Why You Need This Book

At the core of the IBM i, beneath the ever-growing landscape of hardware features, programming languages, and weird acronyms, lies its database: DB2 for i. This is the machine's most remarkable feature, a true differentiator that has kept the IBM i "modern" and a step ahead of the competition for decades. Everything goes through the database, from performance data to business data, in a uniquely integrated and seamless fashion. It's actually funny that the system's most remarkable feature is often neglected by system administrators, developers, and integrators. They simply ignore it or at least don't take full advantage of its power and versatility.

This book's objective is to provide people with beyond-the-basics SQL knowledge and the tools to get more out of the IBM i database. This means the book can be useful to IBM i veterans who have RPG and COBOL roots, system administrators who are looking to get more information out of the system, or even Java and .NET developers who need to "talk" to the IBM i database.

How Can You Get More Out of the Database?

This journey begins with the very foundations of SQL, by recapping the data definition and data manipulation SQL instructions, commonly referred to as DDL and DML. However, SQL is not a theoretical "thing": it's used to manipulate real data. The difference between this book and many SQL tomes is that every concept, example, and technique will be applied to a close-to-reality database. The UMADB database serves an application that is supposed to run a university. I say supposed to run because, as you'll see in a few pages, this database is poorly constructed and not very flexible. As the book progresses, this database will be improved, chapter by chapter. Applying the techniques

explained in each chapter will result in a much better database that really takes advantage of what DB2 for i has to offer.

The early chapters will recap DML and DDL, explore the system catalog, and help you build a database that is friendlier to both users (for instance, through the use of longer and descriptive column names) and programmers (with views, stored procedures, user-defined functions, and triggers, to name just a few enhancements).

Since we've entered the database-design realm, there's no harm in taking another step and improving the database, by normalizing it. An entire chapter is devoted to the database normal forms concept; here, again, the UMADB will be used as an example to illustrate every concept and technique.

The next hot topic is making the database more business-aware, by moving as many business rules and validations into it as possible. Why? Well, doing so solves a common problem with IBM i installations: the database is dumb, simply a repository of data, and the applications (usually "legacy" RPG or COBOL applications) handle all the validations in often cryptic and huge blocks of monolithic code. The problem is that more and more businesses are looking to integrate their core systems' data (which sits comfortably on an IBM i) with the rest of the business applications. These applications are Web or mobile oriented and written using different, "modern" technologies that connect directly to the database, not the "legacy" applications. If the database doesn't "know" the business rules, then the validations will have to be rewritten in new technology—every single time. Naturally, moving business rules into a database requires solid knowledge of a few concepts from the database-design realm, such as column-level constraints, triggers, and referential integrity. Don't worry, these concepts will all be explained in depth and accompanied with plenty of examples, built over the UMADB.

After reading this book, I hope that you, dear reader, will also gain a more profound and useful knowledge of SQL, particularly the "flavor" offered by the DB2 for i—whether you're a developer, system administrator, or an all-round "IT person."