Generating Hypertext Views on Databases

Falquet, Gilles; Prince, Ian James; Guyot, Jacques

Generating Hypertext Views on Databases

Gilles Falquet, Jacques Guyot, Ian Prince

C.U.I - Université de Genève, 24, Rue Général Dufour
CH-1211 Genève, Suisse
Tél: +41 22 705 77 70 - Fax: +41 22 320 29 27
e-mail: {falquet, guyot, prince}@cui.unige.ch
http://cuiwww.unige.ch/db-research/hyperviews/

Resumé:

Cet article présente un langage et un sytème de construction de vue hypertexte (hypervue) surune base de données. Nous étudions tout d'abord différentes manières de représenter les objets d'un base de données sous forme de composants d'un hypertexte: représentation directe des tuples, représentation d'ensembles de tuples et représentation de tuples associés. Nous montrons ensuite comment ces approches sont intégrées dans un langage déclaratif de définition d'hypervues. Ce langage permet de spécifier comment le contenu des différents types de noeuds ainsi que les liens hypertextes sont formés à partir du contenu de la base de données. Nous donnons également des indications sur la conception de la structure hypertextuelle en fonction des relations sémantiques représentés dans le schéma de la base de données.

Le prototype de traducteur qui a été réalisé génère des hypervues pour le système Worldwide Web (W3) à partir d'une base relationnelle. Nous décrivons les principaux composants du système réalisé et montrons comment le processus de traduction peut rémédier à certains défauts du modèle W3 d'hypertexte.

Mots clés: bases de données, sémantique des données, vues, hypertextes, génération d'hypertextes

Abstract:

This paper presents a language and a system to construct a hypertextual view (a hyperview) of the content of a database. We first study different approaches to map database objects to hypertext components: tuple level mapping, tuple sets mapping, and associated tuples mapping. Then we present a declarative hyperview definition language which integrates these approaches. With this language one can specify for each node type and each link type how to construct it from the database contents. We also show how the semantic relationship represented in the database schema can be used to design the hypertext structure.

A prototype translator has been implemented to generate a Worldwide Web (W3) hyperview of a relational database, the main components of this generation system are presented. We also show how the translation process can overcome some shortcoming of the W3 hypertext model.

Keywords: databases, database semantics, views, hypertexts, hypertext generation.

1. Introduction

It is well acknowledged that accessing a database with traditional query languages, like SQL, is too complex for non specialist or casual users. This is particularly true when the intended answer requires information that is spread across several database tables (or classes). In such a situation the user must be able to select a correct logical access path within the database schema and to express this path as a query language expression. The former task may be particularly difficult when the database schema comprises hundreds of tables and thousands of attributes (often bearing misleading names).

The generally adopted solution consists in providing a set of views or access forms to the user. However forms enable users to perform queries quickly and safely, they limit the user's interaction with the database to a set of predefined actions. This is particularly annoying when the user needs information that exists somewhere in the database but none of the predefined forms has been devised to get it. This approach also raises problems related to information distribution over large heterogeneous networks. Since forms are in fact database client applications, each new user must obtain and run these applications on his local system.

Database browsing or navigation tools can remedy such problems by allowing the user to explore the database contents with a single navigation tool.

Several exploration tools have already been proposed for databases, in particular for object-oriented databases (see [SIGMOD 92] for papers on this topic) (see [Isakowitz & al] for a hypermedia methodology based on E-R model).

We chose to take another perspective that consists in presenting the database content as a hypertext. This hypertext can be actually stored in a hypertext system or virtual, in which case, nodes and links are constructed on demand through database queries. Different users with different operating environments can then access this hypertext with a simple hypertext browsing tool, provided it recognizes the common hypertext markup language. In addition, the generated hypertext nodes can be linked to other external hypertext nodes and thus participate in a global hypertext.

Our first prototype has been used to build a W3 [Berners-Lee 94] hypertext view of the Oracle dictionary [Bobrowski 92] which comprises more than one hundred tables.

In the next section we present some features of the relational data model on which the hypertext construction is based. Note that since we use a semantic point of view, our results are also applicable to object-oriented models. In section 3 we analyse several ways of mapping database entities to hypertext components. Section 4 presents a hypertext definition language intended to specify how to construct hypertext nodes and links from the database contents. The implementation technique using W3 and Oracle is shown in section 6. Section 7 contains a comparison with other approaches. The conclusion discusses the benefits of hypertext generation techniques for W3-based hypertexts.

2. Semantic Links in the Relational Model of Data

2.1 Relations, Attributes and Tuples

In the relational model of data [Codd 70] real world entities are represented by relation tuples. The structure of these tuples is given by relation schemes which are sets of attribute names together with their domain (or data type). A tuple of a relation R takes a value for each attribute of R's scheme, this value must belong to the attribute's domain.

Relations can be either explicitly stored in the database or derived (computed) from other relations. In current relational database management systems stored relations are usually called tables and derived relations are called views. In this paper we will stay with the term relation to designate either a table or a view. We will also use the term tuple and attribute while these are often called rows and columns in commercial DBMSs.

Throughout the paper we will base all our examples on the Oracle database dictionary, which is a database comprising approximately 120 tables.

2.2 Inter Relations References

One important characteristic of the relational model is that references between tuples are based on attribute values, as opposed to object-oriented or hypertext models which use immutable object identifiers. However, relations usually possess a primary key which is composed of one or more attributes the values of which uniquely identify each tuple of the relation. For instance the attribute username is a primary key of relation **`USERS(username, user_id, creation_date,...)`** since all users must have a different user name. Primary key values can be used in other relations to refer to a particular tuple of this relation. For instance, each tuple of the relation **`TABLES(table_name, owner, table_space,...)`** refers to a tuple of USERS through the value of its owner attribute. Since the same user may possess several tables, this attribute establishes a "many-to-one" relationship between TABLES and USERS.

References through primary key values, called foreign keys, create many-to-one relationships. In order to implement many-to-many relationships between the tuple of two relations R(r-key, …) and S(s-key, …), the relational model forces to define an "associative" relation T(r-key, s-key, …) which refers to both R and S.

Inter relation references are generally intended to implement different kinds of semantic relationships such as compound-component (part-of), generic-specific (is-a), or general binary relationships. We will see later how the hypertext generation may depend on the kind of semantic relationship considered.

3. Strategies for Mapping Database Contents to Hypertexts Components

Database and hypertext models are based on radically different information representation and processing paradigms [Conklin 87][Nanard Nanard 93]. Hence it is not possible to transform a database into an equivalent hypertext. However, if one focuses on information representation, not considering information retrieval and processing, one can use different strategies to map the contents of a database to hypertext components.

3.1. Direct Tuple Level Mapping

The most straightforward way to map database entities to a hypertext structure consists in mapping each relation tuple t to a hypertext node node(t). In this situation, (a subset of) the attribute values of the tuple form the content of the corresponding node. The key attributes' values provide the identity of the node.

Semantic relationships based on attribute values can give rise to links between the nodes corresponding to the related tuples. For instance, let *t* be a tuple of **`TABLES`** which has value **`"Joe"`** on the attribute owner and let *u* be the tuple of **`USERS`** such that *u*.**`username`** = **`"Joe"`**; this induces a hypertext link from the node(*t*) to node(*u*).

If the target hypertext model does not support reverse traversal of links one must also generate a link from node(u) to node(t).

This approach is simple but it raises two problems:

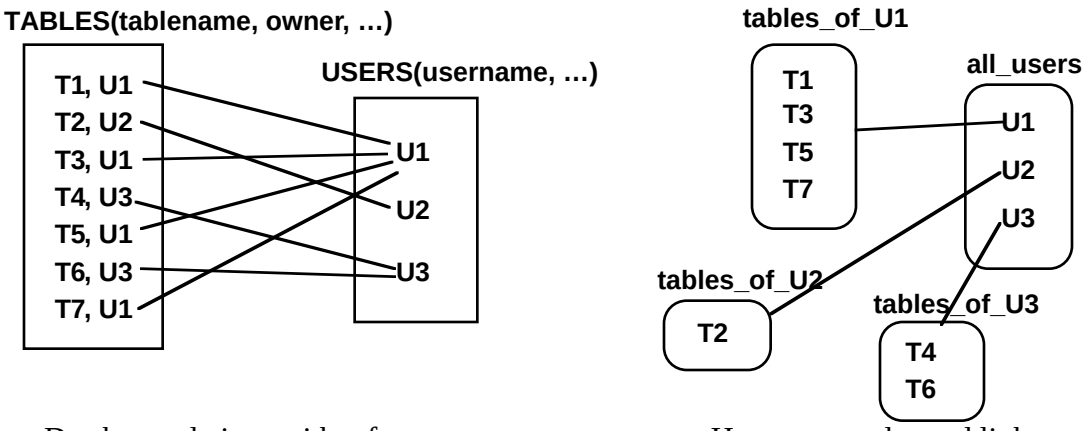
- it may generate a large amount of nodes and links, which can result in user's disorientation. For instance if user u owns 50 tables then there will be 50 links starting from u and leading to 50 nodes (one for each table).
- there is no way to see a set of data as a whole. For instance, to see the names of all the tables of user u it is necessary to navigate to each one of the 50 nodes linked to u.

However, it may be an acceptable solution for relations with many attributes and relatively few tuples.

3.2. Mapping Homogeneous Sets of Tuples to Nodes

In this approach each node is the representation of a set of tuples of a relation, nodes are structured as a set of items, each one of these being the representation of one selected tuple. This structure implies that there are now three kinds of links, namely: links between two nodes; links between an item and a node; and links between two items.

In general the content of a node will be made of the tuples of a relation which satisfy a given predicate. For instance, the content of a node **`tables_of_Joe`** could be a representation of all the tuples *t* of relation **`TABLES`** which satisfy *t*.**`owner = "Joe"`**, i.e., it will show all the tables owned by user Joe. Figure 1 below shows the hypertext structure one can create this way to represent the information content of two relations **`TABLES`** and **`USERS`** and their relationship through the foreign key owner



Database relations with references

Hypertext nodes and links

Figure 1. Mapping two connected relations to hypertext nodes and links

Compared to the previous approach, this one may greatly reduce the number of nodes and links necessary to represent the database content. In addition, it offers a more synthetic view of data. For instance, a single navigation step from a user item is sufficient to reach all the user's tables.

Another advantage of this approach lies in the possibility to directly represent one-to-many relationships, even in hypertext models (like W3) which do not support multi-headed links.

3.3 Mapping Sets of Related Tuples to Nodes

In the previous section we have shown how to define nodes by selecting sets of tuples from the same relation. We now consider nodes which are made of tuples coming from different relations. In this case the selection criteria is the existence of a relationship between the different tuples represented in a node. In other words, grouping occurs along the inter-relation reference axis instead of the within-relation axis.

This type of node construction is particularly useful to reconstruct complex entities because the relational data model does not provide constructs to represent complex hierarchic entities, i.e. entities which are composed of several other simple or complex entities. Such complex entities must be decomposed and stored as tuples of different relations.

For example, a user subschema is composed of tables, views, procedures, and triggers; a table is itself composed of a set of columns, etc. Such a complex object is represented by

- a tuple *u* of relation **`USERS(username, …)`**
- n_1 tuples of relation **`TABLES(table_name, owner, …)`** which refer to *u* through attribute owner

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.