Raima® Inc.

CHOOSING THE RIGHT EMBEDDED DATABASE MANAGEMENT SOLUTION FOR YOUR PROJECT
BUILD VS. BUY CONSIDERATIONS

Scott Meder
Senior Regional Sales Manager
scott.meder@raima.com

www.raima.com

- Ⓡ **Short Introduction to Raima**

- Ⓡ **What is Data Management**

- Ⓡ **What are your requirements?**

- Ⓡ **How do I make the right decision?**
  - **- Architecture**
  - **- Database**
  - **- Operation**
  - **- Project**

- Ⓡ **Conclusion**

- **Privately Owned Company**

- **Headquartered in Seattle, WA**

- **Long-standing embedded database market leader**

- **20,000 SDK/more than 20 million deployments**

- **Management Team background with many Fortune 100 companies including Adobe, Boeing, CA IBM, Microsoft, and Oracle/SUN.**

Business Automation

EMC² where information lives | hp invent | IBM | Atempo | XEROX

Telecom

3COM | Alcatel·Lucent | CISCO | Juniper NETWORKS | FUJITSU

Industrial automation

invensys. Wonderware | ALSTOM | BOSCH | GE | MITSUBISHI ELECTRIC

Aerospace & Defense

BOEING | BAE SYSTEMS | GENERAL DYNAMICS Strength On Your Side® | LOCKHEED MARTIN We never forget who we're working for™

Medical

BECKMAN COULTER | Instrumentation Laboratory Werfen Group | LAWSON™ | Ortho-Clinical Diagnostics a Johnson-Johnson company | SIEMENS

www.raima.com
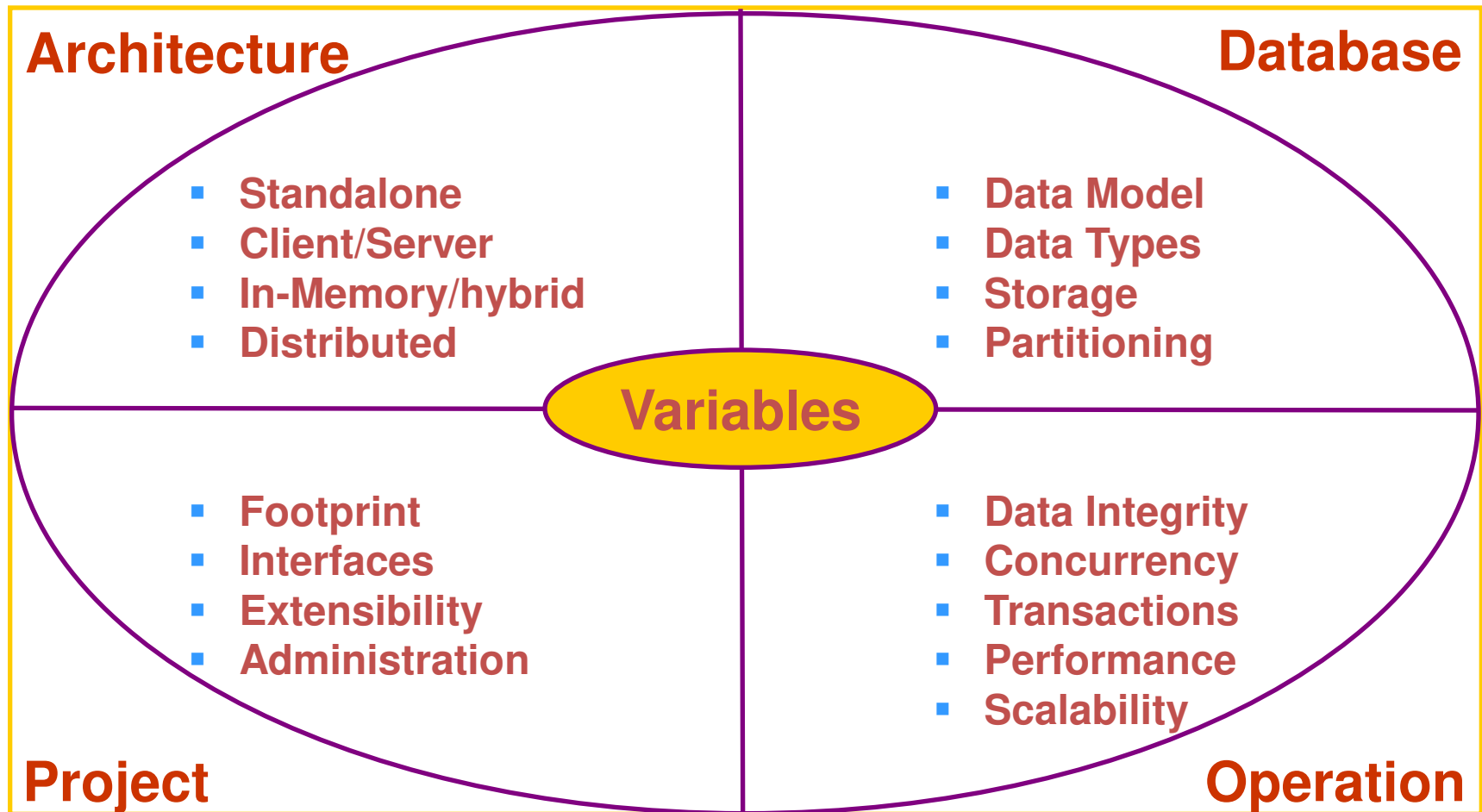
- Software that implements a data store (may be volatile)
- At a minimum, stores and retrieves data
- May have various levels of sophistication:
  - Enforce data integrity rules
  - Manage concurrency (multiple processes/threads/tasks)
  - Manage transactions (commit/rollback)
  - Manage relationships between data entities
  - Perform database recovery
  - Perform hot online backup
  - Perform data replication/mirroring/ synchronization
- Can be home grown, a commercial offering, public domain or open source
- The landscape is changing …
  - Hardware Advances
  - WEB and Connectivity
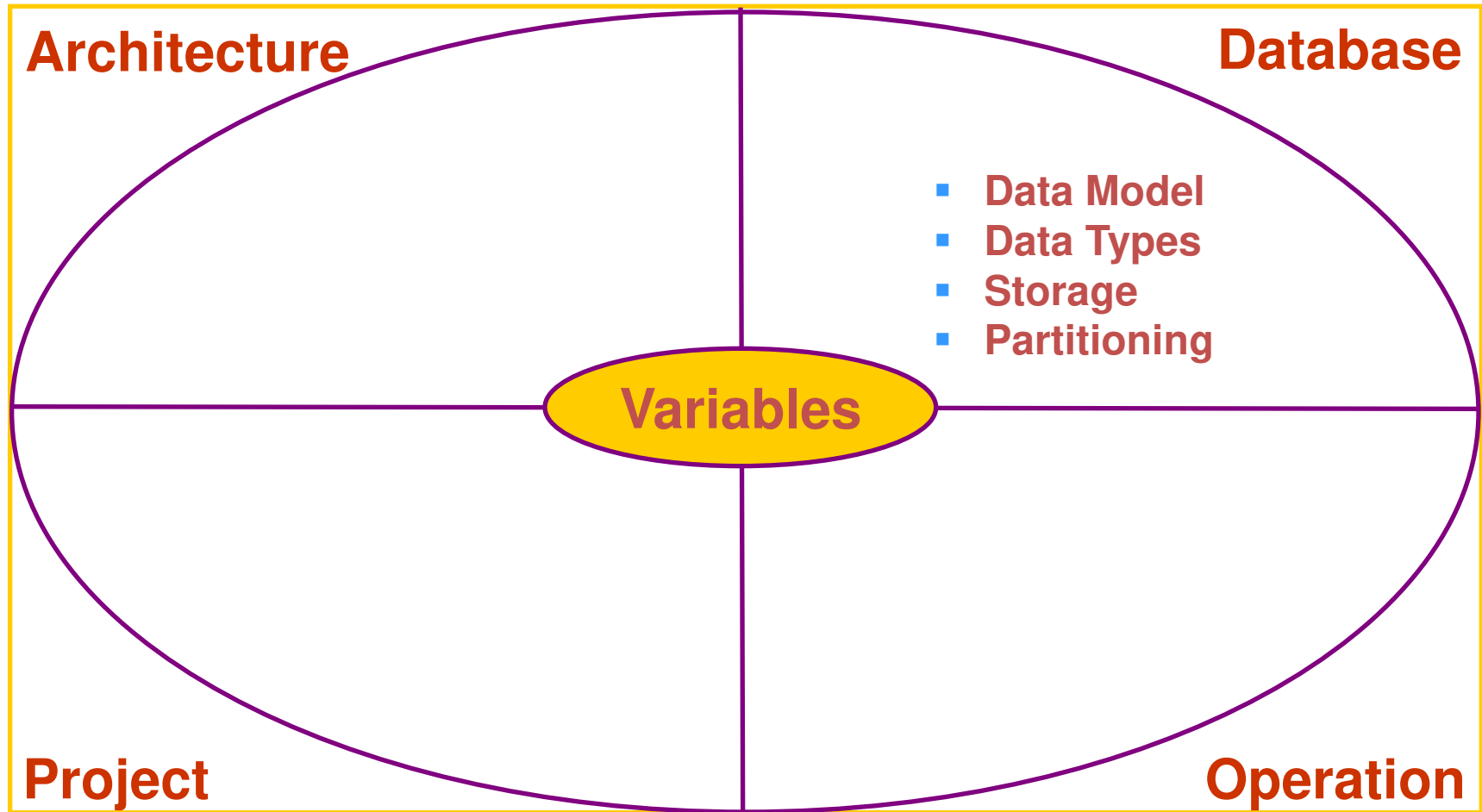 - Increased Application Complexity

- Requirements are application specific and usually diverse and conflicting:
    - Fast and predictable performance
    - High reliability and availability
    - Manage complex data in tiny footprints
    - Interact with external systems
    - Use flash memory
    - I can't loose data if my user pulls the plug

- Basically there are two options:
    - Home Grown (… for the brave)
    - Commercial, Public Domain and Open Source Offerings

Raima®

Architecture

- Standalone
- Client/Server
- In-Memory/hybrid
- Distributed

Database

- Data Model
- Data Types
- Storage
- Partitioning

Variables

Project

- Footprint
- Interfaces
- Extensibility
- Administration

Operation

- Data Integrity
- Concurrency
- Transactions
- Performance
- Scalability

- Standalone
  - Process links directly to DB manager
  - Each process has exclusive access to database
  - Very fast – no conflicts with other processes
- Client/Server
  - Code can be partitioned on client and server
  - Concurrency/locking managed centrally by server
  - Clients can be local or connected through network
  - Network performance affects client
- In-Memory
  - Can be stand-alone or client-server
  - Is my complete schema/database in-memory?
  - Can I configure in-memory data and in-memory indexes?
  - What about fault tolerance?
- Distributed
  - Mirroring for remote read-only access?
  - Replication for aggregation or to 3rd party server
  - Control of transaction traffic

Raima®

Architecture

Database

- Data Model
- Data Types
- Storage
- Partitioning

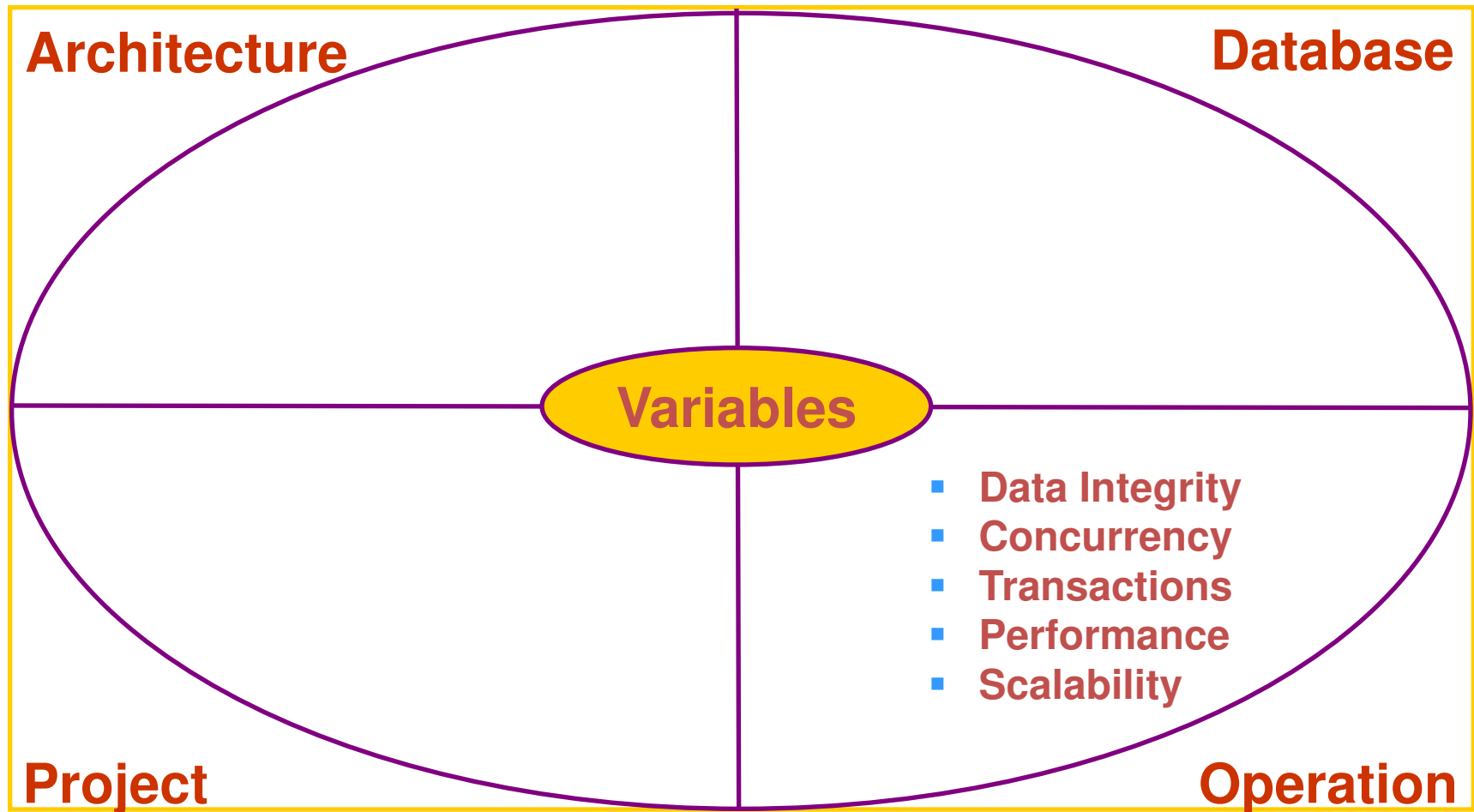**Variables**

Project

Operation

Data Model

- The model determines how precisely you can represent your real-world data structures in a database.
    - Relational
    - Network (Hierarchical, subset of the network model)

- With some solutions, like Raima Database Manager (RDM), can mix different data models within the same design.

- Don't confuse SQL with the Relational Model
    - SQL is a procedural language not a data model
    - An SQL engine will return data as rows/columns, regardless of the underlying data model

- Data types can affect performance, storage and ease of data modeling. Not all DBs are equal - choose carefully!
  - Simple data types (single-valued)
    - C-types: integer, long, float, etc.
    - SQL-types: number, date, time, etc.
  - Complex data types (multi-valued)
    - C types: arrays and structures
    - SQL (post 99):   arrays and row (structures)
  - Bulk data types
    - Text: Varchar
    - Binary: BLOB

- You need fine control, but control is DB dependent
  - Table space control
    - Map Logical table to physical file (1:1 or 1:many?)
    - Page size (record clustering affects performance)
  - Device control
    - Device types (disk, flash, CD/DVD, memory)
    - Location of physical files (local drive, network, WAN)
    - Read-only attributes (e.g. for look-up only data)
- Partitioning a DB for LAN and WAN access and scalability
  - Localize transactions
  - Unified queries
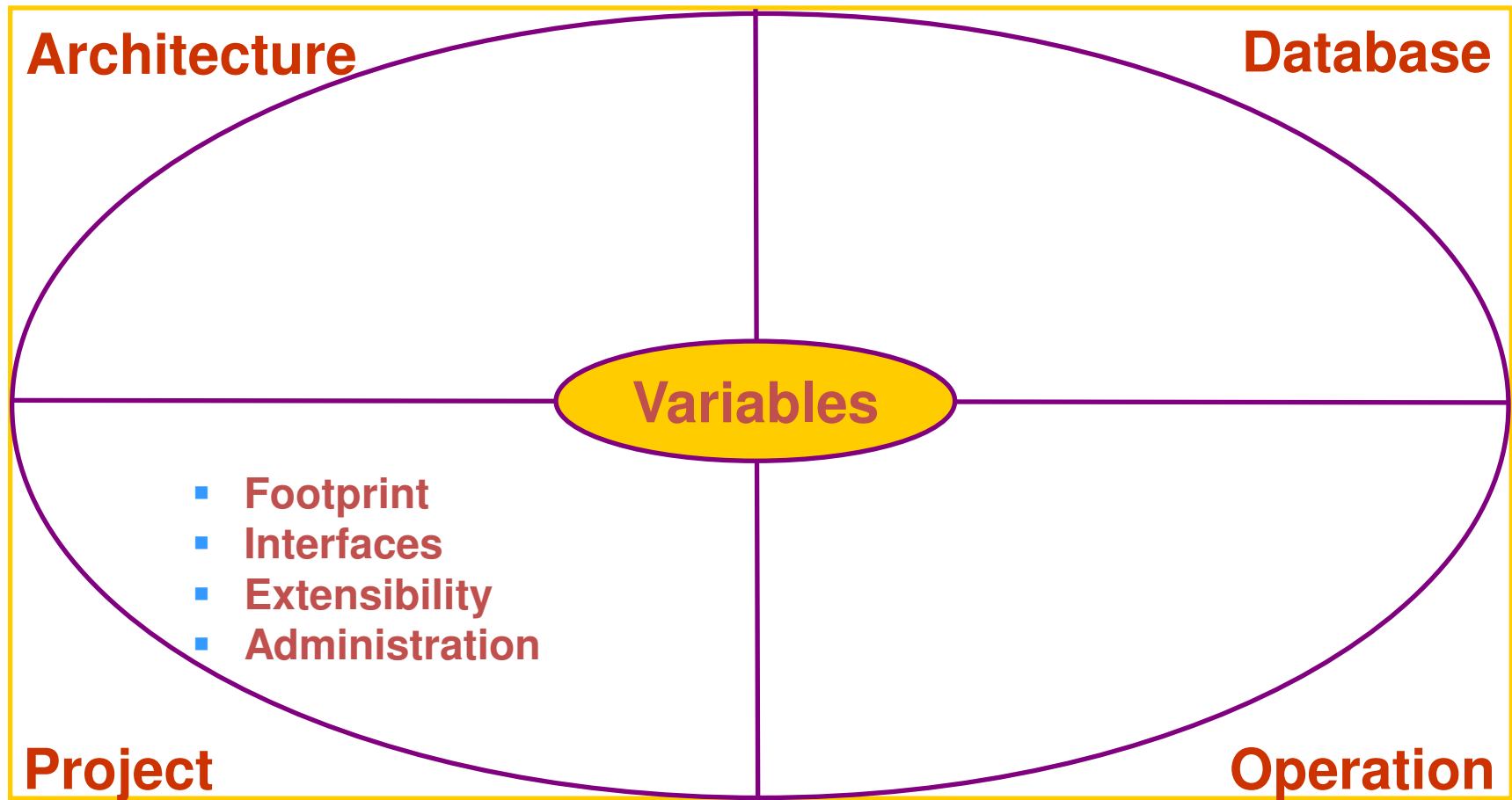 - Dynamically add partitions

Raima®

Architecture

Database

Variables

- **Data Integrity**
- **Concurrency**
- **Transactions**
- **Performance**
- **Scalability**

Project

Operation

- **If the Database (DB) does not enforce integrity you have to do it!**
    - Data consistency
        - Schema defined referential integrity
    - Data recovery
        - Automatic crash recovery
        - Data mirroring, replication and fail-over

- Concurrency provides the ability to share data. Just about all DBs provide this. Things to look out for:
  - Locking levels (granularity):
    - Database
    - Table
    - Page
    - Record/Row
    - Field (Enterprise Systems)
  - Each level of granularity increases concurrency but creates processing overhead.

- Transaction management is needed to safeguard data integrity when data is shared. If the DB does not provide it, you have to!

- ACID Properties:

  - **Atomicity** refers to the ability of the DBMS to guarantee that either all of the changes in a transaction are performed or none of them are.

  - **Consistency** refers to the database being in a legal state when the transaction begins and when it ends. This means that a transaction can't break the rules, or integrity constraints, of the database.

  - **Isolation** refers to the protection each user-process of the same database has from changes made by other user-processes. A process will see a Consistent database, and other processes will not see changes until a transaction is committed.

  - **Durability** refers to the guarantee that once the user has been notified of a successful commit, the transaction will persist, and not be undone. This means it will survive system failure, and that the database system has checked the integrity constraints and won't need to abort the transaction.

- Excluding hardware, the main performance factors are:
  - Architecture and Configuration:
    - In-memory fastest, stand-alone next fastest
    - Client-server (as fast as stand-alone if code on server)
  - Data model and DB features
    - Pointer-based joins faster than index-based
    - Asynchronous transactions and file I/O
  - Interfaces
    - Navigational APIs are fastest for transaction processing
    - SQL - impedance mismatch, query preparation/execution

- Excluding hardware, the main scalability factors are:
  - Multi-threading
  - Multi-core support
  - Cache and file I/O efficiency
  - Granularity of locks

Raima®

Architecture

Database

Variables

- Footprint
- Interfaces
- Extensibility
- Administration

Project

Operation

- Footprint and features go hand in hand
    - Deeply Embedded Systems
        - Non-server systems, 0.5 – 1.5 MB executable size
    - Lightly Embedded Systems
        - client/server systems, 1.6 – 2.0 MB executable size
    - Enterprise Systems
        - client/server systems, +80MB executable size

    - Standard or non-standard interface?
        - Standard high-level interfaces
            - ADO.Net, ODBC, JDBC
            - SQL C API

        - Non-standard interfaces
            - Low-level navigational APIs
            - Administrative APIs

        - Languages
            - C/C++, JAVA

- Client-Server DBs typically support:
    - Server managed application code
    - Stored procedures (look like virtual tables)
    - User-defined functions (scalar or aggregate)
- Non-server DBs are typically customized at the source level

- You don't want to sell your application with a database administrator!

With the increasing sophistication and complexity of embedded systems, the decision to buy a DBMS is gaining favor over building a homegrown solution.

- Build:
  - If you are brave
  - If you have a lot of time to spare
  - If you can commit to maintaining internal support expertise

- Buy:
  - If you want to avoid unnecessary development risks
  - If you want to leverage the expertise of a DB vendor
  - If you want to minimize reliability and support problems

For more information and whitepapers visit us at: www.raima.com

Raima®

Raima's mission is to provide the best technology and competence to developers of embedded systems for **collecting, storing, managing** and **moving** information while providing exceptional return on investment to our customers and shareholders.

Raima®

QUESTIONS ?