Database Coding

- Software engineering principles should be applied to database coding.
- This is important to keep the code clean and to speed up development cycles.

Database Naming Conventions

- A naming convention describes how names are to be formulated.
- Naming conventions allow some information to be derived based on patterns, which helps developers to easily search for and predict the database's object names.
- Database naming conventions should be standardized across the organization.
- There's a lot of debate on how to name database objects.
- For example, some developers prefer to have prefixes or suffixes to distinguish the database object type from the names.
- For example, you could suffix a table or a view with `tbl` and `vw`, respectively.

- With regard to database object names, you should try to use descriptive names, and avoid acronyms and abbreviations if possible.
- Also, singular names are preferred, because a table is often mapped to an entity in a high-level programming language; thus, singular names lead to unified naming across the database tier and the business logic tier.
- Furthermore, specifying the cardinality and participation between tables is straightforward when the table names are singular.

- In the database world, compound object names often use underscores, but not camel case, due to the ANSI SQL standard specifications regarding identifier quotation and case-sensitivity.
- In the ANSI SQL standard, non-quoted identifiers are case-insensitive.

- In general, it's up to the developer to come up with a naming convention that suits their needs; in existing projects, don't invent any new naming conventions, unless the new naming conventions are communicated to the team members.

- In this book, we use the following conventions:
  - The names of tables and views are not suffixed
  - The database object names are unique across the database or within schemas
  - The identifiers are singular, including table, view, and column names.
  - Underscores are used for compound names
  - The primary key is composed of the table name and the suffix ID
  - A foreign key has the same name of the referenced primary key in the linked table

- Don't use keywords to rename your database objects.
- The list of SQL keywords can be found at https://www.postgresql.org/docs/current/static/sql-keywords-appendix.html

PostgreSQL Identifiers

- The length of PostgreSQL object names is 63 characters; PostgreSQL also follows ANSI SQL regarding case-sensitivity.
- If you wanted to use camel case or restricted symbols, such as dashes to name database objects, you could achieve that by putting the identifier name in double quotes.

- PostgreSQL identifier names have the following constraints:
  - The identifier name should start with an underscore or a letter. Letters can be Latin or non-Latin.
  - The identifier name can be composed of letters, digits, underscore, and the dollar sign. For compatibility reasons, the use of the dollar sign isn't recommended.
  - The minimum length of the identifier is typically 1 character, and the maximum length is 63.

Documentation

- Documentation is essential for developers, as well as business owners, to understand the full picture.
- Documentation for the database schema, objects, and code should be maintained.
- ER and class diagrams are very useful in understanding the full picture.
- There are tons of programs that support UML and ER diagrams.
- You can generate ER and UML diagrams by using graph-editing tools, such as yEd, or an online tool, such as draw.io.
- Also, there are many commercial UML modeling tools that support reverse-engineering code.

- Code documentation provides an insight into complex SQL statements.
- PostgreSQL uses `--` and `/**/` for single-line and multiline comments, respectively.
- The single-line comment, `--`, works on the rest of the line after the comment marker.
- Therefore, it can be used on the same line as the actual statement.
- Finally, PostgreSQL allows the developer to store the database object description via the `COMMENT ON` command.

Version control systems

- It's recommended you maintain your code using a revision-control system, such as Git or SVN.
- When writing SQL code, it's better to create an installation script and execute it in one transaction.
- This approach makes it easy to clean up if an error occurs.
- Also, a good practice is to create a rollback script to quickly return schema to the previous state if something goes wrong at the application level.

- Database objects have different properties: some are a part of the physical schema, and some control database access.
- The following is a proposal for organizing the database code in order to increase the **separation of concerns (SoC)**.

- For each database in a PostgreSQL cluster, you should maintain the DDL script, for objects that are part of the physical schema, and the DML script, which populates the tables with static data, together.
- The state of an object, such as a table or index in the physical schema, is defined by the object structure and the data that is contained by this object; thus, the object can't be recreated without being dropped first.
- Also, the structure of the physical schema object doesn't change often.
- In addition, the refactoring of some of the physical schema objects, such as tables, might require data migration.
- In other words, changing the definition of a physical schema object requires some planning.

- Store the DDL scripts for objects that aren't part of the physical schema, such as views and functions, separately.
- Keeping the definitions of views and functions together allows the developer to refactor them easily.
- Also, the developer will be able to extract the dependency trees between these objects.

- Maintain the DCL script separately.
- This allows the developer to separate the security aspect from the functional requirements of the database.
- This means that the database developers and administrators can work closely without interfering in each other's work.