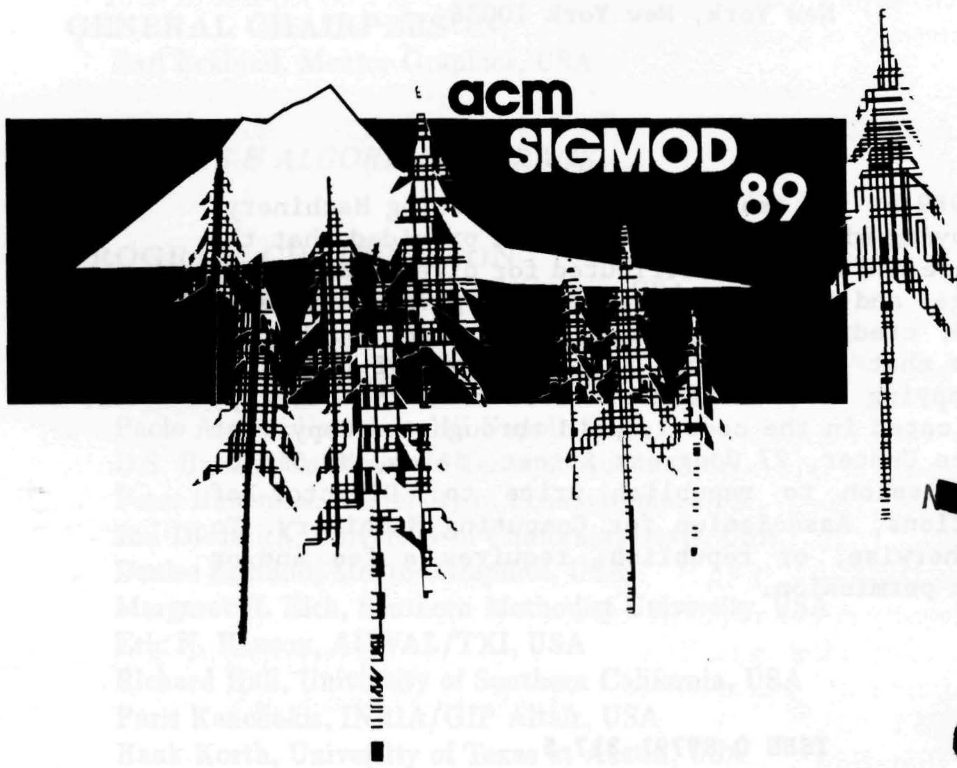
Proceedings of the1989 ACM SIGMOD International Conference on the Management of Data Portland, Oregon



SIGMOD RECORD
Volume 18, Number 2, June 1989

acm PRESS

Editors: James Clifford, Bruce Lindsay and David Maier

QA
76.9
D3
A3x
1989

The Association for Computing Machinery
11 West 42nd Street
New York, New York 10036

[redacted]

ISBN 0-89791-317-5

Additional copies may be ordered prepaid from:

ACM Order Department	*Price:*	
P.O. Box 64145	Members	$25.00
Baltimore, MD 21264	All others	$33.00

ACM Order Number: 472890

Extending a Relational Database with Deferred Referential Integrity Checking and Intelligent Joins[1]

Stephanie Cammarata (steph@rand.org)
Prasadram Ramachandra (ram@rand.org)
Darrell Shane (shane@rand.org)

The RAND Corporation
1700 Main Street
Santa Monica, CA 90406-2138
(213) 393-0411

ABSTRACT

Interactive use of relational database management systems (DBMS) requires a user to be knowledgeable about the semantics of the application represented in the database. In many cases, however, users are not trained in the application field and are not DBMS experts. Two categories of functionality are problematic for such users: (1) updating a database without violating integrity constraints imposed by the domain and (2) using join operations to retrieve data from more than one relation. We have been conducting research to help an uninformed or casual user interact with a relational DBMS.

This paper describes two capabilities to aid an interactive database user who is neither an application specialist nor a DBMS expert. We have developed deferred Referential Integrity Checking (RIC) and Intelligent Join (IJ) which extend the operations of a relational DBMS. These facilities are made possible by explicit representation of database semantics combined with a relational schema. Deferred RIC is a static validation procedure that checks uniqueness of tuples, non-null keys, uniqueness of keys, and inclusion dependencies. IJ allows a user to identify only the "target" data which is to be retrieved without the need to additionally specify "join clauses". In this paper we present the motivation for these facilities, describe the features of each, and present examples of their use.

[1] This research was sponsored by the Defense Advanced Research Projects Agency under the auspices of RAND's National Defense Research Institute, a Federally Funded Research and Development Center sponsored by the Office of the Secretary of Defense. Views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official opinion of DARPA, the U.S. Government, or any person or agency connected with them.

1. Introduction

With the advent of workstation environments, interactive software, and public domain databases, the use of DBMS is no longer limited to database administrators (DBAs), operations managers, and application programmers. Personnel in many different facets of a workplace are experimenting with DBMS for organizing, maintaining, and sharing information [McCa82]. In many cases, little or no database design is undertaken before a database is generated. Concerns for update anomalies and consistency maintenance, studied in theoretical discussions of relational database management, are rarely addressed in the practical data management activities of many organizations. Often, a novice user simply "relationalizes" a flat file into an intuitive set of tables. The resulting first normal form database implicitly relates tables through common attributes among the relations.

A complete representation of a user's application database should attempt to encode 1) the schema for every relation, 2) the data stored in the relations, and 3) the semantic relationships among relations. The first two categories are captured in every relational system. However, the semantics of the application is seldom expressed explicitly and is usually left to an individual user to interpret. Unfortunately, few DBMS tools and languages have facilities to store semantics and aid users in interpreting these semantics [Tou82, Blum87, Neuh88, Jone87]. Although common attributes between tables are based on underlying semantic relationships between relations, the relational model and its various implementations place no restrictions on the naming of attributes. Experienced users may establish their own conventions for relation and attribute names but no relational DBMS represents or enforces such conventions. Therefore, without an explicit conceptual model, it is difficult for users to access and validate the information they need [Curt81].

Developing an information model during database design is one means of expressing this information [Nava86]. However, commitment to such a formal effort is infrequent and the resulting model is usually a paper documentation aid, unavailable to interactive users. Another DBMS support tool, the data dictionary, interfaces a database to external applications by defining interface entities, application transactions, and generating reports, but is not suitable for a casual interactive user [Alle82, Dolk87]. Our approach recommends generating a knowledge base or information dictionary to capture previously implicit semantics of an existing relational schema and database. Research efforts toward integrating DBMS with expert systems have also adopted similar techniques [Reha85, Al-Z87, Schu88]. The universal relation model indirectly represents relational metadata by aiming to achieve complete access path independence [Maie82].

We have developed two capabilities, deferred Referential Integrity Checking (RIC) and Intelligent Join (IJ), which extend the operations of a relational database management system by utilizing explicit semantics and supplemental metadata combined with a relational schema. Deferred RIC is a database validation process that checks uniqueness of tuples, non-null keys, uniqueness of keys, and inclusion dependencies. This procedure can be invoked by the user or the system at specified intervals. IJ allows a user to identify only the "target" data which is to be retrieved without the need to additionally specify "join clauses". IJ subsequently navigates through the relations to generate the necessary join operations. These capabilities are supported by a metadata network constructed from both the relational schema and extended metadata stored in an information dictionary.

In the next section we present our previous work focussing on relational metadata and introduce an example database that we refer to throughout the remainder of the paper. Section 3 discusses a network representation for relational metadata which facilitates both Referential Integrity Checking and Intelligent Join algorithms. Discussion and examples of RIC are presented in section 4, and section 5 details IJ. We conclude with a discussion of the limitations, and suggestions for future work.

2. IID: An extended information dictionary

In previous work we have developed an Intelligent Information Dictionary (IID) to address the issues described above. IID serves as an interface between an interactive user and the query language of a relational DBMS [Camm88]. IID is implemented in Franzlisp Flavors running on a Sun Microsystems workstation. The dictionary communicates directly with the Ingres relational DBMS (also resident on a Sun machine) through the Lingres system, a Lisp to Ingres interface we previously implemented. Lingres provides the full functionality of the Quel query language, accessed from Lisp and Flavors.

2.1 IID functionality

IID aids a user in understanding the organization of a relational database by representing both the constructs of a relational database, and domain specific knowledge acquired from an application specialist. IID serves to augment a relational schema with supplemental metadata. By combining domain knowledge with knowledge of relational database concepts, IID supports interactive tools for browsing, customized data manipulation, and interactive value checking. Figure 1 shows the schema and extensional data of our test database, `atlas`. This geography database consists of seven relations, each with a primary key (which is underlined, "====", in figure 1). In many relational database applications, users are supplied with only the information shown in figure 1. The `atlas` database includes many of the typical anomalies found in first normal form databases.

In figure 2 we present portions of IID metadata, supplied by a domain expert, for the relations `animal` and `country` including a metadata description for the `atlas` database. In addition to metadata entries for relations, IID also represents column metadata such as value constraints, units information and conversion, and default values. IID is intended to serve the users' need for extended database capabilities, and not as a data model. However, because IID represents and uses the semantics of the database, considerable overlap exists between the capabilities of IID and ER (Entity-Relationship) or semantic modeling [Hull87]. In the future, we plan to develop a more complete modeling environment based on the existing IID framework.

One important component of an IID knowledge base is information about "interlinks". Interlinks represent the semantic information prescribing exactly how two or more relations are implicitly related. This information is generally referred to as integrity constraints, expressing structural conditions of a relational database. Knowledge of semantic interlink information and key attributes is essential for interactive users. However, without a facility like IID, there is no repository for this information. In figure 3, we show interlink metadata describing the relationships between the relations `weather`, `country`, and `vegetation`. Interlink identifiers correspond to the attribute "interlink-list" in figure 2. Specification of common columns or "join fields" are indicated in the "from-column-list" and "to-column-list". Information contained in interlinks combined with the attribute "key-list" found in the relation metadata make explicit the information needed by users for checking the referential consistency of a relational database and for manipulating the underlying data.

Until now, interlink information in IID was strictly passive. Facilities were available for a user to browse through an IID knowledge base to learn about the database. However, information about interlinks and keys did not actively contribute during the manipulation of the DBMS [Koss87, Gray88]. Our objective for the work described in this paper, was to build an operational extension to the capabilities of a relational DBMS which would automatically use this information to aid the user in retrieving data and checking for structural integrity.

2.2. Extended capabilities supported by IID

One important reason for representing relational metadata is to enable referential integrity checking (RIC). Validating referential integrity involves two categories of constraints: key constraints and referential constraints. A key constraint is implied by the existence of candidate keys and requires unique and non-null key values. Referential constraints are entailed by the relationship between a key in one relation and a foreign key in another. At any given time, the value of a foreign key in the first relation must be either null, or must be a key value in some tuple of the other relation.

Much research has addressed referential integrity, however, with the exception of Sybase, we know of no other commercial DBMS (excluding PC-based DBMS) which enforces referential integrity in real time during database manipulation [Casa88]. Our philosophy, however, is not to provide "immediate" referential checking. Instead, we are promoting "deferred" referential checking. In many cases, the benefits of immediate checking do not justify the excessive overhead [Lafu82, Hato88]. Deferred RIC can be initiated by the user or by the system at specified points in time. For example, many application databases are not designed using theoretical principles of relational normalization. Therefore, the consistency of such databses is questionable. Furthermore, users are not prevented from changing the value of a key attribute or violating inclusion dependencies. Deferred RIC fauna table

country	animal	lat	long	distfeature
india	tiger	60	75	burning eyes
ussr	penguin	70	100	majestic
usa	dog	40	100	up-to-tricks
canada	penguin	60	100	sloppy
china	dog	40	125	listless
australia	kangaroo	30	130	jumps
	elephant	60	75	trunked
india		60	75	
		60	75	
india	dog	60	75	hooded
japan	tiger	75	10	burning-eyes
india	tiger	60	75	body-stripes

country table

country	latnorth	latsouth	longeast	longwest
india	78	8	60	90
ussr	75	40	20	190
usa	50	30	125	65
canada	75	50	125	60
china	55	25	75	135
australia	10	40	115	155
	50	30	100	150
india	0	0	0	0

weather table

latnorth	latsouth	longeast	longwest	zone	avgrain
78	8	60	90	tropic	60
75	40	20	190	tundra	20
50	30	125	65	temperate	50
75	50	125	60	tundra	40
55	25	75	135	tropic	60
10	40	115	155	temperate	20
45	35	20	0	mediterranean	40
75	0	50	30		50

natwildlife table

country	natanimal	natbird
india	tiger	peacock
ussr	penguin	
usa		eagle
australia	kangaroo	kiwi
china		
canada		

economy table

country	gnp	percapita
india	210	300
ussr	2400	8370
usa	4200	17220
canada	450	17430
china	275	260
australia	231	12000
usa	4200	17220

animal table

animal	distfeature	countryfound	maxspeed	anmtype	avgheight	avgweight	avglife
tiger	burning-eyes	india	40	carnivorous	36	150	20
elephant	trunk	china	10	herbivorous	100	500	30
kangaroo	jumps	australia	25	pouched	84	180	15
penguin	sloppy	ussr	2	polar	36	100	30
dog	listless	usa	20	omnivorous	36	80	12

vegetation table

zone	avgrain	maxtemp	mintemp	treetype	maincrop
temperate	30	100	-30	deciduous	corn
tropic	60	110	0	evergreen	rice
tundra	*20*	*80*	*-60*	*coniferous*	*none*
equatorial	75	110	40	evergreen	bamboo

Figure 1: Atlas database with anomalies

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.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase® Smarter legal research.