tional query languages and dependencies covering all the relevant material with great care. The first five parts of this monograph deal with the relational model only. In the last part the clean world of the relational model is left for more complex situations: First the issue of incomplete information is addressed, then two extensions of the relational model, complex values and object database are introduced and finally, dynamic databases, i.e the behaviour of a database in time, is rudimentarily studied. Only the chapter on database design issues is kept to the bare minimum and the reader is referred by the authors to the book by Heikki Mannila and Kari-Jouko Räihä for a more detailed exposition.

Finally, the late Paris C. Kanellakis gives a very concise and recommendable overview of database theory which alone will keep him immortal. His contribution to database theory was immense and only his untimely and tragic death deprived database theory of his further insightful contributions. Both books under review are also a tribute to his work. Paris, his wife and two children perished in an absurd air accident during a Christmas trip to their family in 1995.
normal forms for a relation specified by such dependencies were introduced, with Boyce-Codd normal form the best motivated, although not always obtainable while preserving all the functional dependencies originally used in the specification. But the quest for a general definition of a practically motivated normal form in the presence of more general dependencies was abandoned due to various failures. A notable exception is the introduction of ER-normal form derived from entity relationship design. Activity in design theory, which deserves this name, has come almost completely to a halt.

For a logician, this last problem should be a real challenge. Let us call two first order theories formulated in different vocabularies equivalent if they are bi-interpretable under some notion of interpretability. Can we formulate rigorous and robust criteria which will distinguish two different syntactic representations of equivalent theories, and which will tell us under which conditions we should prefer one over the other? This question has been traditionally shunned by logicians, but in the context of database theory it is of far reaching practical importance. If this question is to be addressed seriously in the future, new logical tools have to be developed. It is to be hoped that some logicians will face this challenge creatively.

Database theory in a more general sense comprises many other aspects of computer science, software engineering and data modeling. But each of the other aspects of database theory is not restricted to databases only: Concurrency control is inherent to any file system and is also part of operating systems, query optimization can be viewed as special aspect of compilation, and the implementation theory of databases is similar to more general questions of software engineering. Database theory in this wider sense is not a methodologically coherent theory, but a set of fragments of theories united by a common practical application. I have drawn here the very restricted picture of relational database theory, which is of interest to the logician and which is covered by the books and handbook article under review.

The two books and the handbook article under review are complementary, and the three together give the most updated and most sophisticated view of relational database theory. I have used these three texts as the main source of material for both my introductory and advanced database courses. I have profited immensely from these texts and the students of the more advanced course responded enthusiastically to these expositions.

The book of Heikki Mannila and Kari-Jouko Rääihä is the only systematic book on database design which will appeal to the logician. It addresses all the issues listed above in a comprehensive way, developing the relational model and its special case, the entity-relationship model rigorously and convincingly. However, query languages are not treated in this book.

The book by Serge Abiteboul, Richard Hull and Victor Vianu is a bit more recent and keeps the promise given in the title. It gives the logical foundations of relational databases, mostly query languages. The authors guide the reader cautiously from the basics in first order logic through all the intricacies of rela-
model theory courses and books. As both the compactness theorem and the completeness theorem fail for first order logic over finite models and the expressive power of first order logic over finite models is very limited, other techniques and languages play important roles, drawing from extensions of first order logic using generalized first or second order quantifiers and similar constructs from abstract model theory. The main results in these areas identify certain query languages as capturing low level complexity classes when the domain of the database is assumed to be linearly ordered by one of the relations. For example the polynomial time computable queries correspond to the queries expressible in first order logic augmented with a monotone least fixed point operator. The reason why this order assumption is needed for complexity classes strictly below \( \mathbf{NP} \) is rather well understood, but finding similar characterizations without an order available is one of the challenging open problems in the field. The theory of query languages can easily be adapted to other data models, such as hierarchic and object oriented databases.

The second question forms the core of database dependency theory, a question very much en vogue in the late 70ties and early 80ties. Functional dependencies assert that certain projections of a given relation are functions. Inclusion dependencies assert that a projection of a relation is a subset of a projection of another relation. Join dependencies arise when requiring that a relation can be reconstructed from some of its projections. In general, dependencies were identified as first order formulas which are domain independent (depending only on the relations) and have limited \((\forall \exists^*)\) quantifier complexity. Domain independent formulas are equivalent to their relativizations to the (definable) union of the domain of the basic relations, which gives a recursive set of formulas semantically equivalent to domain independent formulas. However, deciding whether an arbitrary first order formula is domain independent remains undecidable. It would be nice, for practical design issues, if the domain independent \(\forall \exists^*\)-formulas had a decidable consequence relation over finite relational structures, but it was soon realized that this was not the case. Clearly, the consequence problem for \(\forall\)-formulas is decidable, in fact, it is complete for non-deterministic exponential time. In the search for a practical subset of the \(\forall \exists^*\)-formulas a classification of dependencies evolved and the undecidability of the consequence problem was proven for very restricted classes of \(\forall \exists^*\)-formulas. Even for the most basic dependencies (functional and inclusion dependencies) required by standard practical database design (entity relationship diagrams) the consequence problem is undecidable. With these results, further research in dependency theory fell out of fashion, unjustifiably so, although the last open decision problem, for embedded multivalued dependencies was only solved in 1993 by C. Herrmann. Dependency theory for other than the relational data model is still in the making.

The last question above, concerning the criteria for various choices of relational data representation, is the oldest. Its first proposed solutions were formulated when only functional dependencies were considered. Various nor-
JSL Reviews of


J.A. Makowsky
Department of Computer Science
Technion–Israel Institute of Technology
Haifa, Israel
e-mail: jancs@cs.technion.ac.il

As Database Theory is maturing so are the books dealing with this subject. Many books published in the 70ties and 80ties were either introductory, and rather superficial, or reported still about ongoing research and often followed more the biographical research line of their authors than strict methodological necessities. In contrast, the 90ties are marked with the emergence of a stable body of non-trivial database theory for the relational model. The main issues concern an evolving set of sound tools for database and query language design.

Database theory has been a blessing for research in mathematical logic. Ever since the engineers in data processing have embraced the relational model of data representation as their underlying fundamental data model in the late 60ties, the search for and research in database theory has stimulated new questions and reevaluations of established techniques in mathematical logic. The basic questions concern three quite different aspects:

- Given a well designed database, what queries should and can be asked in suitable formal query languages?
- For a database to be well designed, what type of constraints (dependencies) should be allowed and maintained under updates of the database?
- Given two different relational representations (designs) of a database, can we formulate criteria which describe rigorously the various advantages and disadvantages of the two representations?

Historically, the first question was addressed last and has been satisfactorily answered since. The resulting theory is now commonly called the *theory of query languages*. It generated considerable renewed interest in the model theory of *finite models*, a part of model theory completely neglected in standard