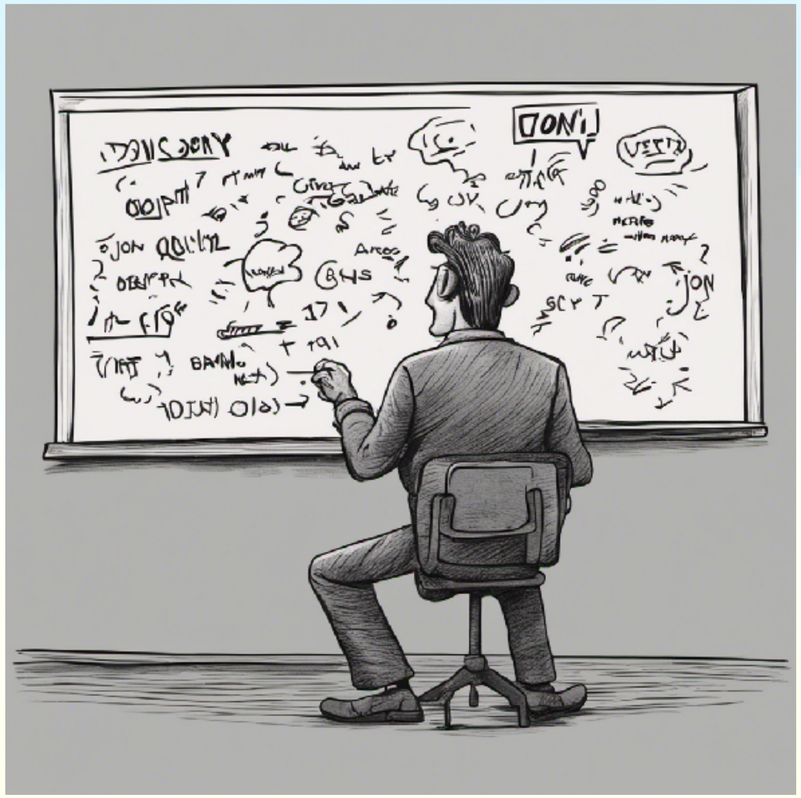
Interactive Browsing and Navigation in Relational Databases

Minsuk Kahng, Shamkant B. Navathe, John T. Stasko, and Duen Horng (Polo) Chau
Presented in INFO 290T by Shreya Shankar

Join queries are ubiquitous but hard to write

- Data is normalized in relational DBs, so many common queries require joins
- It's hard for nontechnical users to formulate join queries
- Typical visual query builders separate query construction from result presentation

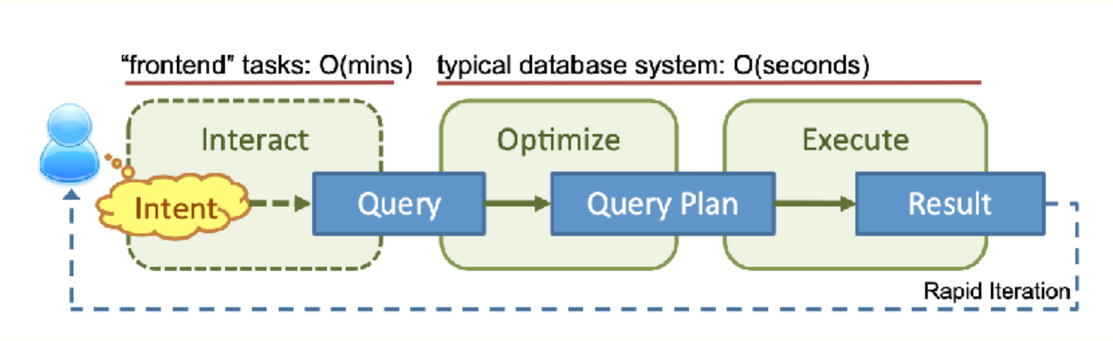


Prior Work

- Query-By-Example systems date back to 1970s
  - Prioritized usability aspects of database systems, e.g., visual query builder (graphical interface)
  - Requires precise knowledge of schema
- Keyword search systems: simpler querying (via natural language) but lack interactivity
- Direct manipulation: tools are interactive (e.g., spreadsheets) but not designed for relational data
- **Want: a tool that allows users who don't have knowledge of the schema to write queries that involve joins**

Inspiration: Guided Interaction

- Guided Interaction (Nandi and Jagadish, VLDB 2011)
  - Humans directly manipulate data and iteratively build up their queries, spending most of their time understanding the data and schemas
  - But the database world view is: given a precise query, optimize & execute it
  - What if DBs could support *guided interaction*?
  - Enumerate valid interactions, show insights, etc



Background: Presentation Data Model

- How can users build intuition for their data via query results without having full awareness of the schema?
- Presentation Data Model: sits above logical & physical layers in the DB; users can browse entities and their relationships without writing SQL
- ETable: the authors' presentation data model that represents 1:M and M:M in an interactive table

ETable: Driving Example

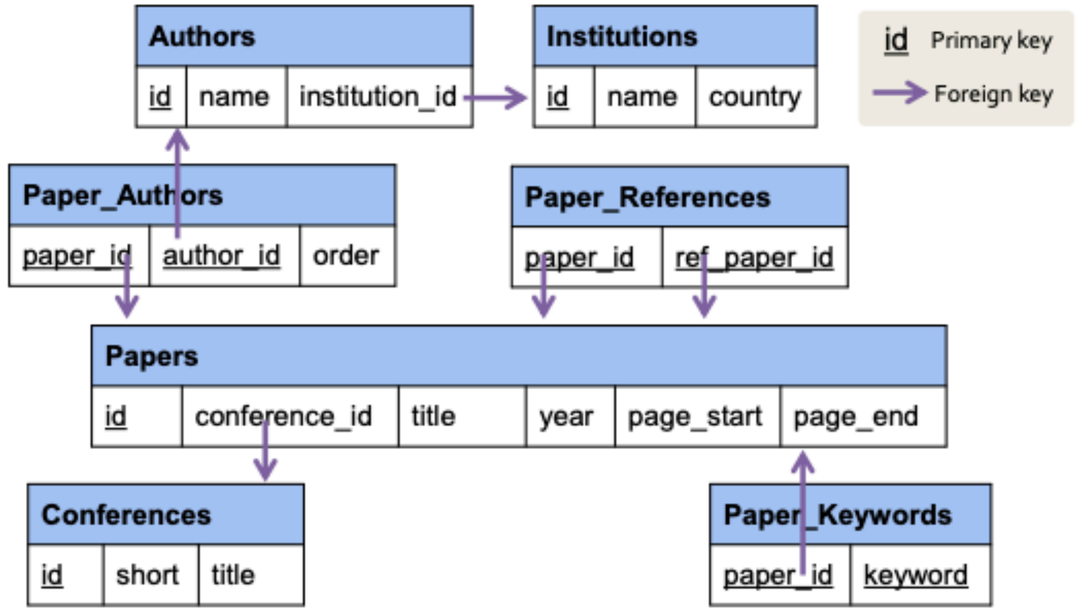


Figure 3: The relational schema of the academic data set used in this work, 7 relations in total.

ETable

Papers filtered by `Paper_keywords.keyword like '%user%' AND Conferences.acronym = 'sigmod'`

id	title	year	page_	page_	Conference acrony	Authors names	Papers (referencing) titles	Papers (referenced) titles	Paper_keyw keywords
2575	Making database systems usable	2007	13	24	SIGMOD	H. V. Jaga…, Adriane Ch…, Aaron Elki…, Magesh Jay…, Yunyao Li 7	XRANK: Ran…, NaLIX: an…, DaNaLIX: a…, Assisted q…, Towards a… 12	QueryViz:…, Exploring…, Efficient…, Homebrew d…, The intera… 25	user inter fact…, gen usability
2628	Addressing diverse user prefer…	2007	641	652	SIGMOD	Zhiyuan Ch…, Tao Li 2	Adaptive w…, Enhanced w…, Context-se…, Automatic…, Ordering t… 10	Making dat…, Supporting…, Skimmer: r…, Diversity…, Efficient… 13	informatic prefe…, da human fact algorithms
2701	Assisted querying using instan…	2007	1156	1158	SIGMOD	Arnab Nand…, H. V. Jaga… 2		Predicting…, The intera…, FreeQ: an…, Efficient…, Location-a… 8	query, key interface, autocomple inter…

Figure 1: *ETable* integrates multiple relations into a single enriched table that helps users browse databases and interactively specify operators for building complex queries. This example presents a list of SIGMOD papers containing the keyword "user" from an academic paper database collected from DBLP and the ACM Digital Library. Each column represents either a base attribute of a paper or a set of relevant entities obtained from other tables (e.g., `Conferences`, `Authors`). If a relational database were used to obtain the same information, 9 tables would need to be joined, and the results produced can be hard to interpret because of many duplicated cells.

ETable

Papers filtered by `Conferences.acronym = 'sigmod' AND Paper_keywords.keyword like '%user%'`

id	title	year	page_	page_	Conference acrony	Authors names	Papers (referencing) titles	Papers (refer titles
2575	Making database systems usable	2007	13	24	SIGMOD	H. V. Jaga…, Adriane Ch…, Aaron Elki…, Magesh Jay…, Yunyao Li	7	QueryViz:… Exploring… Efficient… d…, The in
2628	Addressing diverse user prefer…	2007	641	652	SIGMOD	Zhiwan Ch…, Tao Li	Adaptive M… nhance… ntext-se… Automatic… t…	Making dat…
2701	Assisted querying using	2007	1156	1158	SIGMOD	Arnab Nand…, H. V. Jaga…	2	Predicting intera…, F

Filter
Pivot
Sort by Count Desc

a Click reference

b Click count

c Click Pivot button

Results for each of the three actions:

a Authors filtered by `name = 'Arnab Nandi'`

id	name	Institutions name	Papers titles	
28529	Arnab Nandi	Ohio State…	Effective…, The intera…, SPIN: sear…, TopChurn:…, Beyond Bin…	12

b Authors filtered by `Papers.title = 'Making database..'`

id	name	Institutions name	Papers titles
5848	Adriane Chapman	The MITRE…	Why not, Efficient…, Making dat…, Provenance…, TIMBER: A…
9971	Aaron Elkiss	University…	Making dat…
17374	H. V. Jagadish	University…	On Effecti…, Approximat…, DataLens:…, WiseMarket…, Effectiv
17580	Magesh Jayapandian	University…	Expressive…, Making dat…, Automating…
23226	Yunyao Li	IBM Almade…	Selectivit…, Facilitati…, Gumshoe qu…, Constructi…, The Sys
28529	Arnab Nandi	Ohio State…	Effective…, The intera…, SPIN: sear…, TopChurn:…, Beyond Bi
45677	Cong Yu	Google, Mo…	Shallow In…, Interactiv…, Distribute…, Recommenda…, CloudDB

c Authors filtered by `Papers.Conferences.acronym like '%sigmod%' AND Papers.Paper_keywords.keyw`

After pivoting | After aggregation

id	name	Papers titles (filtered)		Institutions name
17374	H. V. Jagadish	DataLens:…, Assisted q…, Making dat…, Skimmer: r…	4	University…
28529	Arnab Nandi	Skimmer: r…, Making dat…, Assisted q…	3	Ohio State…
32723	Christopher Ré	Towards a…, A demonstr…	2	University…

ETable Solution

- Presents results in one table with references to entities in cells
- Direct manipulation interface that iteratively constructs queries while a user interacts with the table
  - E.g., click on the author count button to project the author names
  - E.g., click pivot button to get a view of who wrote the most papers at SIGMOD with “user” in the title
- Rows represent entities and columns represent attributes *or* related entities

ETable: Typed Graph Model

- Secret sauce for how users can explore relations without knowing the schema
- Typed Graph Database: Relational schema and entities are preprocessed into *database schema* and *database instance* graphs
- Nodes are entities (and multi-valued & low-cardinality attributes), edges are relations

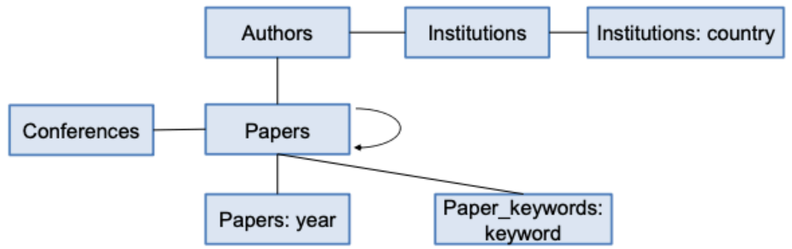


Figure 4: TGDB schema graph constructed from the relational schema in Figure 3. Each rectangle represents a node type, and each edge is an edge type.

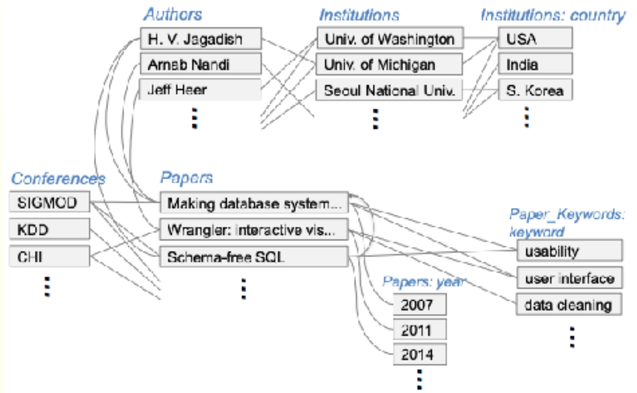


Figure 5: A part of the TGDB instance graph constructed from the academic data set used in this paper, following the schema in Figure 4. Node types shown in blue italic font.

ETable: Presentation Data Model

- Users first select specific elements of the schema and instance graphs (primary node, participating nodes, participating edges, selection conditions/filters)
  - This creates an initial *query pattern*
  - Users can apply selection, node addition, and node “shift focus” operators
- Query execution: extracts matched node instances from instance graph, then transforms result into an ETable

ETable: Presentation Data Model

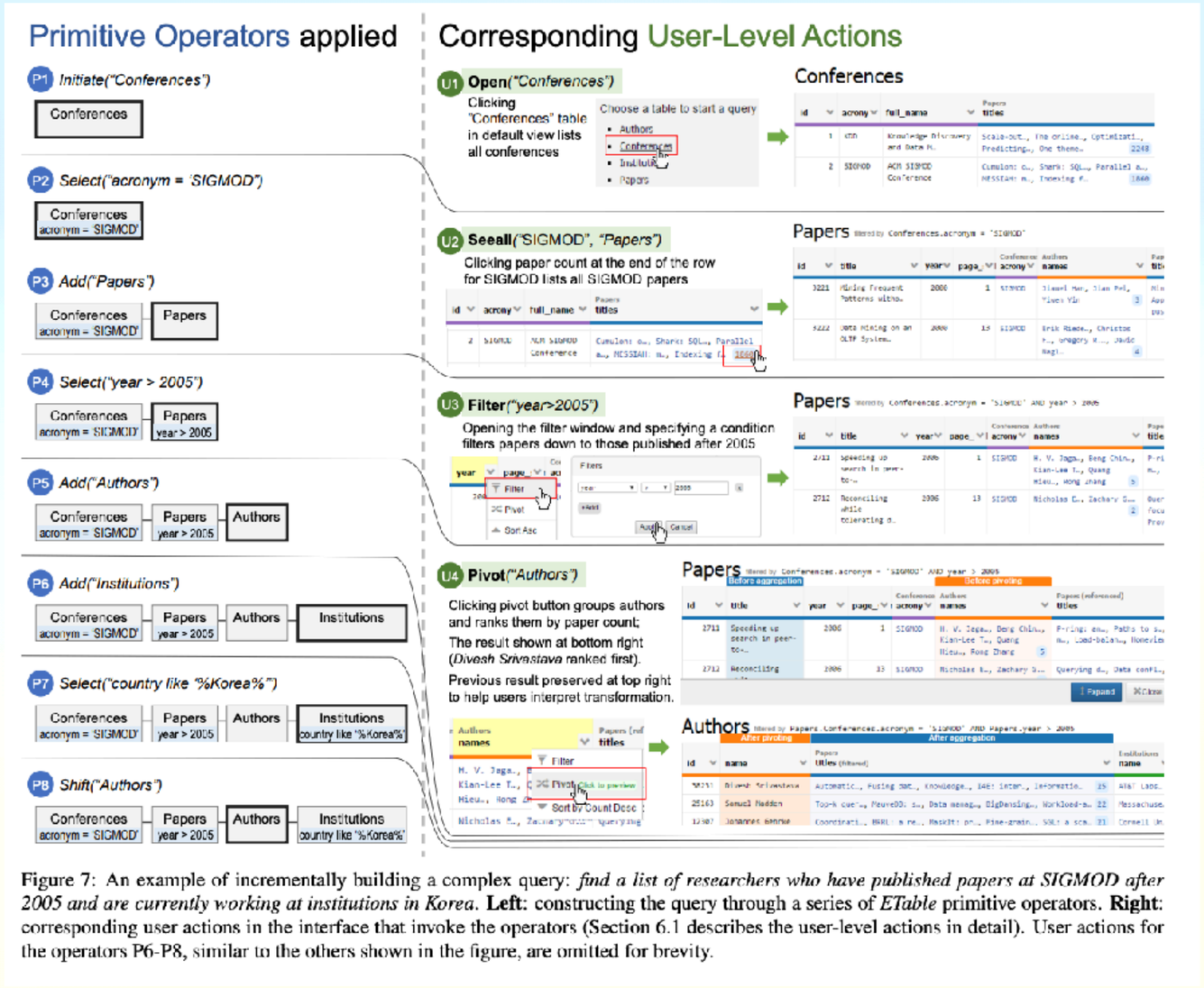


Figure 7: An example of incrementally building a complex query: *find a list of researchers who have published papers at SIGMOD after 2005 and are currently working at institutions in Korea*. **Left**: constructing the query through a series of *ETable* primitive operators. **Right**: corresponding user actions in the interface that invoke the operators (Section 6.1 describes the user-level actions in detail). User actions for the operators P6-P8, similar to the others shown in the figure, are omitted for brevity.

ETable: Architecture

- Interactive front-end in HTML, JS, and D3.js
- Application server in Python
- Postgres DB that stores TGDB schema and instance graphs in 4 relational tables: nodes, edges, node_types, edge_types

User Study

- 12 graduate students who had taken databases or had industry experience with databases
- 6 querying tasks: 1/3 finding attribute values, 1/3 filtering, 1/3 aggregation. 5 minutes per task
- Baseline: Navicat Query Builder (graphical interface)
- ETable is much faster, and users found it easy to use

Question	Avg.
1. Easy to learn	6.42
2. Easy to use	6.33
3. Helpful to locate and find specific data	6.25
4. Helpful to browse data stored in databases	6.67
5. Helpful to interpret and understand results	5.58
6. Helpful to know what type of information exists	6.00
7. Helpful to perform complex tasks	6.00
8. Felt confident when using *ETable*	5.92
9. Enjoyed using *ETable*	6.42
10. Would like to use software like *ETable* in the future	6.50

Table 3: Subjective ratings about *ETable* using 7-point Likert scales (7: *Strongly Agreed*. 1: *Strongly Disagreed*).

Future Work

- Improve expressiveness: support all relational algebra operators and more complex operations like set operations & complex aggregations
- Reuse intermediate results to improve query runtime
- Better suggestions for columns to display in the ETable