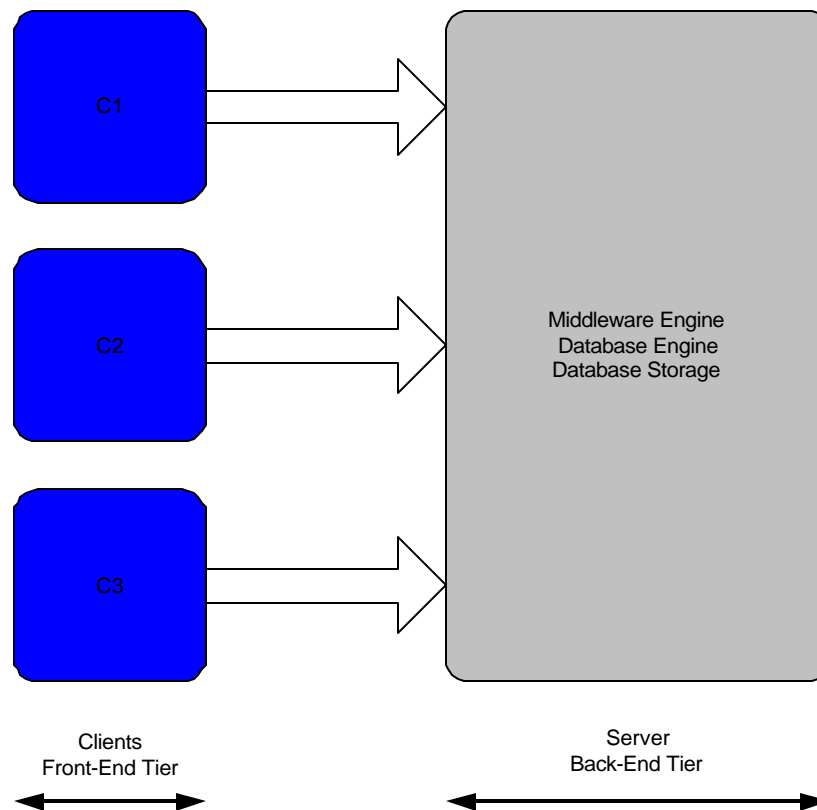


# TIERFLEET WHITE PAPER

## 1. TECHNOLOGY OVERVIEW

### 1.1 MULTI-TIER MODELS

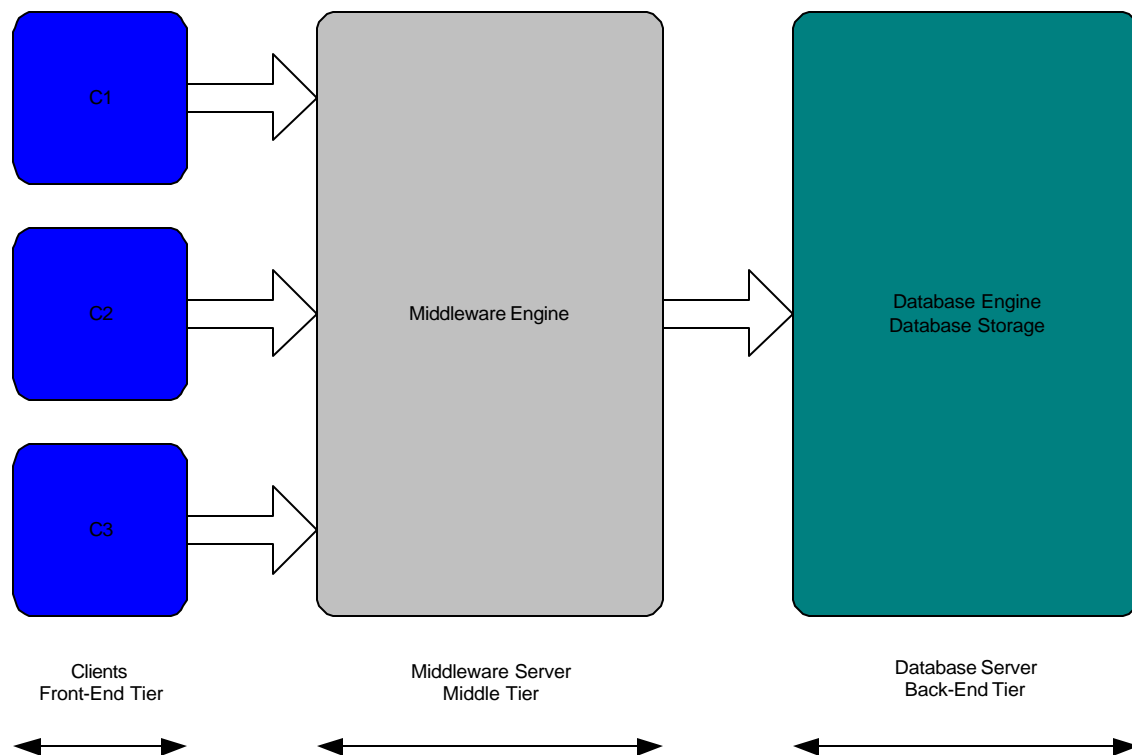
With the recent advances in information technology infrastructure and wide spread use of multi-tier business models, it makes sense to start the technology discussion by reviewing the general concept of multi-tier models and the specific role of database management systems in those models. In addition, it makes sense to review the concept of client-server business models in a limited scope. Multi-tier applications provide the infrastructure for many different business solutions today. The idea behind the client-server architecture is to provide multiple users with access to the same data. In its most basic form, a client-server is a two-tier model in which a client talks to a server across the network to request some type of remote service. For example, a title company user may point to a centralized storage repository server to request some information about a property title. The request goes to the server that is responsible for providing business logic and database content to the client. Figure 1 displays a two-tier architecture.



**Figure 1:** A two-tier architecture

In a two-tier architecture, one server is responsible for offering both business logic and database contents. A two-tier application provides multiple workstations with a uniform presentation layer that communicates with a centralized data storage layer. The presentation layer is generally the client and the data storage layer is the server. The problem with two-tier architecture is that it is not capable of adapting to changing environments and scaling with growing user and data volume. This is due to the fact that the client carries the presentation logic and a part of business logic while the server carries the other part of business logic as well as the required resources such as the database servers. In a multi-tier application environment, it is very important to address the availability and scalability issues of server side tier(s) thoroughly.

The scalability and changing environment issues in a two-tier architecture can be addressed to a certain degree by extending the two tiers to three. A three-tier architecture isolates the data processing in a central location that can be easily changed without impacting the clients. In a three-tier architecture, the presentation logic resides on the first tier namely the client, the business logic on the middle tier, and other resources such as database, reside on the back-end third tier. The middle tier of a three-tier architecture, usually the application server, handles the data processing issues and plays the role of the interface between the front-end tier and the back-end tier. It is the primary interface that directly interacts with the clients located in the first tier and hides all of the functional details of the back-end tier from the clients. In the three-tier architecture, the middle tier server interacts with the client while the back-end tier is the database engine that interacts only with the middle tier server. Figure 2 displays a three-tier architecture.



**Figure 2:** A three-tier architecture

## 1.2 ENTERPRISE METRICS IN MULTI-TIER ENVIRONMENTS

The most important enterprise features required by business solutions are guaranteed availability and performance. In order to provide guaranteed availability, a solution must provide both high availability and scalability for the target applications. Load balancing is a crucial component of building scalable systems. High availability guarantees that end users always receive the quality of service desired. Without first determining availability, scalability alone is of limited value. A general discussion of high-availability and scalability solutions follows.

The first generation of guaranteed availability solutions have not completely solved both high availability and scalability issues. Most first generation solutions provide rudimentary load balancing in order to address the issue of scalability and provide some degree of increased availability. However, they are not capable of performing thorough availability analysis before deciding where to send traffic. Directing traffic to an overburdened server does not provide the expected quality of service. This leaves businesses vulnerable to providing customers with poor response or missing content. Since customers can easily move on to the competitor offerings to find what they are after, businesses require a better solution.

The second generation of guaranteed availability solutions remedied some of the problems of the first generation by presenting a single virtual address to end users and mapping requests sent to that address to multiple servers. However, these second generation solutions were not capable of providing true high availability. These devices were passive in nature and performed no active verification of the availability of servers or content on those servers. Instead, they relied on the failure of actual traffic to detect that a server was incapable of responding. While providing adequate scalability, these solutions fall short of providing the functionality required by businesses demanding continuous uptime.

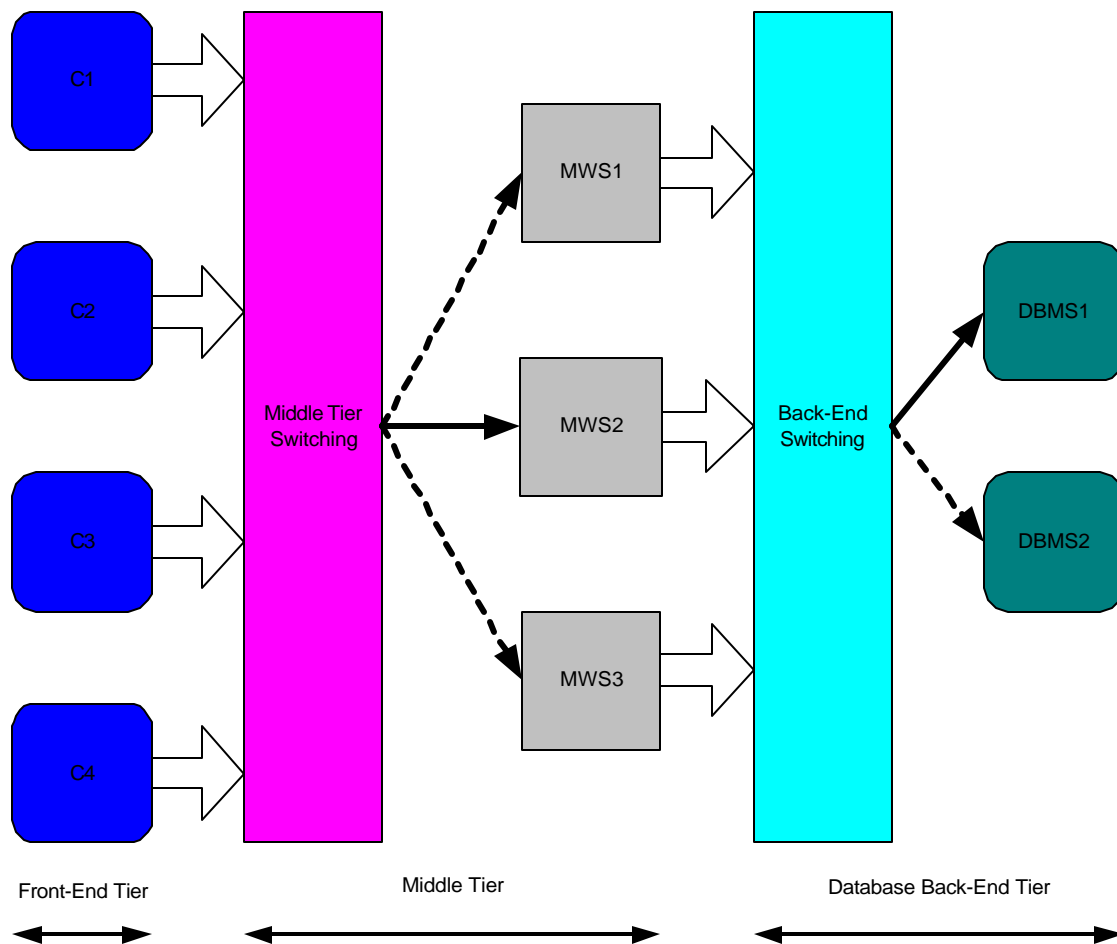
The followings are the factors that can impact availability of a business solution. It is important to note that these factors almost equally apply to different tiers of a multi tier model.

- Server failure - The server becomes unavailable due to a hardware or operating system failure.
- Software failure - Individual applications can hang or stop responding even though other applications are healthy.
- Content failure - The server and application are working properly but are responding to requests with a response that does not contain the right content.
- Heavy traffic load - Servers have a well-defined response curve in relation to load. As traffic increases, they are able to respond to requests promptly until the server reaches a point at which it stops responding to any request. This behavior can be viewed as binary in nature - the server is either on or it is off.
- Network unavailable - If the link between the server and the outside world becomes unavailable, the server becomes unreachable.

A state-of-the-art third-generation guaranteed availability solution can address the first issue by proactive hardware monitoring of its server components. The second issue can be addressed by proactive monitoring of software daemons running on the servers. Application monitoring on the servers can guarantee the content failure issue is addressed. The fourth issue is addressed by introducing a time out value. If any of the servers fail to provide the response within the specified time, another server is asked to respond to the content inquiry. The last issue can be addressed only if the servers are geographically distributed in which case introducing a time out value can solve the problem. Otherwise, all of the servers located on the same segment of a network are usually impacted by network congestion the same way. In summary, a third-generation solution is an intelligent system that can complement and manage incoming traffic. The traffic can, then, be dynamically distributed across a group of servers running a common application while making the group appear as one server to the network. This has created the need for a more intelligent system that can complement and manage incoming traffic – a function known as load balancing. In this type of scenario, traffic can be dynamically distributed across a group of servers running a common application while making the group appear as one server to the network. This approach allows the traffic to be distributed more efficiently, offering greater economies of scale, and providing significantly greater availability. Such systems monitor the health of the servers and make decisions on where to route traffic to optimize performance and availability. This ensures the requests will be sent to the most available servers, providing excellent and predictable Quality of Service (QoS). Finally, a successful solution appears in the form of a turnkey solution that offers a simplified management tool. Turnkey systems are “thin server appliance” software/hardware solutions that can easily enhance traffic performance. Such systems are optimal for all-inclusive solutions that manage a specific server/application, are plug-and-play and maintain an open and flexible architecture to grow over time. The Turnkey design usually places two redundant integrated servers between the back-end server array and the clients, operating jointly as parallel and hot-spare servers. This redundancy offers fail-safe, cost-effective operation and significantly minimizes maintenance. Servers can be upgraded and managed without any downtime or effect on the network. Additionally, turnkey systems rely on an operating system neutral approach allowing the organization to implement any type of application including but not limited to web servers into the mix. A turnkey approach offers a strong balance between high functionality, performance, flexibility and cost-effectiveness.

The typical architecture of three-tier models deployed by business solutions is evolving with guaranteed availability and performance requirements. For a typical multi tier model consisting of the front-end tier, the middle tier, and the back-end tier, a relatively large number of first tier client requests simultaneously asking for remote services can gradually create a bottleneck out of the components of the second and third tiers. In addition, the failure of any of standalone middle-tier and back-end tier servers leads to unacceptable failure of the overall multi-tier model. To address these issues the three-tier architecture has been expanded to the multi-tier architecture in which the second and third tiers consist of multiple layers. Multi layer web server solutions were first introduced to address scalability and high-availability issues for the middle tier in the three-tier architecture. They did so by balancing the load against a number of middle tier servers while keeping the functionality transparent from the first layer clients. Companies such as F5 Networks and Ipivot have all successfully offered solutions belonging to this category. The lack of a focused effort to address the scalability and performance issues in the back-end tier has led different businesses to come up with ad-hoc implementation-dependent efforts to cover the back-end scalability and performance gap. TierFleet's

technology is introducing a novel implementation-independent model to address scalability, availability, and performance issues for the back-end database tier in a way comparable to what has been offered for the middle tier. However it is important to note that despite the similarities in concept with the middle tier scalability models, the design and implementation techniques of back-end tier are drastically different with those of middle tier. Figure 3 shows an overall-scalable multi-tier environment deploying a scalable, highly available solution relying on middle tier scalability solutions as well as TierFleet back-end scalability solution.



**Figure 3:** A multi-tier solution deploying TierFleet back-end architecture

## 2. PRODUCT REVIEW

### 2.1 QUERYMASTER PRODUCT CONCEPT SUMMARY

TierFleet technology, according to what was discussed in the previous section, is the underlying foundation for building a third generation guaranteed availability product for the database back-end tier. The high performance, highly scalable, switching product is capable of eliminating the database back-end bottleneck from the business solutions. Such a product is not database dependent. It can be integrated with any database engine and can offer a unified view of data among multiple back-end database servers. The underlying technology is compatible with all SQL-based RDBMS and ORDBMS engines. It is also fully compliant with the new XML-based standards such as W3C XQuery (<http://www.w3.org/TR/xquery>) aiming at creating universal XML-based data platforms for different DBMS implementations. The connectivity interface is fully transparent to any kind of client connectivity tool of the target database. Different flavors of the family of **QueryMaster** products taking advantage of the same core technology are certified for different database platforms. These platforms include but are not limited to PostgreSQL, Microsoft SQL, Oracle 9i, and IBM DB2.

The product is offered in a turnkey form factor also known as a thin server appliance. It is originally offered as a single or a dual (high availability) rack mountable unit. The thin server is a low cost bundled model that runs on standard Intel-based hardware platform and standard Linux operating system platform. The core, database replication, and management software rely on patent pending software technology to offer enterprise database features such as high-availability, dynamic scalability, proportional performance enhancement, and layer-7 application switching. The user-friendly management tool is used to setup the QueryMaster server(s) and control the interaction with the back-end database servers. In addition, TierFleet makes use of a set of extensive home grown benchmarking tools for testing the functionality and performance of the suite of products.

TierFleet QueryMaster product plays the role of the interface between the database client and the back-end database server array in a business solution. The product offers enterprise metrics: high-availability, scalability, performance enhancement, and application layer load balancing also known as layer 7 switching. Once the product is placed in front of a number of database engines, it appears as the database itself to any database client using a standard or proprietary database connectivity interface. Some of the connectivity interface examples are JDBC, ODBC, or PostgreSQL "PSQL" client utility. Simply put, there will be no change in the operation of the database solution and the deployment of the connectivity tools from the perspective of the database client as QueryMaster products are simply introduced as the database to the client. The client simply uses the network and server settings of the QueryMaster product when trying to use a database connectivity tool such as JDBC. The QueryMaster server accepts an incoming connection request assuming there is no problem with database user authentication and requested database resources.

The management tool is offered in the form of a web-based graphical user interface (GUI). The GUI provides a centralized point of management allowing access to the server from a browser remotely. The GUI provides the system administrator with the full functionality set including the naming and network settings of the high-availability pair of QueryMaster servers as well as the back-end database servers, the dynamic choice of load-balancing algorithm, the ability to start/stop services, and dynamic overview of solution's vital statistics. The statistics can help system administrator monitor the distribution of SQL queries in the specific environment and select the load-balancing algorithm accordingly. For the back-end database server management, the GUI also provides the system administrator with an option to dynamically add new database servers or replace/remove existing database servers without impacting the operation of the overall database solution. When adding a new database server, the GUI allows for dynamically synchronizing the view of new database data content with that of the existing databases prior to bringing the new database online. The management information are saved and synchronized among multiple QueryMaster servers. The GUI has been designed and implemented from the ground up to provide a unique look and feel for private labeling and OEM prospects without any programming effort.

## **2.2 QUERYMASTER PG1000 AND PG2000 PRODUCTS**

TierFleet QueryMaster PG1000 and PG2000 product lines are the first in the family and have been certified for PostgreSQL database. In the course of implementation, a patent-pending database independent replication scheme has also been deployed to create and preserve the unified view of data among back-end database server array nodes. While PG1000 product is a single server model, PG2000 product is a clustered server remaining operational independent of the failure of any of the two servers. The PG2000 servers allow for online replacement of any faulty component or any of the two QueryMaster servers while offering continuous availability. Both products are capable of dynamically adding/removing back-end database servers and coping with the related database synchronization issues while providing continuous availability.

### **COMPONENTS**

The followings describe single QueryMaster server appliance components.

#### **Hardware**

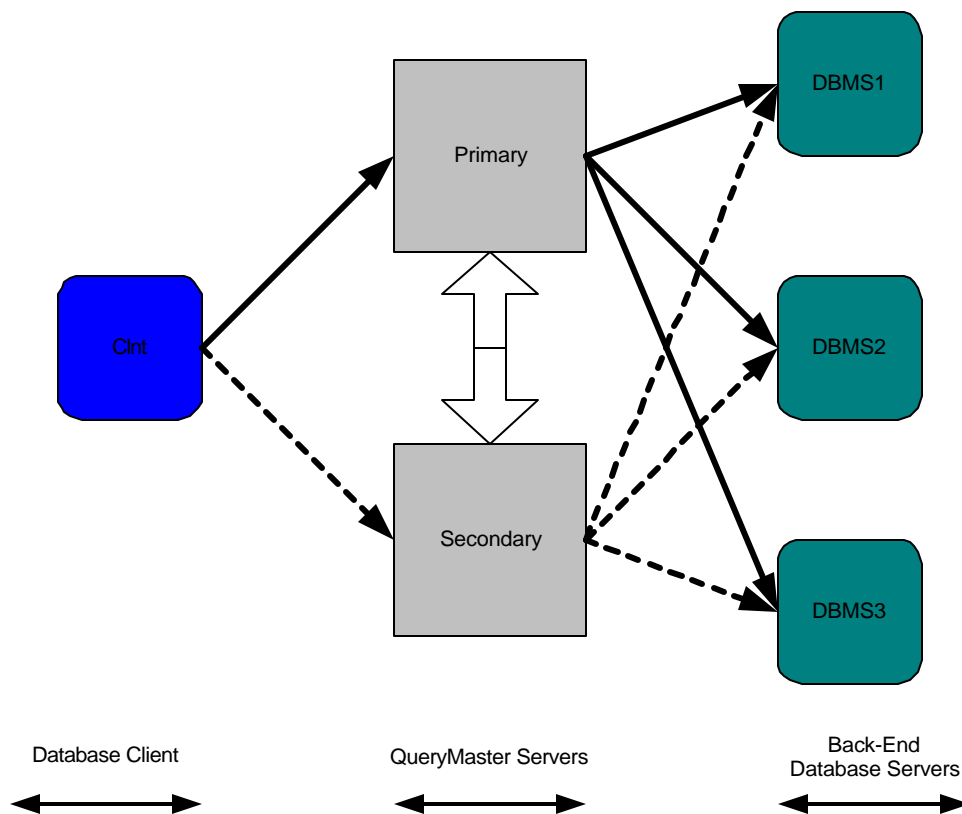
- Intel-based M/B with the CPU and RAM
- Storage components including dual hard disks
- Optical devices including CD-ROM and/or DVD-ROM
- Fast Ethernet/Gigabit Ethernet network interface cards
- Rack mount enclosure
- Redundant power supplies
- Redundant cables and fans

## Software

- Linux operating system
- TierFleet software bundle including
  - Enterprise core software
  - Database replication and synchronization software
  - Web-based management software

### 2.3 QUERYMASTER PG1000 AND PG2000 PRODUCTS DEPLOYMENT

The deployment of the products is best illustrated by means of an example. Figure 4 displays the example set up. In the example scenario, a database client such as a business logic server connects to a single logical database server. The logical server consists of two QueryMaster PG2000 servers and three PostgreSQL database servers. The active QueryMaster server represents the physical connectivity front end for the logical server. The database client, hence, connects to the primary QueryMaster server using one of the standard or proprietary database connectivity tools.



**Figure 4:** An illustration of QueryMaster product deployment

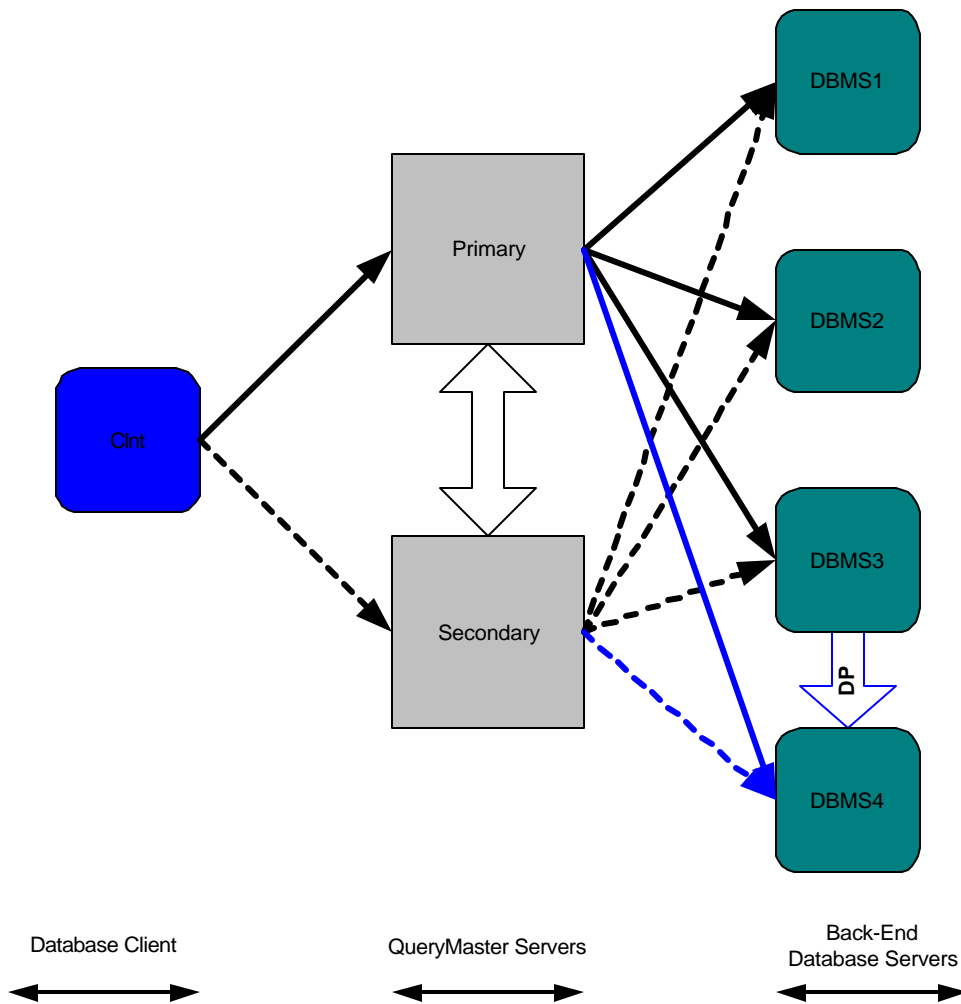
Examples of such tools are JDBC, ODBC, or proprietary PostgreSQL PSQL tool. The active QueryMaster server keeps its own database connections to the back-end database servers transparent from the client. The solid arrows in figure 4 illustrate the connections from the database client to the active QueryMaster server and the connections from the active QueryMaster server to the back-end database servers. The number of database clients is irrelevant to the operation of the model and has been chosen to be one for the sake of simplicity. The database client is always under the impression that the QueryMaster server is the database itself accepting standard database connections followed by SQL queries and returning query results. In addition, there is no difference between the deployment of PG1000 and PG2000 models from the standpoint of database client as it only communicates with a logical database server with specified network settings of a QueryMaster server at all times.

In case of using PG1000 product the logical database server settings are those of single QueryMaster server. In case of using PG2000 product, the logical database server settings are those of the primary QueryMaster server when it is healthy and the secondary QueryMaster server otherwise. In either case the settings of the logical server will always remain the same. The management and configuration information of PG2000 QueryMaster servers remains synchronized at all times while they are both healthy. The QueryMaster servers constantly monitor the health status of each other making use of a low profile internal ping process. The network settings of the primary QueryMaster server including the IP address, fully qualified domain name, and the rest of the parameters are immediately transferred to the secondary QueryMaster server as the result of having any interruption in the availability of the primary server. The dashed arrows in figure 4 illustrate the connections associated with the secondary QueryMaster server that will replace the connections associated with the primary QueryMaster server in case of primary failure. Consequently, in the worst case scenario the database client may only observe a broken database connection that is automatically re-established after timing out. In that situation, the database client does not notice any change in the settings of the database server it is connecting to although it is connecting to a different physical server.

For the example scenario of figure 4, there are originally 3 back-end database servers utilized by the primary or in case of primary failure, secondary QueryMaster server. For the example scenario we call this server, the active server. The database client neither interfaces with any of the back-end database servers nor is aware of their existence. With the arrival of a SQL query, the active QueryMaster server selects the back-end database server with the lowest load and assigns the SQL query to that node. The back-end database server with the lowest load is determined relying on one of the several load balancing algorithms selected by the system administrator. QueryMaster servers offer a number of such algorithms. The choice of a specific algorithm highly depends on the distribution of SQL queries in the target environment. For environments with uniform distribution of the queries a simple algorithm such as random, round-robin, or weighted round-robin may be selected. For environments with non-uniform distribution of the queries a more sophisticated algorithm identifying the back-end database server with the least number of active queries, the least number of active connections, the fastest response time, or the normalized fastest response time may be selected. In any case, the selected algorithm may be changed dynamically if the distribution of the queries in the environment changes. The active QueryMaster server constantly monitors the health status of the back-end database servers and identifies any faulty back-end server immediately. Consequently, the active QueryMaster server only assigns SQL queries to the healthy back-end database servers.

The SQL queries may be simple standalone queries or complex SQL compounds such as database join operations and prepared statements. Nevertheless, if the query is a read, it is simply processed by the selected back-end server and the potential result is returned to the database client after processing. However, processing update SQL queries is more complicated due to the fact that it changes the data content of the back-end database servers. Any changes made to the data content of one of the back-end database servers has to be identically made to all of the other online back-end database servers in order to preserve the unified view of data among the back-end database servers. The active QueryMaster server enforces an atomic all-or-none update mechanism that ensures the data content of the entire back-end database servers remain synchronized at all times. The active QueryMaster server makes use of connection reuse and connection pooling to enhance the performance when communicating with both database clients and back-end database servers.

The following discussion describes dynamic scalability scenario. Lets assume that the objective is to scale the solution by increasing the number of back-end database servers. Figure 5 illustrates the scenario for the example system of figure 4.



**Figure 5:** An illustration of scalability for a solution managed by QueryMaster product

In this scenario the system administrator prepares a standalone fresh database server indicated by DBMS4 in figure 5 and plugs the server into the network environment of the solution. Once the new database server is up and running, the system administrator can access it from the active QueryMaster server management utility by creating a new back-end database server record including the new server settings. This information is also sent to the other QueryMaster server to keep the management information consistent. The system administrator next has to proceed with synchronizing the data contents of the new back-end database server with the existing three other database servers indicated by DBMS1, DBMS2, and DBMS3 in figure 5. Although the operation is controlled and initiated from active QueryMaster server management utility, the data content migrates from the existing database server with the lowest load to the new database server. Data path DP in figure 5 indicates the flow of data migration. The solution remains operational during data migration process and any new changes happened after the start of migration process is inserted into the data content of DBMS4 before the migration process ends. After the DBMS4 data content is synchronized, its availability is announced. In order for DBMS4 to operate as a component of the solution, the system administrator has to bring the database server online by issuing an explicit command from the QueryMaster management utility. After the command is issued, the active QueryMaster server will bring the new database server online at a point in time when there is no active transaction in progress. The active QueryMaster server proceeds with establishing new connections with the new database server once it is online. The solid blue arrow in figure 5 indicates the new connections.

Note that the above scenario remains the same from the functionality standpoint if the system administrator is forced to replace a faulty back-end database server or take it offline, repair it, and return it back online.

### **3. CONCLUSION**

TierFleet is introducing an overall database solution based on a distributed architecture known as a redundant array of independent computers (RAIC). TierFleet RAIC architecture consists of QueryMaster appliance servers and standard back-end database servers. From the standpoint of database clients, TierFleet database solution appears as one logical database server. The range of database clients includes but is not limited to middle-tier business logic servers, browser clients, and legacy software solutions interacting with back-end database systems. The self-contained logical database server is very easy to set up and maintain. The logical server can dynamically scale proportional to the growth of the business it is serving by adding extra servers when needed. The logical server has no single point of failure meaning that it remains operational in case of facing failure in any of its server components. Additionally, deployment of a full-blown database solution based on TierFleet architecture does not introduce any change in the regular operation of the components interfacing with standard database systems. Deploying such an architecture impacts one of the most important metrics of the return of investment (ROI), the cost of down time. It does so by reducing the down time of a business while allowing for proportional future scalability.