PackageBuilder: Querying for packages of tuples

Kevin Fernandes*
University of Massachusetts
Amherst, USA
kevinf@umass.edu

additional authors:
Matteo Brucato   Rahul Ramakrishna   Azza Abouzied   Alexandra Meliou

ABSTRACT

PackageBuilder is a system that extends query engines to support package generation. A package is a collection of tuples with certain global properties defined on the collection as a whole. In contrast to traditional query answers where each answer tuple needs to satisfy the query predicates, each answer package needs to satisfy global constraints on the collection of tuples: e.g., a package of recipes that collectively do not exceed 2,200 calories. PackageBuilder introduces simple extensions to the SQL language to support package-level predicates, and includes a simple interface that allows users to load datasets and interactively specify package queries. Our system allows users to interactively navigate through the result packages, and to provide feedback by fixing tuples within a package. PackageBuilder automatically processes this feedback to refine the package queries, and generate new sets of results.

Categories and Subject Descriptors:
H.2.3 [Languages]: Query languages
Keywords: PackageBuilder, package queries, constraint optimization

1. INTRODUCTION

Traditional database queries define selection predicates that each tuple in the result needs to satisfy. For example, it is easy to query a database of recipes to retrieve all the gluten-free meals, assuming that the amount of gluten is reported as an attribute for each recipe. Traditional SQL queries fall short in scenarios that require a set of answer tuples to satisfy constraints collectively. Such scenarios arise in a variety of applications.

Example 1.1 (Meal planner). *An athlete needs to put together a dietary plan in preparation for a race. She wants a high-protein set of three meals for the day, that are between 2000 and 2500 calories in total. All meals should be gluten-free.*

*Kevin Fernades is a senior undergraduate student, and active contributor to the development of the meal-planner application MealBot within the PackageBuilder framework. The other contributors are listed as additional authors.

SIGMOD'14, June 22–27, 2014, Snowbird, UT, USA.
ACM 978-1-4503-2376-5/14/06.
http://dx.doi.org/10.1145/2588555.2612667.

This example cannot be expressed in traditional SQL: SQL queries can easily check conditions that apply to each tuple individually (e.g., no gluten), but they do not offer ways to specify constraints that need to be verified collectively over a set of answer tuples (e.g., enforce a limit on the total number of calories).

We present PackageBuilder, a system that augments database functionality to support the creation of *packages*. A *package* is a collection of tuples that individually satisfy *base constraints* and collectively satisfy *global constraints*. The base constraints are equivalent to regular selection predicates. For example, in the meal planner application, the gluten-free restriction is a base constraint, as it can be verified independently on each meal. In contrast, the requirement that total calories should be within 2,000 to 2,500 calories needs to be assessed over a collection of meals.

2. A QUERY LANGUAGE FOR PACKAGES

We designed PaQL, a SQL-based package query language for PackageBuilder, which allows for the declarative specification of packages. We use the meal planner scenario (Example 1.1) to highlight PaQL's key features. The following query builds the athlete's daily meal package:

```
SELECT      PACKAGE(R) AS P
FROM        Recipes R
WHERE       R.gluten = 'free'
SUCH THAT   count(*) =3 AND
            sum(calories) BETWEEN 2000 and 2500
MAXIMIZE    sum(protein)
```

The introduction of the keyword PACKAGE differentiates PaQL queries from relational SQL queries. Semantically, PACKAGE, constructs *bags* from the tuples of the base relations listed in the FROM clause. With no further constraints, there are infinitely many packages that can be built from non-empty base relations[1].

The *objective* clause MAXIMIZE is unique to packages as well: it specifies that out of all packages that satisfy the base and global constraints, the ones with larger values in the MAXIMIZE clause are preferable. The objective clause can

[1]A package can have duplicate items within it. For example a recipe can repeat multiple times.

list more than one objective, e.g., MAXIMIZE sum(protein), sum(carbs).

A package query defines two kinds of constraints. *Base constraints* are defined in the WHERE clause, and they are equivalent to selection predicates: any tuple in the package needs to satisfy all base constraints. In the example query, the base constraint R.gluten = 'free' specifies that each meal in the package should be gluten-free. *Global constraints* are defined in the SUCH THAT clause: each global constraint needs to be satisfied collectively in the whole package. For example, count(*)=3 specifies that the entire package should have exactly 3 meals.

3. COMPUTING PACKAGE RESULTS

Evaluating package queries is nontrivial: even if packages do not allow duplicate tuples, the number of possible packages is in the worst case exponential in the number of base tuples. In contrast to preference queries [6, 3], PackageBuilder does not define preferences across constraints, and relies on different evaluation methods that combine the use of constraint optimization solvers, heuristic pruning, and local search. PackageBuilder uses and extends the Tiresias query engine [4] to evaluate PaQL queries. The evaluation engine translates package queries to constraint optimization problems, and employs state-of-the-art constraint solvers to derive valid packages. Even though PackageBuilder uses the Tiresias query engine it has several differences:

- Package queries specify tuple collections (packages), whereas the how-to queries in Tiresias specify modifications (updates) to underlying datasets.
- The PackageBuilder engine allows a tuple to appear multiple times in a package result; this feature does not map to any operation in Tiresias.
- PaQL is SQL-based whereas Tiresias uses a variant of Datalog.
- PackageBuilder supports arbitrary boolean formulas in the SUCH THAT clause, whereas Tiresias only supports conjunctive how-to queries.
- PackageBuilder employs additional heuristic and pruning techniques to increase the efficiency of package queries.

4. UNDERGRADUATE INVOLVEMENT

The undergraduate author is developing the MealBot application as a demonstration of the PackageBuilder framework. MealBot allows users to seamlessly create meal plans that are customized to their nutritional needs. For example, children struggling with obesity are often prescribed a diet that includes 5 meals a day such that each meal is between 300 and 450 calories and the meal plan itself maximizes protein intake and minimizes fat intake. A child's parent can use MealBot to specify complex constraints and retrieve suitable meal plans. MealBot provides different interaction methods to specify and refine meal plans, both using PaQL as well as allowing users to enter constraints through a visual interface.

We list here the major efforts that involved the undergraduate author:

Backend support (ongoing). Interactions with MealBot are translated behind the scenes to PaQL queries. This layer, developed in Python, transforms PaQL queries to constraint optimization problems and invokes CPLEX [2] to evaluate them. This is a team effort that focuses on different evaluation strategies and design of optimization heuristics for PaQL queries. The undergraduate author is involved in the implementation of greedy heuristics that explore package solutions in a random-walk fashion. The undergraduate author is also heavily involved in testing and modifying components of the system.

Front-end development (ongoing). The MealBot interface is also a team effort, aiming to support various forms of package representations and query interactions. The undergraduate author is involved in various components of the interface support, including the development of a *suggestion* interface: when a user of MealBot selects an attribute, tuple, or value, the interface provides intelligent suggestions for possible constraints.

Data collection (completed). For the purposes of MealBot, we collected a rich dataset of recipes, including detailed nutritional data from two web sources: Yummly [5] and allrecipes.com [1]. Our web crawler used the Cheerio and PG modules of the Node.js platform. All of the collected data contains absolute nutritional information, such as calorie count, and each attribute's daily percentage value based on a 2000 calorie diet. MealBot uses PostgreSQL as the backend data management system, and all collected recipes are stored in a single table.

5. RESEARCH DIRECTIONS

We are currently exploring several interesting research directions within the PackageBuilder project. Package queries are strictly more complex than regular SQL queries, and their evaluation is not feasible with exhaustive search. We are exploring solutions that reduce that search space prior to the execution of package predicates, and we are experimenting with different greedy heuristics.

On the interaction front, we are developing smart filters that allow users to navigate the solution space more efficiently. The challenge is that typically there is a large number of valid package solutions, and each package in itself can be large. As a result, users are overwhelmed by the results, and have a hard time understanding them. Our goal is to find meaningful summarizations of the solution space, for example, grouping similar packages together and quickly finding related packages with each user selection.

6. REFERENCES

[1] Allrecipes. http://allrecipes.com.

[2] IBM ILOG. CPLEX: High-performance software for mathematical programming and optimization. http://www.ilog.com/products/cplex.

[3] W. Kießling, M. Endres, and F. Wenzel. The preference sql system - an overview. *IEEE Data Engineering Bulletin*, 34(2):11–18, 2011.

[4] A. Meliou and D. Suciu. Tiresias: The database oracle for how-to queries. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 337–348, Scottsdale, AZ, May 2012.

[5] Yummly. http://www.yummly.com.

[6] X. Zhang and J. Chomicki. Preference queries over sets. *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, 0:1019–1030, 2011.