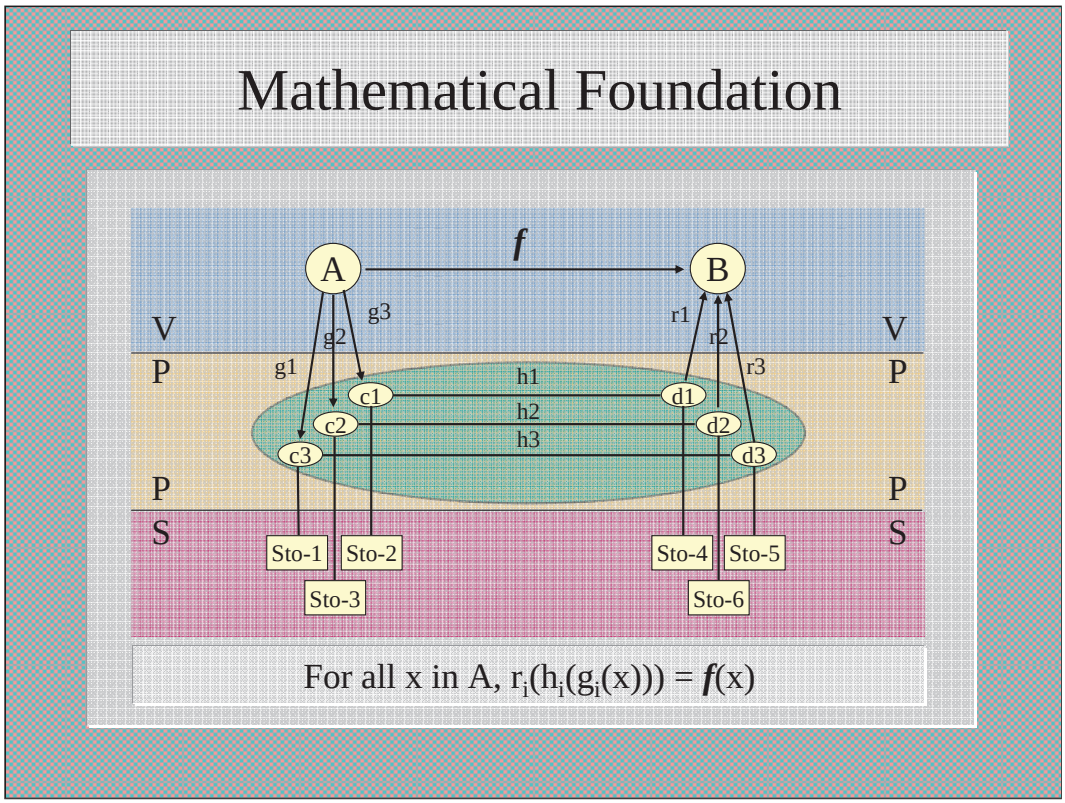

This is what mathematicians would call a commutative diagram.

Think of a system in terms of five components: three architectural layers (V, P and S) connected by two interfaces (V/P and P/S). V represents the application view level, P the database processing level and S the secondary storage level. From a mathematical point of view, the three levels are structurally distinct, disjoint spaces comprising operations and operands. The user view and processing levels are linked by the V/P interface (commonly called the query language interface) and the processing and storage levels are linked by the P/S interface (commonly called the I/O interface).

Let A and B represent any two collections of relational tables visible to application programs. We model the transformation of A into an implementation-friendly representation in P by g_i to c_i and the transformation of B into an implementation-friendly representation in P by r_i from d_i. The process, **f** , which transforms A to B, has meaning within V, but not within P. The problem: To find h_i that map c_i to d_i such that for all x in A, ri(hi(gi(x))) = f(x).

Systems, like XSP, which are based on the mapping of mathematical identities between spaces are said to be strongly data independent because no space in the mix knows how any other space organizes data. This level of independence goes beyond that supported by any RDBMS. RDBMSs insulate application logic from external data structures; XSP goes one step farther. It makes it possible to exert adaptive control over system performance (Imagine being able to replace a poor performing data structure with something better).