Equivalently and more intuitively, A,B is a formal concept precisely when: For computing purposes, a formal context may be naturally represented as a 0,1 -matrix K in which the rows correspond to the objects, the columns correspond to the attributes, and each entry ki,j equals to 1 if "object i has attribute j. It is however misleading to consider a formal context as boolean, because the negated incidence "object g does not have attribute m" is not concept forming in the same way as defined above. Concept lattice of a formal context[edit] The concepts Ai, Bi of a context K can be partially ordered by the inclusion of extents, or, equivalently, by the dual inclusion of intents. In this order, every set of formal concepts has a greatest common subconcept , or meet. Its extent consists of those objects that are common to all extents of the set. Dually , every set of formal concepts has a least common superconcept, the intent of which comprises all attributes which all objects of that set of concepts have. These meet and join operations satisfy the axioms defining a lattice , in fact a complete lattice. Conversely, it can be shown that every complete lattice is the concept lattice of some formal context up to isomorphism. Attribute values and negation[edit] Real-world data is often given in the form of an object-attribute table, where the attributes have "values". Formal concept analysis handles such data by transforming them into the basic type of a "one-valued" formal context. The method is called conceptual scaling. It is in general not assumed that negated attributes are available for concept formation. But pairs of attributes which are negations of each other often naturally occur, for example in contexts derived from conceptual scaling. For possible negations of formal concepts see the section concept algebras below. For each finite formal context, the set of all valid implications has a canonical basis, [7] an irredundant set of implications from which all valid implications can be derived by the natural inference Armstrong rules. This is used in Attribute Exploration, a knowledge acquisition method based on implications. As a basic example we mention the arrow relations, which are simple and easy to compute, but very useful. They are defined as follows: Since only non-incident object-attribute pairs can be related, these relations can conveniently be recorded in the table representing a formal context. Many lattice properties can be read off from the arrow relations, including distributivity and several of its generalizations. They also reveal structural information and can be used for determining, e. Extensions of the theory[edit] Triadic concept analysis replaces the binary incidence relation between objects and attributes by a ternary relation between objects, attributes, and conditions. An incidence g,m,c then expresses that the object g has the attribute m under the condition c. Although triadic concepts can be defined in analogy to the formal concepts above, the theory of the trilattices formed by them is much less developed than that of concept lattices, and seems to be difficult. Extensive work has been done on a fuzzy version of formal concept analysis. These two operations are known

as weak negation and weak opposition, respectively. This can be expressed in terms of the derivation operators. Concept algebras generalize power sets. Weak negation on a concept lattice L is a weak complementation, i. Weak composition is a dual weak complementation. A bounded lattice such as a concept algebra, which is equipped with a weak complementation and a dual weak complementation, is called a weakly dicomplemented lattice. Weakly dicomplemented lattices generalize distributive orthocomplemented lattices , i. For a survey, see Kuznetsov and Obiedkov [14] or the book by Ganter and Obiedkov, [8] where also some pseudo-code can be found. Since the number of formal concepts may be exponential in the size of the formal context, the complexity of the algorithms usually is given with respect to the output size. Concept lattices with a few million elements can be handled without problems. Many FCA software applications are available today. Most of these tools are academic open-source applications, such as:

3.12 FINDING CONCEPTUAL CLASS HIERARCHIES pdf

2: Section 4 - Static Modeling

Elaboration - Domain Models - Finding conceptual classes and description classes - Associations - Attributes - Domain model refinement - Finding conceptual class hierarchies- Aggregation and Composition- UML activity diagrams and modeling.

Apdx E - Explicit Constructions E. In retrospect, it would have been good to warn in Sec 1. This is most relevant to highly technical parts such as Sec 7. An additional comment regarding the abstract RAM model pages The current presentation is over-simplified and aimed at the question of computability, while ighnoring the question of complexity. In the context of polynomial-time computation, one typically postulates that all registers and memory cell can hold integers that of logarithmic length i. In such a case, operations like multiplication place the least significant part of the result in one register and its most significant part in another. It would have been better to assert as the main result that the set of Boolean functions and the power set of the integers have the same cardinality. While this simplifies definition suffices in the context of mere computability, it is not good enough for the context of resource bounded computation e. The problem is that the way the definition is formulated allows the machine to invoke the oracle on the answer provided by a previous call at unit time rather than in time that is related to writing the new query. An easy fix is to use two oracle tapes, one on which queries are written, and the other on which the orcale answers are obtained. This is exactly the convention used in the context of space complexity. An alternative solution is to use the convention that the query i. Karp-reductions are often referred to as polynomial-time many-to-one reductions. In fact, such reference can be found also in later chapters of this book. Similarly, the last sentence in Section 2. Minor error in the advanced comment on page Typo on line 11 of page The guideline falls short of addressing all issues. It is not clear how to emulate the execution of the reduction on a given input, since the reduction may be adaptive. The solution is to add a dummy query that is the shortest yes-instance resp. In general, note that constructible complexity bounds on reductions can be enforced regardless of the correctness of the answers. Another alternative proof of Theorem 2. This exercise is meant to prove that the said problem is NP-hard. Proving that it is in NP is beyond the scope of this text; it requires showing that if a solution exists then a relatively short solution exists as well. Btw, there is a typo in the guideline: A propos Exer 2. We mention that this notion of a universal NP-proof system and the fact that the standard NP-proof system for 3SAT is universal underlies the proofs of several "furthermore" results in Section 9 e. Errata regarding Exer 2. The case of 3-Colorability is far from being an adequate exercise; see recent work by R. In item 1 as throughout , the functions are length-increasing, not length-preserving. More importantly, it seems that a more intuitive exposition is possible. Consider all these directed edges and using the length-increasing condition , we get an infinite collection of infinite paths that cover all vertices in both worlds. We seek a perfect matching that is consistent with the directed edges, since such a matching will yield the desired isomorphism provided we can efficiently determine the mate of each vertex. The matching is obtained by detecting the endpoint of the inifinite path on which a vertex resides; once the endpoint is detected, a perfect matching of the vertices on this path is uniquely defined. Comment regarding Exer 2. An additional exercise for Chap 2: Typo on page Def. On page line 3 just after Thm. Minor clarification re the proof of Thm 3. Errata regarding Exer 3. Addendum regarding THM 4. See proof and discussion. Typo in Corollary 4. The machine scans its output tape in one direction. Typo on page 3rd line from bottom: As noted by Guillermo Morales-Luna, the notions of probability ensembles i. Indeed, it might have been a good idea to include a glossary of technical terms such as these as another appendix. Such an appendix may contain also other definitions e. The proof to be presented shows that MA2 is contained in MA1; that is, we can eliminate the error in the completeness condition. A better text may read: Clarification for Exercise 6. In all items, we assume that the reductions only make queries of length that is polynomially related to the length of their input. This guarantees that error probabilities that are negligible in terms of the length of the queries are also negligible in terms of the length of the input. This comment is crucial only for search problems, where error reduction is not necessarily possible. In Item 1 of Prop 7. NP should be PC. See comment regarding Sec 1. An alternative and less direct

3.12 FINDING CONCEPTUAL CLASS HIERARCHIES pdf

solution is showing that the existence of non-uniformly hard one-way functions implies that PC contains search problems that have no polynomial-size circuits a. Note that the a. Better formulation for the 6th line following Def 8. Page lines sketch of proof of Thm 8. The two inputs of the extractor are presented here in reverse order. See comment below regarding Apdx D. Typo on page line 27 just before the itemized list: Typo on page line 4: Typo on page , 3rd line of Sec 8. Typo on line 22 of page Typo on page just after Eq 9. Typo on page header of Def 9. Better formulation for 2nd sentence of Sec 9. Minor error on page line Typo on page line 7: Typo on page last paragraph: Also replace "can be reduce" by "can be reduced". Two typo on page last paragraph: In the 4th line following Thm 9. To obtain a linear size instance either reduced from "restricted 3SAT" in which each variable appears in O 1 clauses and you can show a linear-time reduction from 3SAT to this restricted form or use consistency conditions only on a spanning tree of the set of clauses that share a variable. Typo on page 5th line: The heading "Approximations" should be unnumbered. Copyeditor error on page The definition should be numbered B. This reversal was unconscious. I guess the non-standard order strikes me as more natural, but the difference is not significant enough as to justify deviation from the standard. Typo on page 5th line in the paragraph on amplification:

3.12 FINDING CONCEPTUAL CLASS HIERARCHIES pdf

3: Abstract Classes (C++) | Microsoft Docs

This book "Object Oriented Analysis and Design" is about an introductory idea on pattern design and implement the projects on Object Oriented concepts. Finding conceptual class.

Is a representation of real world conceptual classes, not of software components. Domain Model is a tool of communication. Domain Modeling is driven by use cases as they become known. If a use case appears, the modeling team should look at them to assess whether they contain anything that could have a strong impact on the domain model. The team that builds the domain model should be a small group two to four people that includes developers and domain experts. The smallest viable team would be one developer and one domain expert. A conceptual class may be considered in terms of its symbol, intension and extension. Words or images representing the conceptual class. The definition of a conceptual class. The set of examples to which the conceptual class applies. When creating a domain model it is usually the symbol and intensional view that are of most practical interest. The domain model is build over several iterations in the elaboration phase. It is better to over specify a domain model with lots of fine-grained conceptual classes than under specify it. It is common to miss conceptual classes during the initial identification step and to discover them later during the consideration of attributes or associations or during design work. It is valid to have attributeless conceptual classes or conceptual classes which have a purely behavioral role in the domain instead of an information role. Class identification can be done through the identification of nouns in the use case text or using a conceptual class category list. Some of the verbs in the requirements become methods of the nouns they reference. A report object of other information in a domain model is not useful since all its information is derived from other sources. List the candidate conceptual classes using the conceptual class category list and noun phrase identification. Draw them in the domain model. A Domain Model is not absolutely correct or wrong, but more or less useful. Specification Conceptual Classes are used to avoid duplicated data for every instance of a given class. Instead of having certain attributes that will store always the same values we can put them in a separate class and avoid accidental loss of information that needs to be maintained. Domain model offers a conceptual perspective. It may show conceptual classes, associations, multiplicity and attributes. An optional reading arrow indicates the direction to read the association name it does not indicate direction of visibility or navigation. If not present it is conventional to read the association from left to right or top to bottom, this is not a UML default but a common convention. The reading direction arrow has no meaning in terms of the model; it is only an aid to the reader of the diagram. If a new class is discovered during design modeling then that class should be added to the Domain Model only if it adds communication value to it. Generalization is the activity of identifying commonality among concepts and defining superclass general concept and subclass. Generalization-Specialization class hierarchy is known during Design Modeling as class hierarchy or inheritance. Generalization from the Conceptual perspective is valid to say that a type B is a subtype of a supertype A. Realization is also known as Interface Implementation and indicates that one class implements beheavior specified by another. Realization is deliberately similar to generalization. It indicates that one class implements some behavior specified by another. It is permissible for one implementation class to realize another, this means that the realizing class must conform to the interface but need not use inheritance. In a specification model there is no difference between realization and subtyping. Dependency relationship indicates that one element has knowledge of another element. In class diagrams the dependency relationship is useful to depict non-attribute visibility between classes, in other words, parameter, global, or locally declared visibility. From the specification perspective generalization means that the interface of the subtype must include all elements from the interface of the supertype. Generalization at the implementation perspective is associated with inheritance in programming languages. Generalization is class inheritance, Realization is interface implementation. Subclass "is a" Superclass. A conceptual class partition is a division of a conceptual class into disjoining subclasses. Strong motivations to partition a class into subclass: The subclass has additional attributes of interest. The subclass has additional associations of interest. The subclass concept is operated on, handled, reacted to, or manipulated differently than the superclass or other subclasses in ways

that are of interest. The subclass concept represents an animate thing for example, animal, robote that behaves differently that the super class or other subclasses in ways that are of interest. Motivations to generalize and define a superclass: The potential conceptual subclasses represent variations of similar concept. All subclasses have the same attribute that can be factored out and expressed in the superclass. All subclasses have the same association that can be factored out and related to the superclass. If every member of a class C MUST also be a member of a subclass, then class C is called an abstract conceptual class. When modeling changing states do not model the states of a concept X as sub of X. Define a state hierarchy and associate the states with X, or ignore showing the states of a concept in the domain model and show the states in an state diagram instead. In a Domain Model if a class C can simultaneously have many values for the same kind of attribute A, do not place attribute A in C. Place attribute A in another class that is associated with C. For example a Person can have many phone numbers. Place the phone number in another class. Design model offers a specification or implementation perspective. It may show conceptual classes, associations, operations, multiplicity and navigability. A Design Model illustrates the specifications for software classes and interfaces, typical information includes: Looking in the Domain Model to name design classes reduces the representation gap between the Design Model and Domain Model. Setters and getters should not be included in the Design Model. Programming Language native libraries are not shown in the Design Model. Types of the attributes, method parameters and return values may optionally be shown. During elaboration the Design Model will accompany the use-case realization interaction diagrams, they may be created for the most architecturally significant classes of the design. During construction Design Model will continue to be generated from the source code as an aid in visualizing the static structure of the system. Reference Attributes are shown as an association to another classes in the Design Model. Avoid making or updating any documentation or model unless there is a concrete justification for future use. A Class Diagram describes the types of objects in the system and the various kinds of static relationships that exist among them. There are two principal kinds of static relationships: Class Diagrams show the attributes and operations of a class and the constraints that apply to the way objects are connected. The diagrams are interpreted as describing things in the real world. The diagrams are interpreted as describing software abstractions or components with specifications and interfaces but with no commitment to a particular implementation. The diagrams are interpreted as describing software implementations in a particular technology and language. If you take the conceptual perspective you draw a diagram that represents the concepts in the domain under study. These concepts will naturally relate to the classes that implement them. The conceptual perspective is considered language-independent. If you take the specification perspective we are looking at the interfaces of the software not the implementation. If you take the implementation perspective we are looking at the software implementation. A stereotype is a mechanism to categorize an element in some way. Stereotypes are the core extension mechanism for the UML. An interface is a collection of signatures that specify only the interface. A class contains both interface and implementation. Abstract classes and interfaces are similar but there is a difference. Both allow you to define an interface and defer its implementation, however the abstract class allows you to add implementation of some of the methods. An abstract class is shown with an italicized name. If every member of a class C must also be a member of a subclass, then class C is called abstract conceptual class. Generalization Subclassing and inheritance is shown with a solid line and a fat arrow pointing to the superclass. Either a separate target or shared target arrow style may be used. Realization is illustrated with a dashed line and a fat arrow. Dependency it is illustrated with a dashed arrow line. Another way to illustrate interfaces is by using small circles coming of the classes that implement them, often called lollipops. Exceptions thrown can be listed in another compartment labelled "exceptions". Exceptions caught are modeled as a kind of operation handling a signal. Class scope features are underlined on a class diagram. Class scope operations or attributes apply only to the class rather than an instance.

3.12 FINDING CONCEPTUAL CLASS HIERARCHIES pdf

4: Computational Complexity: A Conceptual Perspective [Goldreich]

conceptual modelling), we could define hierarchies that only satisfy the property of inheritance. The only purpose of this kind of hierarchies is to share and reuse code.

Middle income to low income 8. The intervening twelve years had made things worse. America, he notes, is "more of a caste society than we like to think. And the caste lines have lately become a lot more rigid. In other words, during the first thirty years or so after World War II, the American dream of upward mobility was a real experience for many people. That is, over the past generation upward mobility has fallen drastically. Very few children of the lower class are making their way to even moderate affluence. However, downward mobility was more marked in the US; American workers are more likely to suffer a reduction in their real earnings than workers in Europe. The OECD is pulling its punches. This means that children born to poor families in Britain and the USA are less likely to fulfil their full potential than in other countries and are less likely to break free of their backgrounds than in the past. In other words, we find it harder to earn more money and get better jobs than our parents. Moreover, not only is social mobility in Britain much lower than in other advanced countries, it is actually declining and has fallen markedly over time. The findings were based on studies of two groups of children, one set born in the s and the other in the s. In the UK, while 17 per cent of the former made it from the bottom quarter income group to the top, only 11 per cent of the latter did so. Mobility in the Nordic countries was twice that of the UK. While only the US did worse than the UK in social mobility The puzzle of why, given that there is no evidence of American exceptionalism or higher social mobility, the myth persists has an easy solution. It has utility for the ruling class in maintaining the system. By promoting the myth that people can find the path to the top easy then the institutions of power will not be questioned, just the moral character of the many who do not. Needless to say, income mobility does not tell the whole story. Increases in income do not automatically reflect changes in class, far from it. A better paid worker is still working class and, consequently, still subject to oppression and exploitation during working hours. As such, income mobility, while important, does not address inequalities in power. Similarly, income mobility does not make up for a class system and its resulting authoritarian social relationships and inequalities in terms of liberty, health and social influence. And the facts suggest that the capitalist dogma of "meritocracy" that attempts to justify this system has little basis in reality. Logically, this is not surprising. There is no reason to think that more unequal societies should be more mobile. The greater the inequality, the more economic power those at the top will have and, consequently, the harder it will be those at the bottom to climb upwards. To suggest otherwise is to argue that it is easier to climb a mountain than a hill! Unsurprisingly the facts support the common sense analysis that the higher the inequality of incomes and wealth, the lower the equality of opportunity and, consequently, the lower the social mobility. Finally, we should point out even if income mobility was higher it does not cancel out the fact that a class system is marked by differences in power which accompany the differences in income. In other words, because it is possible in theory for everyone to become a boss this does not make the power and authority that bosses have over their workers or the impact of their wealth on society any more legitimate just because everyone -- in theory -- can become a member of the government does not make government any less authoritarian. Because the membership of the boss class can change does not negate the fact that such a class exists. Ultimately, using usually highly inflated notions of social mobility to defend a class system is unconvincing. After all, in most slave societies slaves could buy their freedom and free people could sell themselves into slavery to pay off debts. If someone tried to defend slavery with the reference to this fact of social mobility they would be dismissed as mad. The evil of slavery is not mitigated by the fact that a few slaves could stop being slaves if they worked hard enough. Loewen reports that "ninety-five percent of the executives and financiers in America around the turn of the century came from upper-class or upper-middle-class backgrounds. Fewer than 3 percent started as poor immigrants or farm children. David Montgomery, Beyond Equality, pg. According to a survey done by C. Meritocracy, after all, does not imply a "classless" society, only that some mobility exists between classes. The fact that the capitalist media are the biggest promoters of the "end-of-class" idea should

make us wonder exactly why they do it. Whose interest is being served by denying the existence of classes? Clearly it is those who run the class system, who gain the most from it, who want everyone to think we are all "equal. Hence they use the media as propaganda organs to mould public opinion and distract the middle and working classes from the crucial issue, i. This is why the mainstream news sources give us nothing but superficial analyses, biased and selective reporting, outright lies, and an endless barrage of yellow journalism, titillation, and "entertainment," rather than talking about the class nature of capitalist society see section D. This is why it is virtually taboo in mainstream academic circles to suggest that anything like a ruling class even exists in the United States. Students are instead indoctrinated with the myth of a "pluralist" and "democratic" society -- a Never-Never Land where all laws and public policies supposedly get determined only by the amount of "public support" they have -- certainly not by any small faction wielding power in disproportion to its size. To deny the existence of class is a powerful tool in the hands of the powerful. As Alexander Berkman points out, "[o]ur social institutions are founded on certain ideas; so long as the latter are generally believed, the institutions built on them are safe. Government remains strong because people think political authority and legal compulsion necessary. Capitalism will continue as long as such an economic system is considered adequate and just. The weakening of the ideas which support the evil and oppressive present day conditions means the ultimate breakdown of government and capitalism. It presents a picture of a system in which only individuals exist, ignoring the differences between one set of people the ruling class and the others the working class in terms of social position, power and interests. This obviously helps those in power maintain it by focusing analysis away from that power and its sources wealth, hierarchy, etc. It also helps maintain the class system by undermining collective struggle. To admit class exists means to admit that working people share common interests due to their common position in the social hierarchy. And common interests can lead to common action to change that position. Isolated consumers, however, are in no position to act for themselves. One individual standing alone is easily defeated, whereas a union of individuals supporting each other is not. Throughout the history of capitalism there have been attempts by the ruling class -- often successful -- to destroy working class organisations. Because in union there is power -- power which can destroy the class system as well as the state and create a new world. Thus the average person is brought to accept current society as "fair" and "just," or at least as "the best available," because no alternatives are ever allowed to be discussed. Given that the existence of classes is often ignored or considered unimportant "boss and worker have common interests" in mainstream culture, its important to continually point out the facts of the situation: To be class conscious means that we are aware of the objective facts and act appropriately to change them. This is why anarchists stress the need for "class consciousness," for recognising that classes exist and that their interests are in conflict. The reason why this is the case is obvious enough. As Alexander Berkman argues, "the interests of capital and labour are not the same. Capitalist industry is the process of continuing to appropriate the products of labour for the benefit of the master class. It is clear that your interests as a worker are different from the interests of your capitalistic masters. The better wages the boss pays you, the less profit he makes out of you. It does not require great philosophy to understand that. Taking the example of the USA, the immediate post-war period the s to the s were marked by social conflict, strikes and so forth. From the s onwards, there was a period of relative social peace because the bosses managed to inflict a series of defeats on the working class. Workers became less militant, the trade unions went into a period of decline and the success of capitalism proclaimed. If the interests of both classes were the same we would expect that all sections of society would have benefited more in the s onwards than between the s to s. This is not the case. While income grew steadily across the board between and s, since then wealth has flooded up to the top while those at the bottom found it harder to make ends meet. A similar process occurred in the s when Alexander Berkman stated the obvious: They have persuaded the workers that they have the same interests as the employers. And they laugh in their sleeves and thank the Lord that you are such an idiot. That only by struggle can you gain respect and an increased slice of the wealth you produce but do not own. And that there is "an irreconcilable antagonism" between the ruling class and working class "which results inevitably from their respective stations in life. For example, James Madison in the Federalist Paper 10 states that "those who hold and those who are without have ever formed distinct interests in society. The difference is that the ruling

class wants to keep the class system going while anarchists seek to end it once and for all. It could therefore be argued that anarchists actually want an "anti-class" consciousness to develop -- that is, for people to recognise that classes exist, to understand why they exist, and act to abolish the root causes for their continued existence "class consciousness," argues Vernon Richards, "but not in the sense of wanting to perpetuate classes, but the consciousness of their existence, an understanding of why they exist, and a determination, informed by knowledge and militancy, to abolish them. In short, anarchists want to eliminate classes, not universalise the class of "wage worker" which would presuppose the continued existence of capitalism. More importantly, class consciousness does not involve "worker worship. In reply, anarchists agree, yes, there are "only" individuals but some of them are bosses, most of them are working class. This is an objective division within society which the ruling class does its best to hide but which comes out during social struggle. And such struggle is part of the process by which more and more oppressed people subjectivity recognise the objective facts. And by more and more people recognising the facts of capitalist reality, more and more people will want to change them. Currently there are working class people who want an anarchist society and there are others who just want to climb up the hierarchy to get to a position where they can impose their will to others. But that does not change the fact that their current position is that they are subjected to the authority of hierarchy and so can come into conflict with it. And by so doing, they must practise self-activity and this struggle can change their minds, what they think, and so they become radicalised. This, the radicalising effects of self-activity and social struggle, is a key factor in why anarchists are involved in it. It is an important means of creating more anarchists and getting more and more people aware of anarchism as a viable alternative to capitalism. And what you do. Hence we see anarchists like Bakunin and Kropotkin, former members of the Russian ruling class, or like Malatesta, born into an Italian middle class family, rejecting their backgrounds and its privileges and becoming supporters of working class self-liberation.

3.12 FINDING CONCEPTUAL CLASS HIERARCHIES pdf

5: B.7 What classes exist within modern society? | Anarchist Writers

One of the most promising concepts in the development of the next generalization of data models is the object-oriented approach. This paper describes a conceptual data model that integrates elements of semantic relationships with object orientation concepts to develop a model for a Bill of Materials (BOM).

After loading a dimension, you can display the default hierarchy. To display the default hierarchy: In the navigation tree, right-click the name of a dimension. Figure shows the Primary hierarchy of the Product dimension. Figure Displaying the Product Primary Hierarchy Description of "Figure Displaying the Product Primary Hierarchy" Creating Cubes Cubes are informational objects that identify measures with the exact same dimensions and thus are candidates for being processed together at all stages: Cubes define the shape of your business measures. They are defined by a set of ordered dimensions. The dimensions form the edges of a cube, and the measures are the cells in the body of the cube. To create a cube: Expand the folder for the analytic workspace. Right-click Cubes, then choose Create Cube. The Create Cube dialog box is displayed. On the General tab, enter a name for the cube and select its dimensions. On the Aggregation tab, click the Rules subtab and select an aggregation method for each dimension. If the cube uses more than one method, then you may need to specify the order in which the dimensions are aggregated to get the desired results. You can ignore the bottom of the tab, unless you want to exclude a hierarchy from the aggregation. If you run the advisors after mapping the cube, Oracle OLAP can determine the best partitioning and storage options. However, if you want to define these options yourself, then complete the Partitioning and Storage tabs before creating the cube. The new cube appears as a subfolder under Cubes. Figure shows the Rules subtab for the Units cube with the list of operators displayed. Each measure belongs to a particular cube, and thus shares particular characteristics with other measures in the cube, such as the same dimensions. The default characteristics of a measure are inherited from the cube. To create a measure: Expand the folder for the cube that has the dimensions of the new measure. Right-click Measures, then choose Create Measure. The Create Measure dialog box is displayed. On the General tab, enter a name for the measure. The new measure appears in the navigation tree as an item in the Measures folder. Figure shows the General tab of the Create Measure dialog box. You can use expressions when mapping cubes in the tabular view. This capability enables you to perform tasks like these as part of data maintenance, without any intermediate staging of the data: Perform calculations on the relational data using any combination of functions and operators available in the OLAP expression syntax. Create measures that are more aggregate than their relational sources. For example, suppose the Time dimension has columns for Day, Month, Quarter, and Year, and the fact table for Sales is related to Time by the Day foreign key column. In a basic mapping, you would store data in the cube at the Day level. However, you could aggregate it to the Month level during the data refresh. Using a technique called one-up mapping, you would map the cube to the Month column for Time, and specify a join between the dimension table and the fact table on the Day columns. Click Help on the Mapping window for more information about these techniques. To map a cube: In the navigation tree, expand the cube folder and click Mappings. The Mapping window contains a schema navigation tree on the left and a mapping table for the cube and its dimensions. This is the tabular view. To enlarge the Mapping window, drag the divider to the left. In the schema tree, expand the tables, views, or synonyms that contain the data for the measures. Drag-and-drop the source columns onto the appropriate cells in the mapping table for the cube. After you have mapped all dimensions and measures, click Apply. Drag the divider back to the right to reduce the size of the Mapping window. After you save the mappings, Analytic Workspace Manager provides the join conditions for base-level mappings such as the ones shown here. It improves the performance of large measures in the following ways: Improves scalability by keeping data structures small. Each partition functions like a smaller measure. Keeps the working set of data smaller both for queries and maintenance, since the relevant data is stored together. Enables parallel aggregation during data maintenance. Each partition can be aggregated by a separate process. Simplifies removal of old data from storage. Old partitions can be dropped, and new partitions can be added. The number of partitions affects the database resources that can be allocated to

loading and aggregating the data in a cube. Partitions can be aggregated simultaneously when sufficient resources have been allocated. The Cube Partitioning Advisor analyzes the source tables and develops a partitioning strategy. You can accept the recommendations of the Cube Partitioning Advisor, or you can make your own decisions about partitioning. Run the Cube Partitioning Advisor after mapping the cube to a data source and before loading the data. You can change the partitioning strategy at any time using the Cube Partitioning Advisor, but you will need to reload the data afterward. You can specify your own partitioning strategy only when creating the cube. Choosing a Dimension for Partitioning If your partitioning strategy is driven primarily by life-cycle management considerations, then you should partition the cube on the Time dimension. Old time periods can then be dropped as a unit, and new time periods added as a new partition. In Figure , for instance, the Quarter level of the Time dimension is used as the partitioning key. The Cube Partitioning Advisor has a Time option, which will recommend a hierarchy and a level in the Time dimension for partitioning. If life-cycle management is not a primary consideration, then run the Cube Partitioning Advisor and choose the Statistics option. The Cube Partitioning Advisor will develop a strategy designed to achieve optimal build and query performance. To run the Cube Partitioning Advisor: Map the cube to its data source, if you have not done so already. On the navigation tree, select the cube to display its property pages. On the Partitioning tab, click Cube Partitioning Advisor. Wait while the Cube Partitioning Advisor analyzes the cube. When it is done, the Cube Partitioning Advisor displays its recommendations. Evaluate the recommendations of the Cube Partitioning Advisor. Select Accept Partition Advice to accept the recommendations. The cube will be re-created with the new partitions. Clear the Accept Partition Advice box to reject the recommendations. Figure shows the Cube Partitioning Advisor dialog box. Each Quarter and its descendants are stored in a separate partition. If there are three years of data in the analytic workspace, then partitioning on Quarter produces 12 bottom partitions, in addition to the default top partition. The top partition contains all remaining levels, that is, those above Quarter such as Year and those in other hierarchies such as Fiscal Year or Year-to-Date. Figure illustrates a Time dimension partitioned by Quarter. However, loading and aggregating the data for your business measures typically takes more time to complete. Unless you are developing a dimensional model using a small sample of data, you may prefer to run the build in one or more background processes. In the navigation tree, right-click the Cubes folder or the name of a particular cube. The Maintenance Wizard opens on the Select Objects page. Select one or more cubes from Available Target Objects and use the shuttle buttons to move them to Selected Target Objects. If the dimensions are already loaded, you can omit them from Selected Target Objects. On the Dimension Data Processing Options page, you can keep the default values. You can also select the number of processes to dedicate to this build. The number of parallel processes is limited by the smallest of these numbers: Click Help for information about these choices. Figure shows the build submitted immediately to the Oracle job queue. To display the data in a cube: In the navigation tree, right-click the cube. Choose View Data from the pop-up menu. The Measure Data Viewer displays the selected measure in a crosstab at the top of the page and a graph at the bottom of the page.

3.12 FINDING CONCEPTUAL CLASS HIERARCHIES pdf

6: Mapping the Object Hierarchy to XML Data | Microsoft Docs

www.enganchecubano.com, I think the point is to demonstrate to newbies the foundations of Object Hierarchies in terms that are familiar to them. If you start with streams, you'll find that many people have no conceptual framework on which to build and thus you'll simply lose them.

Hierarchies have their own special vocabulary. These terms are easiest to understand when a hierarchy is diagrammed see below. In an organizational context, the following terms are often used related to hierarchies: Multiple hierarchies are possible per dimension taxonomy or Classification-system , in which selected levels of the dimension are omitted to flatten the structure Terms about Placement Hierarch , the apex of the hierarchy, consisting of one single orphan object or member in the top level of a dimension. The root of an inverted-tree structure Member , a member or node in any level of a hierarchy in a dimension to which superior and subordinate members are attached Orphan , a member in any level of a dimension without a parent member. Often the apex of a disconnected branch. Orphans can be grafted back into the hierarchy by creating a relationship interaction with a parent in the immediately superior level Leaf , a member in any level of a dimension without subordinates in the hierarchy Neighbour: Auburn In a mathematical context in graph theory , the general terminology used is different. Most hierarchies use a more specific vocabulary pertaining to their subject, but the idea behind them is the same. For example, with data structures , objects are known as nodes , superiors are called parents and subordinates are called children. Degree of branching[edit] Degree of branching refers to the number of direct subordinates or children an object has in graph theory, equivalent to the number of other vertices connected to via outgoing arcs, in a directed graph a node has. Hierarchies can be categorized based on the "maximum degree", the highest degree present in the system as a whole. Categorization in this way yields two broad classes: In a linear hierarchy, the maximum degree is 1. Note that this is referring to the objects and not the levels; every hierarchy has this property with respect to levels, but normally each level can have an infinite number of objects. An example of a linear hierarchy is the hierarchy of life. In a branching hierarchy, one or more objects has a degree of 2 or more and therefore the minimum degree is 2 or higher. The broad category of branching hierarchies can be further subdivided based on the degree. A flat hierarchy is a branching hierarchy in which the maximum degree approaches infinity, i. Therefore, a flat hierarchy is often not viewed as a hierarchy at all. For example, diamonds and graphite are flat hierarchies of numerous carbon atoms which can be further decomposed into subatomic particles. An overlapping hierarchy is a branching hierarchy in which at least one object has two parent objects. History of the term[edit] Possibly the first use of the English word "hierarchy" cited by the Oxford English Dictionary was in , when it was used in reference to the three orders of three angels as depicted by Pseudo-Dionysius the Areopagite 5thâ€”6th centuries. Since hierarchical churches, such as the Roman Catholic see Catholic Church hierarchy and Eastern Orthodox churches, had tables of organization that were "hierarchical" in the modern sense of the word traditionally with God as the pinnacle or head of the hierarchy , the term came to refer to similar organizational methods in secular settings. This is an example of a hierarchy visualized with a triangle diagram. For example, the few Directors of a company could be at the apex , and the base could be thousands of people who have no subordinates. An example of a triangle diagram appears to the right. An organizational chart is the diagram of a hierarchy within an organization , and is depicted in tree form below. Examples include fractal maps, TreeMaps and Radial Trees. Visual hierarchy[edit] In the design field, mainly graphic design, successful layouts and formatting of the content on documents are heavily dependent on the rules of visual hierarchy. Visual hierarchy is also important for proper organization of files on computers. An example of visually representing hierarchy is through the Nested clusters. The Nested clusters represents hierarchical relationships by using layers of information. The child element is within the parent element, such as in a Venn diagram. This structure of representing hierarchy is most effective in representing simple relationships. For example, when directing someone to open a file on a computer desktop, one may first direct them towards the main folder, then the subfolders within the main folder. They will keep opening files within the folders until the designated file is located. For more complicated hierarchies, the stair structure represents hierarchical

relationships through the use of visual stacking. Visually imagine the top of a downward staircase beginning at the left and descending on the right. The child elements are towards the bottom of the stairs and the parent elements are at the top. This structure is effective when representing more complicated hierarchies where steps are not placed in obvious sequences. Further steps are concealed unless all of the steps are revealed in sequence. In the computer desktop example, a file that is being sought after can only be found once another file is opened. The link for the desired file is within another document. All the steps must be completed until the final destination is reached. In plain English, a hierarchy can be thought of as a set in which: The first requirement is also interpreted to mean that a hierarchy can have no circular relationships ; the association between two objects is always transitive. The second requirement asserts that a hierarchy must have a leader or root that is common to all of the objects. Hierarchy mathematics Mathematically, in its most general form, a hierarchy is a partially ordered set or poset. Within this system, each element shares a particular unambiguous property. Objects with the same property value are grouped together, and each of those resulting levels is referred to as a class. Operations such as addition, subtraction, multiplication and division are often performed in a certain sequence or order. Usually, addition and subtraction are performed after multiplication and division has already been applied to a problem. The use of parenthesis is also a representation of hierarchy, for they show which operation is to be done prior to the following ones. In this problem, typically one would multiply 5 by 7 first, based on the rules of mathematical hierarchy. But when the parentheses are placed, one will know to do the operations within the parentheses first before continuing on with the problem. These rules are largely dominant in algebraic problems, ones that include several steps in order to solve. The use of hierarchy in mathematics is beneficial in order to quickly and efficiently solve a problem without having to go through the process of slowly dissecting the problem. Most of these rules are now known as the proper way into solving certain equations. Nested hierarchy[edit] Matryoshka dolls , also known as nesting dolls or Russian dolls. Each doll is encompassed inside another until the smallest one is reached. This is the concept of nesting. When the concept is applied to sets , the resulting ordering is a nested hierarchy. A nested hierarchy or inclusion hierarchy is a hierarchical ordering of nested sets. Each doll is encompassed by another doll, all the way to the outer doll. The outer doll holds all of the inner dolls, the next outer doll holds all the remaining inner dolls, and so on. Matryoshkas represent a nested hierarchy where each level contains only one object, i. The general concept is both demonstrated and mathematically formulated in the following example:

3.12 FINDING CONCEPTUAL CLASS HIERARCHIES pdf

7: Formal concept analysis - Wikipedia

3/11/ Lecture 7 Objectives Explain the purpose of domain modelling. Explain what a conceptual class is. List techniques for finding conceptual classes. Define the terms Aassociations and Attributes. Identify and explain different types of association present in a domain model. Create a domain class diagram for a given domain, using correct UML class diagram notation.

This is depicted using a filled triangle, called a direction indicator, an example of which is shown on the offering of association between the Seminar and Course classes of Figure 5. My advice, however, is if your label is not clear, then you should consider rewording it. The arrowheads on the end of the line indicate the directionality of the association. A line with one arrowhead is uni-directional whereas a line with either zero or two arrowheads is bidirectional. Officially you should include both arrowheads for bi-directional assocations, however, common practice is to drop them as you can see, I prefer to drop them. At each end of the association, the role, the context an object takes within the association, may also be indicated. Inheritance is that mechanism. The UML modeling notation for inheritance is a line with a closed arrowhead pointing from the subclass to the superclass. Many similarities occur between the Student and Professor classes of Figure 2. Not only do they have similar attributes, but they also have similar methods. To take advantage of these similarities, I created a new class called Person and had both Student and Professor inherit from it, as you see in Figure 7. This structure would be called the Person inheritance hierarchy because Person is its root class. The Person class is abstract: Abstract classes are modeled with their names in italics, as opposed to concrete classes, classes from which objects are instantiated, whose names are in normal text. Both classes had a name, e-mail address, and phone number, so these attributes were moved into Person. The Purchase Parking Pass method is also common between the two classes, something we discovered after Figure 2 was drawn, so that was also moved into the parent class. By introducing this inheritance relationship to the model, I reduced the amount of work to be performed. Instead of implementing these responsibilities twice, they are implemented once, in the Person class, and reused by Student and Professor. For example, an airplane is made up of a fuselage, wings, engines, landing gear, flaps, and so on. Figure 8 presents an example using composition, modeling the fact that a building is composed of one or more rooms, and then, in turn, that a room may be composed of several subrooms you can have recursive composition. In UML 2, aggregation would be shown with an open diamond. Another good indication that composition makes sense is when the lifecycle of the part is managed by the whole -- for example a plane manages the activities of an engine. When deciding whether to use composition over association, Craig Larman says it best: If in doubt, leave it out. Unfortunately many modelers will agonize over when to use composition when the reality is little difference exists among association and composition at the coding level. A vocabulary defines the semantics of entity types and their responsibilities, the taxonomical relationships between entity types, and the ontological relationships between entity types. Taxonomies are classifications of entity types into hierarchies, an example of which is presented for persons Figure 9. Ontology goes beyond taxonomy. Where taxonomy addresses classification hierarchies ontology will represent and communicate knowledge about a topic as well as a set of relationships and properties that hold for the entities included within that topic. A taxonomy for people within the university. The semantics of your conceptual model are best captured in a glossary. There are several interesting aspects of Figure 9: It uses UML 2. There are three generalization sets for Person: Nationality, Role, and Gender. These generalization sets overlap - a person can be classified via each of these roles e. This is called multiple classification. Some generalization sets are mutually exclusive from others, not shown in the example, where an entity type may only be in one set. This is referred to as single classification and would be modeled using an XOR exclusive OR constraint between the two or more discriminators.

8: documentation - What's a good example for class inheritance? - Stack Overflow

evolution process [3, 12] to refine a database schema and derive object hierarchies. Existing approaches to adaptive or

3.12 FINDING CONCEPTUAL CLASS HIERARCHIES pdf

evolutionary database design [13, 14] generally provide a set of schema evolution operations such as partitioning a class, merging two.

9: UML 2 Class Diagrams: An Agile Introduction

We can say that, at instantiation time, class hierarchies "get flattened". The phrase object hierarchies, therefore, often refers to whole/part structures. You may have another class, perhaps named Wheel, plus a reference from Car to Wheel so that cars may contain up to four wheels (imagine an array of wheels in a car object, or any other.

3.12 FINDING CONCEPTUAL CLASS HIERARCHIES pdf

Specifications for microfilming manuscripts Kites and flying objects. Problem of Self-love in Saint Augustine Selling a niche practice by John Ventura MapEasys Guidemap to France Proposed fiscal year 2007 budget request for the Department of the Interior Absalom, Absalom Jan Mark Martin heidegger saved my life Advances in carbohydrate chemistry and biochemistry. One hundred bungalows. The mansion on the hill Kneeling, Sitting, Lying The Paragon House Spelling Dictionary British parliamentary election results, 1918-1949 Process and device modeling for integrated circuit design Promises for Mothers Secrets at midnight nalini singh Athol Fugard and Barney Simon Garbage Management in Japan Setting high standards for everyone The growth and structure of eutectics with silicon and germanium 20th Century Magic (Llewellyns High Magick Series) Historical notices of St. Anns parish in Ann Arundel county, Maryland Car Wars Division 5 Set 2 15. Free-trade Diaspora (2003) Abc air band radio Beads of Glass, Beads of Stone Hospital management books Hazrat ali life history in bangla Applied acoustics Training for store service Agriculture and civilization Developing an effective school plan Death of an American Idol Building Divine Intimacy Great ideas of clinical science A Little Light Reading My heart in a suitcase The political economy of oil in Alaska Hereditary bondsmen; or, Is it all in vain?