The CCUBE Constraint Object-Oriented Database System

Alexander Brodsky Victor E. Segal Jia Chen Pavel A. Exarkhopoulo
Department of Information and Software Engineering
George Mason University, Fairfax, VA 22030
U.S.A.
{brodsky, vsegal, jchen1}@gmu.edu

1 Introduction

CCUBE is the first implementation of a constraint object-oriented database system. It is a result of a five-year effort of the research group on Constraint Programming and Databases at George Mason University.[1]

In CCUBE mathematical constraints are used as a flexible and uniform way to represent and manipulate diverse data including spatial and temporal behavior, complex modeling requirements, as well as partial and incomplete information. Application examples, for which no other unified technology exists today, include (1) constraint-based design in the presence of large data sets, (2) spatio-temporal data fusion and sensor management; (3) manufacturing, warehouse and logistics support; (4) electronic trade with complex objectives; and (5) computation of geo-physical parameters from large volumes of raw multi-dimensional data.

CCUBE, unlike most research prototypes, focuses on scalability and competitive performance, and is designed for scalable massive data applications that involve large constraint sets. To that end, the CCUBE technology has been recently selected by NASA, for the proof-of-concept phase, as a tool for high-level specification and efficient generation of products in NASA's Earth Observing System Data and Information System (EOSDIS).

Until now, most work on *Constraint Databases* (CDBs) has been theoretical and focused on expressiveness and complexity. The area of CDBs is being challenged by the question whether it will lead to a technology with a significant practical impact. This parallels, in a sense, to the state of relational databases before the first two prototypes, Ingres and System-R were developed. To move the area of CDB's from theory to practice, the objective of the CCUBE project is precisely this: to demonstrate the practical viability of the CDB technology, by building a system that can work on real-life, real-size, real-performance applications. A detailed description of the system, its theoretical underpinnings and the related work can be found in the full paper describing CCUBE in *CONSTRAINTS, Volume 2(3,4), December 1997, pp. 245 – 278.*

[1] This effort has been supported by NSF, ONR and NASA.

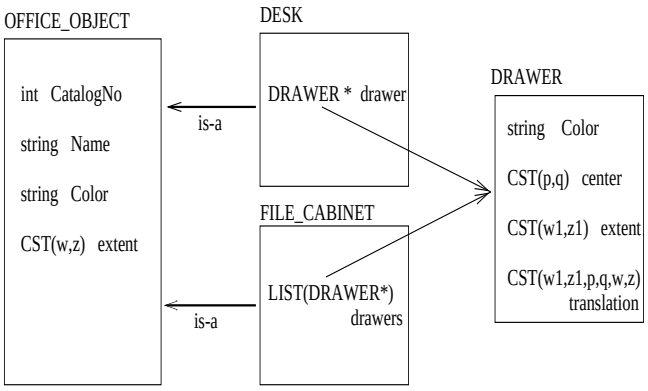

Figure 1: A Constraint Object-Oriented Database Schema

2 CCUBE by Example

Consider a simple office design application, with the schema depicted in Figure 1. The schema uses regular object-oriented features, such as ISA and composition hierarchies, and also what we call *spatio-temporal constraint* (CST) classes, such as CST(w, z), which means, intuitively, a constraint in free variables w and z.

To illustrate, consider an example of a two-dimensional desk ``my-desk'' depicted in Figure 2 as the larger rectangle. The smaller rectangle depicts the desk's

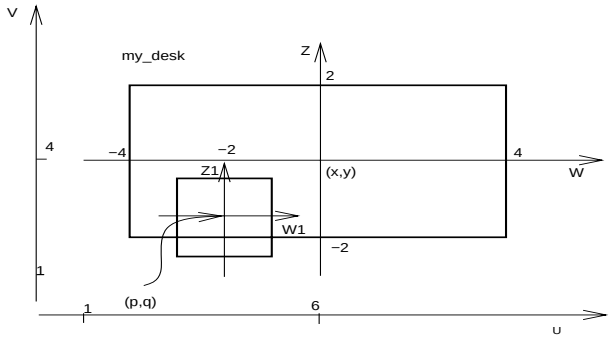

Figure 2: An Instance Of A Desk With Drawer In The Room

drawer, which may open, i.e. move relatively to the desk. Similarly, the desk may be moved in the room. In the figure, the extent of `my-desk` is the set of points $\{(w,z)|(-4 \leq w \leq 4) \wedge (-2 \leq z \leq 2)\}$ which is captured as the CST object $(-4 \leq w \leq 4) \wedge (-2 \leq z \leq 2)$ in free variables w, z, i.e., of class CST(w,z).

Similarly, the extent of the desk's drawer in the drawer's coordinates can be described by the CST object $(-1 \leq w1 \leq 1) \wedge (-1 \leq z1 \leq 1)$. The possible locations (p,q) of the drawer's center in the desk's coordinates can be described by $p = -2 \wedge -3 \leq q \leq -1$; note that the horizontal component of the center, p, equals to a constant since the drawer in the example cannot move left or right; note also that the vertical component, q, is between -3, when the drawer is fully open, and -1 when it is closed. Also, the translation between the desk's W, Z and the room's U, V systems of coordinates can be captured by the constraint $u = x + w \wedge v = y + z$, meaning that if the desk's center is at (x,y), then a point (w,z) in desk's coordinates is (u,v) in the global room's coordinates.[2]

Constraints are also used in CCUBE queries to manipulate, as well as express boolean conditions on CST objects. To illustrate, consider the following CCUBE query that retrieves all desks that intersect the room area `(3 <= u <= 4 && 8 <= v <= 10)`, assuming that the desk is centered at (6,4) and that its orientation is aligned with the room's axes (i.e. the translation equation is $u = w + x \wedge v = z + y$). For each such desk, the query also gives the desk's extent in the room's coordinates.

```
SELECT pair(dsk, CST( (u,v) | dsk_glob ))
INTO   {Bag<CST*>} result
FROM   all_desks AS {DESK*} dsk
DEFINE area   AS {CST}
              (3 <= u <= 4 && 8 <= v <= 10)
DEFINE transl AS {CST}
              (u == x + w && v == y + z)
DEFINE dsk_glob AS {CST}
(dsk->extent && transl && x == 6 && y == 4)
WHERE  SAT(area && dsk_glob )
```

The third `DEFINE` statement defines a 6-dimensional CST object `dsk_glob` (in variables u, v, x, y, w, z), which is than used in the `WHERE` and `SELECT` clauses. In the `SELECT` clause, it is projected to variables u, v, to find the extent in the room's coordinates, which is done by existentially quantifying all the other variables. In the `WHERE` clause, `SAT` is a satisfiabily test of the constraint `area` conjuncted with `dsk_glob`, which finds, whether `dsk` in the room's coordinates intersect the given `area (3 <= u <= 4 && 8 <= v <= 10)`.

Note that only with linear constraints one can express any linear transformations such as rotation, translation and stretch; check convexity, discreteness and boundness, emptiness, containment, disjointness; compute convex hulls, augment objects, change coordinate systems etc. In short, constraints are very expressive. We also claim that query systems with constraints can be implemented very efficiently for important constraint domains.

3 CCUBE Main Features

The CCUBE data manipulation language, *Constraint Comprehension Calculus* is an integration of a *constraint calculus* for extensible constraint domains within *monoid comprehensions*, which were suggested as an optimization-level language for object-oriented queries.

The data model for the constraint calculus is based on *constraint spatio-temporal* (CST) objects, which may hold spatio-temporal constraint data, conceptually represented by constraints (i.e. symbolic expressions). In the current version, linear arithmetic constraints (i.e. inequalities and equations) over reals [3] are implemented. New CST objects are constructed using logical connectives, existential quantifiers and variable renaming, within a multi-typed constraint algebra. The constraint module also provides predicates such as for testing satisfiability, entailment etc, that are used as selecting conditions in hosting monoid comprehension queries.

The general framework of the CCUBE language is the *monoid comprehensions* language, in which CST objects serve as a special data type, and are implemented as a library of interrelated C++ classes. The data model for the monoid comprehensions is based on the notion of *monoid*, which is a conceptual data type capturing

[2] CCUBE currently implements general linear constraints of any dimension, and thus can capture any object composed of multi-dimensional polyhedral sets.

[3] using finite precision arithmetic

uniformly aggregations, collections, and other types over which one can "iterate".

The ability to treat disjunctive and conjunctive constraints uniformly as collections is a very important feature of CCUBE: it allows to express and implement many constraint operations through nested queries, i.e. in the same language as hosting queries. For example, the satisfiability test of a disjunction of conjunctions of linear inequalities is expressed as a monoid comprehension query that iterates over the disjuncts (each being a conjunction), and tests the satisfiability of every conjunction (using the simplex algorithm).

In turn, the ability to express a constraint operation as a sub-query in the hosting query is crucial for what we call *deeply interleaved optimization*: it gives the flexibility to re-shuffle and interleave parts of the constraint algorithm (sub-query) with the hosting query. This re-shuffling can be done by additional global query transformations involving approximations, indexing, regrouping, pushing cheaper selections earlier, replacing sub-queries with special-purpose algorithms, and so forth.

4 Examples Of Applications

Here we discuss in more detail two examples of potential applications for CCUBE.

Electronic Trade with Complex Objectives

A typical scenario of electronic trade over the Internet is this: (1) products and services are specified and priced by numerous suppliers; and (2) bids for products and services imposed by numerous consumers. An example of a simple objective in such trades is to achieve a deal that meets specified minimal or maximal prices as sufficient conditions.

In realistic situations, however, suppliers and consumers may have considerably more complex trade objectives. To illustrate, consider a chemicals manufacturer M, capable of producing a variety of products using different raw materials. M has limitations in resources and can use a variety of manufacturing processes each corresponding to a possibly different profit function. An example of a complex trade objective for M could be finding a set of electronic bids for products M is capable of producing and a set of electronic trades of raw materials that would satisfy the existing business constraints and maximize the profit function. Furthermore, an analyst working for M may ask a variety of *what-if* questions, which go much beyond *optimize-a-function-subject-to-constraints* questions, typical in mathematical programming: Is it possible to improve profits by 5% by making an electronic trade with a raw materials supplier, and then using a better manufacturing process? How much of each raw material should be purchased to satisfy all electronic bid with profitability more than 15%? An electronic trade system, built upon CCUBE can provide a single efficient platform to all these deeply integrated aspects.

Spatio-Temporal Data Fusion and Sensor Management

A typical scenario in an air-space command and control application is this: Sensors are periodically assigned to areas of responsibility. The sensor output is collected, correlated, fused, and analyzed to form a representation of the environment. A database stores information on sensors, targets (hostile planes), target complexes (formations) and platforms (friendly planes).

Constraints are used in this application as a uniform data type for a variety of heterogeneous data: 4D-trajectories, fields of vision, interconnections among different coordinate systems, geographic regions and layers, and and templates (such as formation types) used to define expected behavior of of targets for situation assessment.

In turn, CCUBE queries capture various application activities: (1) All legal sensor assignments (i.e., targets within fields of vision of sensors) are described using constraints and constitute a search space. Finding legal sensor assignments that are optimal according to some criteria, e.g., maximal time until reassignment, is a constraint query. (2) Various coordinate transformations need to be performed. For example, since the sensor reports are relative to the platforms, spatio-temporal data is first translated into a global (uniform) system of coordinates. (3) In a number of states, the decision is made on whether separately reported targets are in fact the same one (based on their spatial and velocity proximity over a period of time); or, vice-versa, whether a target is in fact several targets. The criteria for being the same target is naturally described by constraints and is captured by a constraint query. (4) Formations of targets (e.g., a squadron of fighter jets flying a standpoint tier formation) need to be identified and maintained in the database. The type of a formation is also naturally described by CCUBE queries expressing the geometrical interposition of targets over a period of time.

5 Acknowledgments

This research was sponsored in part by the National Science Foundation (NSF) grants IIS-9734242 and IRI-9409770, and Office of Naval Research under prime grant No. N00014-94-1-1153.