**SQL Injection**

SQL Injection is a software vulnerability that occurs when user-supplied data is used as part of a SQL query. Due to improper validation of data, an attacker can submit a valid SQL statement that changes the logic of the initial query used by the application. As a result, the attackers can view/modify/delete sensitive data of other users or even get unauthorized access to the entire system.

**How Prepared Statements work?**

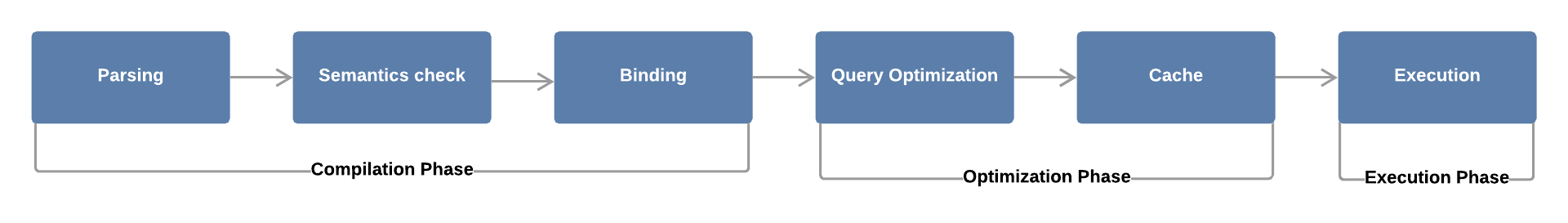


Fig 1: Oversimplified representation of SQL query processing

As you can see, the process involves six steps:

1. **Parsing** — The SQL query is broken into individual words (also called tokens). Syntax error and misspelling checks are performed to ensure the validity of the SQL query.
2. **Semantics Check**— The Database Management System (DBSM) establishes the validity of the query. Does the specified columns and table exist? Does the user have privileges to execute this query?
3. **Binding** — The query is converted into a format understandable by machines: byte code. Next, the query is compiled and sent to the database server for optimization and execution.
4. **Query Optimization** — The DBSM chooses the best algorithm for executing the query, considering the cost.
5. **Cache** — The best algorithm is saved in the cache, so next time when the same query is executed it will skip the first four steps and jump straight to the execution.
6. **Execution —**The query is executed and the results are returned to the user.

**How prepared statement blocks SQL injection**

When you think of prepared statements, think of how [printf](https://en.wikipedia.org/wiki/Printf_format_string) works and how it formats strings.  Literally, you assemble your string with placeholders for the data to be inserted, and apply the data in the same sequence as the placeholders.  SQL prepared statements operate on a very similar concept, where instead of directly assembling your query string and executing it, you store a prepared statement, feed it with the data, and it assembles and sanitizes it for you upon execution.  Great!  Now there should never be another SQL injection again.