DataPlay

Interactive Tweaking and Example-driven Correction of Graphical Database Queries

Azza Abouzied, Joseph Hellerstein, Avi Silberschatz
presented in INFO290T by Chirag Mangani

Berkeley
UNIVERSITY OF CALIFORNIA

Inspiration

- Challenges in writing complex SQL queries, especially quantified queries.
- The natural human approach to specifying quantified queries through trial-and-error.
- Limitations of SQL in this context:
    - no syntax locality
    - absence of non-answers

Quantified Queries

Quantified queries look at groups of items in a database rather than single items. They check if the whole group meets certain conditions, not just one item.

Real-Life Example

When buying flowers we might request a bouquet of "some red and white roses"
But the florist puts together a bouquet that also has lilies and pink roses.

We are therefore accustomed to specifying quantified queries by trial-and-error.

Berkeley
UNIVERSITY OF CALIFORNIA

Challenges of Quantified Queries

- What makes SQL challenging :
    - Discourages Incremental Refinement (syntax locality problem)
    - Lacks presentation of complete query effects (non-answers)

- Existing Solutions: Traditional SQL interfaces, command-line tools, basic visual query builders.

- Identified Gaps: Lack of intuitiveness, no provision for direct feedback or interactive correction, steep learning curves, and barriers for non-technical users.

Syntax Locality Problem

Existential Quantifier (At least one 'A')

```
SELECT * FROM student s, takes t
WHERE t.grade = ‘A’
AND t.student_id = s.id;
```

This query seeks any student who has received at least one 'A' grade. It directly correlates grade 'A' with the student, without considering all grades

Universal Quantifier (All 'A’s)

```
SELECT * FROM student s, takes t
WHERE t.student_id = s.id AND s.id
NOT IN
(SELECT student_id FROM takes WHERE
grade != ‘A’);
```

Checks for students who received 'A's in all subjects. Needs a subquery to exclude students with any grade that is not an ‘A’

Slight change in logical requirements, requires an extensive change in SQL Syntax

Need For Non-Answers

Existential

Answers

Nina Simone	BLUS101	A
Nina Simone	JAZZ101	A
Nina Simone	SOUL101	A
Bill Withers	BLUS101	A
Bill Withers	RYTH101	A

(a)

Non-Answers

Bill Withers	**CLAS101**	**C**
Louis Armstrong	REGA101	B
Bob Marley	BLUS101	C
Bob Marley	RYTM101	C
Bob Marley	JAZZ101	C

(b)

Universal

Answers

Nina Simone	BLUS101	A
Nina Simone	JAZZ101	A
Nina Simone	SOUL101	A
Frank Sinatra	CLAS101	A
Frank Sinatra	MELD101	A

Non-Answers

Bill Withers	BLUS101	**A**
Bill Withers	RYTH101	**A**
Bill Withers	CLAS101	C
Louis Armstrong	JAZZ101	**A**
Louis Armstrong	REGA101	B

- Without non-answers, users might misinterpret the results of the queries
- Seeing 'Bill Withers' in both answers and non-answers clarifies he fits the 'at least one A' category, not necessarily 'straight-A.'

Introducing DataPlay

Design philosophy: Simplifying query specification and debugging.

Key features:

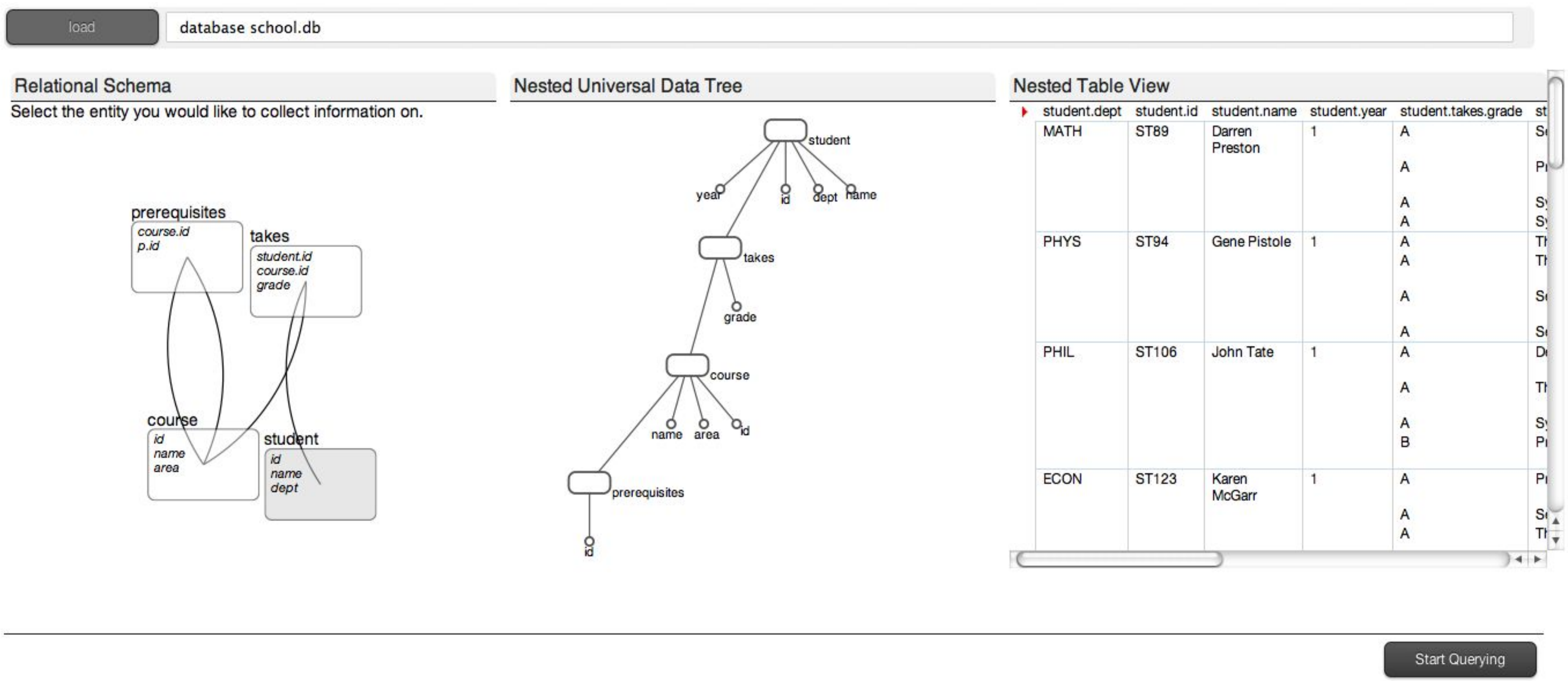
- Construction of graphical queries from user constraints.
- Direct manipulation of graphical queries, enabling semantic refinement due to syntax locality.
- Visual suggestions for query refinements.
- Interactive graphical history viewer.

Demo

https://dslam.cs.umd.edu/dataplay/DB/DataPlay_files/UIST-12.mov

DataPlay: Pivot Interface

Transforms relational database into a nested data tree



Data Model

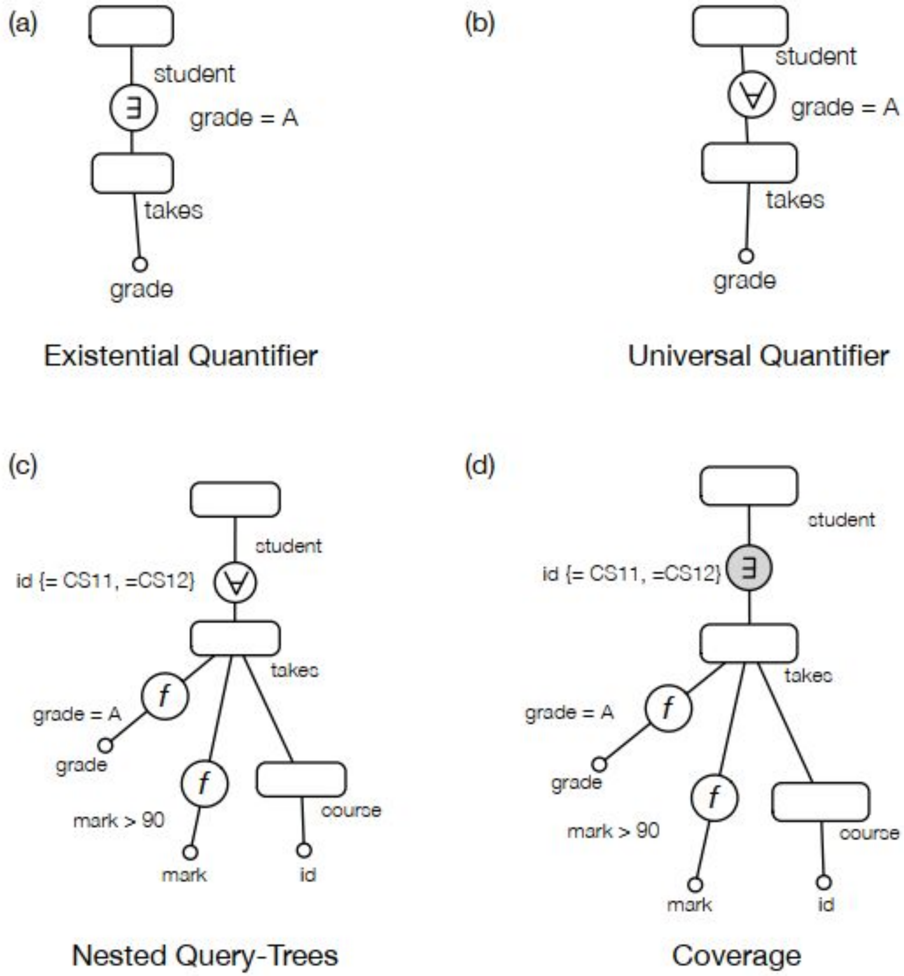
Conceptual Foundation:

Simplifies the traditional relational model into a nested universal relation.
Restriction enhances interface efficiency despite reduced expressive power.

Nested Universal Table:

- Pivot-based approach integrates multiple database tables into one universal table, termed as the "data tree."
- Transformation of any relational schema into a hierarchical data tree using a "keygraph" method.
- No physical restructuring of the database required; an abstract view is generated instead.

Query Trees



Existential Quantifier

Universal Quantifier

Nested Query-Trees

Coverage

Graphical Query Language

Query Trees:

- Specifically useful for Boolean conjunctive quantified queries.
- Overlays data trees with constraints, each node representing a relational query mapping tuples to answers or non-answers.

Operational Symbols:

- ∀ , ∋ , f to signify different operations (universal, existential, functional) within nodes

Advantages Over Traditional SQL:

- Syntax locality maintained by requiring minimal changes for major query adjustments.
- Visually intuitive manipulation options guided by system suggestions, enhancing user experience and exploration.
- Hierarchical representation mirrors the actual data structure, making it easier to understand constraint dependencies.

Query Auto-Correction

Auto-Correction Feature

- Empowers users to mark tuples with 'want in,' 'want out,' 'keep in,' and 'keep out' for answers and non-answers.
- Facilitates immediate query adjustments based on user input.

Working Mechanism:

- Generates all possible query trees by toggling parameters (quantifiers, coverage, constraints' positions).
- Presents query trees that comply with user-defined tuple memberships.

Intuitive User Assistance:

- Visual previews of query tree implications.
- Comparative insights on different query trees with a 'diff' option.

Intelligent Suggestions Ranking:

- Prioritizes query trees causing minimal disturbance to current tuple memberships.
- Assumes users' initial manual efforts are close to intended query structure.

DataPlay Interface Evaluation

Objective: Comparative study Direct Manipulation vs. Automated Query Correction.

Methodology:

- Participants: 13 database-savvy students.
- Procedure: Tutorial, hands-on session, and tasks involving fixing incorrect queries with varying complexities.

Key Results:

- Query complexity significantly impacted task completion times.
- Direct manipulation was more efficient for simpler tasks.
- Auto-correction significantly improved performance for higher complexity tasks (especially '3-tweaks' scenarios).

User Feedback:

- Both features rated highly useful, though preferences varied based on task complexity.
- Participants favored a mixed-initiative approach for optimal efficiency and ease.

Insights from User Study

Performance Insights:

- Auto-correction excels in complex scenarios, offloading cognitive demand from users.
- Direct manipulation offers quicker adjustments for experts or simpler queries.

User Preferences:

- Strong endorsement for a hybrid approach combining both features.
- Specific feedback called for improvements in the presentation of nested data and interaction processes.

Comparative Success:

- DataPlay's effectiveness highlighted by successful complex query adjustments, outperforming traditional SQL methods in similar tasks.

Takeaways and Future Work

Final Takeaways:

- The mixed-initiative interface is vital for catering to diverse complexities and user expertise levels.
- Error Mitigation for complex queries

Future Work: DataPlay v2.0

- Interactive Query Correction with improved accuracy
- Scalable Visualizations allowing deeply-nested data trees