# The Role of Relational Algebra in Relational DBs*

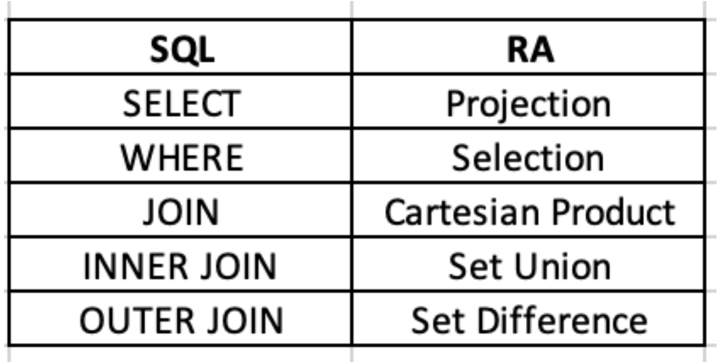
- **As a theoretical query language that helps understanding the expressive power of other query languages for relational databases**
  - The expressive power of the basic relational algebra is well-understood (FOL)
  - Computational complexity of query evaluation.
  - It serves as a good yard-stick to compare the expressive power of query languages for relational data (lower bound for any reasonable query language)
- **As the basis of set-based query evaluation primitives that can be combined and optimised to evaluate complex queries**
  - The theoretical operators form the inspiration in many RDBMSs for the primitive operators for processing relations.
  - The operators form an algebra, and as such have interesting algebraic identities that can be used as rewrite rules, queries can be reformulated as combinations of these operators and subsequently optimised by applying these rules.
  - This allows the query optimiser to **optimise** a query as follows:
    - translate it to a composition of algebra operators,
    - rewrite it to an equivalent algebra expression that is more efficient, and
    - pick for the algebra operators a specific algorithm that is the most efficient, taking into account the whole algebra expression and the current physical organisation of the data in memory.

* Jan Hidders, Prof. Computer Science at Birbeck College, University of London.

# Is SQL modelled after Relational Algebra?*

- At a high level, RA is a grouping of maths concepts, structured with the purpose of performing calculations on relations
  - A relation describes describes a group of data formed into tuples and attributes (rows & columns)
  - Any operation performed on one or more relations, results into a relation!
  - This was described as such by Edgar F. Codd, the creator of the relational model for DBs
- **SQL is an (incomplete) abstraction of Relational Algebra**
  - It makes using (tuple) relational calculus much more simple
- Operator of SQL that are included in SQL are ➔
- SQL is modelled after both, RA & TRC. However,

Tuple Relational Calculus is conceptually closer

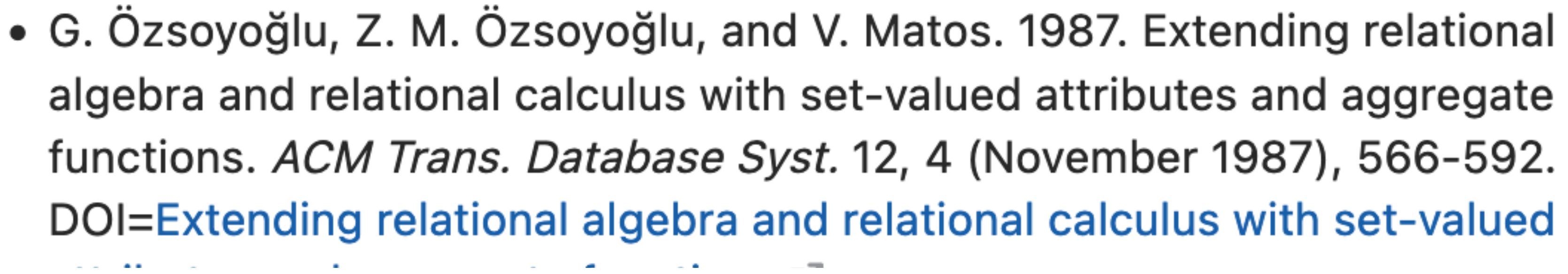


| **SQL** | **RA** |
|---|---|
| SELECT | Projection |
| WHERE | Selection |
| JOIN | Cartesian Product |
| INNER JOIN | Set Union |
| OUTER JOIN | Set Difference |

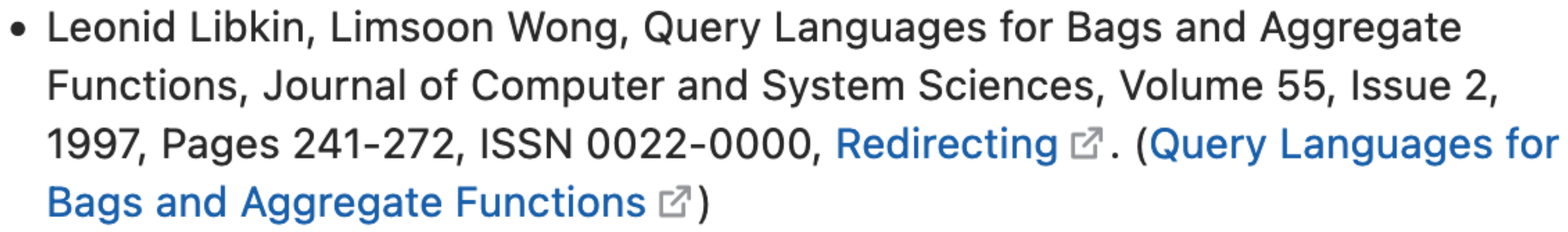
* Jan Hidders, Prof. Computer Science at Birbeck College, University of London, and Orestes Appel, MRU & IBM.

# Write query to find “average” in TRC*

- It cannot be expressed in TRC
  - TRC lacks arithmetic operators nor construction for aggregation
  - i.e. GROUP BY, SUM/MAX/COUNT, etc., which are present in SQL
- After Codd formulated the original RTC (1970), there have been a number of extensions proposed and researched



- G. Özsoyoğlu, Z. M. Özsoyoğlu, and V. Matos. 1987. Extending relational algebra and relational calculus with set-valued attributes and aggregate functions. *ACM Trans. Database Syst.* 12, 4 (November 1987), 566-592. DOI=Extending relational algebra and relational calculus with set-valued



- Leonid Libkin, Limsoon Wong, Query Languages for Bags and Aggregate Functions, Journal of Computer and System Sciences, Volume 55, Issue 2, 1997, Pages 241-272, ISSN 0022-0000, Redirecting. (Query Languages for Bags and Aggregate Functions)

* Jan Hidders, Prof. Computer Science at Birbeck College, University of London, and Orestes Appel, MRU & IBM.