

A



**Blueoak**  
Database Engineering

*WHITE PAPER*

# STORED PROCEDURES

---

THE BENEFITS OF USING STORED  
PROCEDURES

---

**OVERVIEW**

---

This paper will discuss the benefits of using stored procedures in a database application. A stored procedure is a user-defined module written in the scripting language of the DBMS. The language is typically a super set of ANSI SQL.

When a user-defined module is installed into the DBMS, the DBMS compiles it and check for syntax errors. If there are no errors, it builds a parse tree and determines an optimal query plan for the DML<sup>1</sup>. This information is stored in the DBMS and is reused when the module is invoked, thereby avoiding having to re-compute the parse tree and query plan.

The following are a list of reasons to use stored procedures:

- ✓ Security
- ✓ Performance
- ✓ Modularity

In the next sections, we'll provide additional details on the above and more.

---

**SECURITY**

---

Stored procedures can be used to regulate access to the database. The DBA can revoke all table and/or view access to users and only allow access via the stored procedures. New applications deployed either require new stored procedures, which will have security access re-assessed, or use existing stored procedures. The stored procedures are coded to return only the data that ought to be returned and are executed by only the users that are expressly given permission.

---

**PERFORMANCE**

---

When a stored procedure is compiled, a parse tree is built and a query plan is computed. For OLTP applications, when raw SQL is used, the percentage of time spent parsing and generating a query plan over the execution of the query is significant. Empirical analysis shows that stored procedures outperform raw SQL by as much as 10 times.

---

<sup>1</sup> DML refers to **data manipulation language**. SQL is comprised of three DML statements: insert, update and delete.

---

**NETWORK CHATTER**

---

A slow network connection can adversely affect throughput of a database application. Depending on how the front-end has coded its DML, database resources may be held for longer periods than need be due to network congestion.

Stored procedures minimize the amount of data interchanged between the client and the server, ideally, one call is mapped to one user request. Raw SQL may require several distinct interactions with the DBMS server to honor the same request.. By encapsulating the unit of work within a stored procedure, regardless of where the invoker is located on the network, the impact of network congestion or a slow network is minimized.

---

**TUNING/ABSTRACTION**

---

All database applications require SQL tuning once they are deployed. This may be in the form of adding indexes, adding an additional column to an index, re-writing SQL to nudge the optimizer to pick the correct plan, etc. The production DBA is responsible for the upkeep and tuning of the DBMS.

When the SQL is embedded in the front-end, the DBA cannot make any changes without recompiling the front-end. By using stored procedures, the DBA can tune the database without requiring the source code for the front-end. S/he can tune the SQL, get the system performing well and fold the changes back into the main development code line.

---

**MODULARITY**

---

By abstracting the SQL into stored procedures, the application can be developed in parallel by database and front-end developers. The front-end developers can concentrate on the front-end and the database developers can concentrate on the back-end. Using a seasoned database developer to code using set based processing will also provide a significant performance gain.

Relational database systems are based on relational algebra or set theory. As such, they perform better when dealing with sets of data rather than looping through a single row at a time. Empirical data shows that converting row-at-a-time processing to set based processing can realize as much as an order of magnitude or more performance improvement.

---

**BLOCKS AND DEADLOCKS**

---

It is significantly easier to keep deadlocks under control if the entire database transaction is performed within a single database request, as it is when stored procedures are used. Additionally, locks are held for shorter periods of time, thus improving concurrency and potentially reducing the likelihood of deadlocks.

## STORED PROCEDURES

A deadlock avoidance measure is to ensure that resources are accessed in a consistent method, for instance, in alphabetical order. By using stored procedures, it's easier to ensure that a standard is being followed.

---

## REFERENTIAL INTEGRITY

---

Referential integrity is used to ensure that children rows have corresponding parent rows when they are inserted or when a parent is deleted that there are no dependent children, etc. Depending on the complexity of the logic, in order to determine whether to rollback or commit the transaction, raw SQL may require multiple database calls, which reduces throughput – see **Network Chatter** and **Blocks and Deadlocks** – or the logic may need to be replicated across disparate code bases, which causes maintenance and configuration management issues.

Stored procedures and/or triggers offer the capability to centralize referential integrity so that new applications need not worry about corrupting the database.